

Wykład 10. Prywatne Pozyskiwanie Danych

Wykładowca: Stefan Dziembowski

Skryba: Marcin Szuppe & Piotr Jędrzejewski

Streszczenie. W poniższych notatkach do wykładu wprowadzone zostanie pojęcie Prywatnego Pozyskiwania Danych (PIR) i własności obwodów logicznych. Opisany zostanie protokół podstawowy i różne jego implementacje. Dalsza część notatek poświęcona jest optymalizacji przedstawionych rozwiązań. Końcowy fragment daje przykłady innego podejścia do tematu oraz zawiera bibliografię prac poświęconych przedstawionemu zagadnieniu.

1 Wstęp

Notatki do niniejszego rozdziału zostały opracowane na podstawie prac [4,7]. Będziemy omawiać protokoły *Prywatnego Pozyskiwania Informacji* (ang: *Private Information Retrieval (PIR)*). Jest to stosunkowo świeży pomysł, zapoczątkowany w 1995 roku na konferencji FOCS95. W 1998 sformalizowano zasady *PIR*'u.

Nieformalnie mówiąc, mamy użytkownika i bazę danych. Użytkownik pobiera z bazy pewne informacje. Proces ten musi jednak spełniać następujący warunek: baza danych nie może wiedzieć o co pyta użytkownik chociaż ma dać odpowiedź. Jest tak np. gdy użytkownik chce mieć zapewnioną prywatność swoich danych (np. z indywidualnych względów). Jako przykład można podać bazę danych zawierającą kursy akcji. Użytkownik pyta o kurs konkretnej akcji - nie chce jednak aby ktokolwiek wiedział, jakimi akcjami się interesuje.

Ścisłej rzecz ujmując, dany jest wektor $\vec{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$ złożony z n bitów, znajdujący się w bazie danych. Dla uproszczenia zakładamy, że użytkownik posiada numer i i chce poznać i -ty element (bit) wektora \vec{x} .

Powyższy problem posiada rozwiązanie trywialne: baza może wysłać wszystko do użytkownika (cały wektor \vec{x}). Jest to rozwiązanie iście drogie, bowiem koszt tej operacji jest liniowy względem n , a przecież n może być duże. Zatem szukamy rozwiązań

tańszych, nietrywialnych. Założenie o ograniczeniu kosztu do $n-1$ sprowadza się już do nietrywialnego rozwiązania.

Pierwotnie, w 1995 roku, zdefiniowano protokół i podano jego implementację dla przypadku, gdy istnieją przynajmniej dwa serwery o tej samej zawartości, które nie mogą się komunikować między sobą. W tym przypadku *PIR* był możliwy i bezpieczny w terminach teorii informacji.

W dalszej części wykładu będziemy zakładali, że mamy w posiadaniu tylko jedną bazę danych. Opierać się będziemy na pracy Kushilevitz'a i Ostrovsky'ego [7] z 1997 roku.

2 Formalizacja

Rozważmy parę $(\mathcal{U}, \mathcal{DB})$ interaktywnych, zrandomizowanych i wielomianowych maszyn Turinga, gdzie

- \mathcal{DB} jest bazą danych
- \mathcal{U} jest użytkownikiem.

Niech 1^k będzie parametrem bezpieczeństwa. Intuicyjnie jest to długość reprezentacji bitowej liczb, posiadających właściwości gwarantujące trudną obliczalność pewnych problemów teorioliczbowych (zwykle k jest pewną funkcją n , np. n^c dla pewnego $c > 0$).

Schemat działania protokołu:

- \mathcal{DB} pobiera na wejściu parę $(1^k, \vec{x})$, gdzie $\vec{x} \in \{0, 1\}^n$, gdzie n jest wielomianowe w k .
- \mathcal{U} pobiera na wejściu trójkę $(1^k, 1^n, i)$, gdzie $i \in 1, \dots, n$ a 1^n jest parametrem randomizacyjnym.

Interakcja między partiami wygląda następująco:

1. \mathcal{U} wysyła zapytanie $q(i)$ (wygenerowane na podstawie parametrów wejścia) do \mathcal{DB} (pamiętamy, że użytkownik jest zrandomizowany)
2. \mathcal{DB} przetwarza zapytanie i wysyła do \mathcal{U} odpowiedź $a(x, q(i))$.

Teoretycznie można rozważać protokoły z większą liczbą rund. My jednak będziemy rozważali te z jedną rundą (jak powyżej).

Po powyższej interakcji i przetworzeniu odpowiedzi z \mathcal{DB} , na końcu \mathcal{U} zwraca x'_i .

Wymagane są następujące warunki:

POPRAWNOŚĆ: $x_i = x'_i$

PRYWATNOŚĆ: (użytkownika) Jest trudniejsza do zdefiniowania. Zakładamy, że \mathcal{DB} ma ograniczoną moc obliczeniową (bezpieczny obliczeniowo PIR).

Dla dowolnej wielomianowej rodziny obwodów logicznych C_k

$$\max_{i,j} |P[C_k(q(i)) = 1] - P[C_k(q(j)) = 1]|$$

jest zaniedbywalne w k (bliskie zeru).

Przypomnienie: Subtelną różnicą między maszyną Turinga a obwodem logicznym jest to, że maszyna Turinga jest taka sama dla każdego rozmiaru wejścia, obwód jest natomiast budowany dla danego rozmiaru wejścia k . Mówimy zatem o rodzinie obwodów indeksowanych długością danych wejściowych. Pamiętamy też, że obwody mogą obliczać nawet nieobliczalne funkcje (są mocniejsze od maszyn Turinga).

Inaczej mówiąc, jeśli pewna funkcja $i \rightarrow |C_i|$, gdzie $|C_i|$ jest wielomianowe, jest obliczalna przez wielomianową maszynę Turinga, to też jest obliczana przez pewien wielomianowy obwód. Ta implikacja nie zachodzi jednak w drugą stronę.

Intuicja: Dowolny wielomianowy obwód nie potrafi odróżnić $q(i)$ od $q(j)$, czyli chodzi o to, że baza danych (o ograniczonej mocy obliczeniowej) nie potrafi odróżniać zapytania $q(i)$ od $q(j)$. W niniejszym wykładzie będziemy posługiwali się obwodami logicznymi zamiast maszynami Turinga, właśnie ze względu na ich użyteczność.

NIETRYWIALNOŚĆ: Niech

$$CC(m, k) := \max(|q(i)| + |a(x, q(i))|)$$

oznacza *złożoność komunikacyjną protokołu* $(\mathcal{U}, \mathcal{DB})$ (gdzie maksimum przebiega po (x, i) i losowych inputach stron).

Patrzmy zatem na najdłuższą wiadomość, jaka może zostać wysłana w jedną i w drugą stronę, i bierzemy sumę tych długości.

Wymagamy, by zachodził warunek

$$CC(n, k) < n$$

(przynajmniej dla pewnych wartości n i k).

2.1 Uwagi

1. Nie interesuje nas prywatność bazy danych (choć w literaturze są rozważane także przypadki z uwzględnieniem prywatności bazy danych).
2. W literaturze pojawiają się też protokoły o większej liczbie rund.
3. Także zdarzają się protokoły o większej liczbie serwerów baz danych (które nie mogą się ze sobą komunikować). Pierwszy protokół PIR [4] zakładał istnienie co najmniej dwóch serwerów.

3 Założenia złożonościowe są konieczne

Pokażemy, że założenie o wielomianowości rodziny obwodów (że rodzina ta jest ograniczona obliczeniowo) jest konieczne (w przypadku protokołów z jedną bazą danych).

Niech *IT-PIR* oznacza protokół PIR, w którym definiując prywatność nie zakładamy, że rodzina obwodów jest ograniczona obliczeniowo. Dla uproszczenia zakładamy też, że błąd wynosi 0, czyli

$$\max_{i,j} |P[C_k(q(i)) = 1] - P[C_k(q(j)) = 1]| = 0.$$

Rozważać zatem będziemy taki teoretyczny idealny PIR.

Mamy następujący fakt (patrz też [4], Rozdział 5.1).

Twierdzenie 1 *IT-PIR (nietrywialny i z jednym serwerem) nie istnieje.*

Zanim jednak podamy dowód powyższego twierdzenia, wprowadźmy kilka definicji:

Definicja 1 *Transkryptem komunikacyjnym C między partiami* nazywamy raport komunikatów pomiędzy użytkownikiem a bazą danych.

Inaczej: Transkrypt jest funkcją $C(r_U, i, r_D, x)$, gdzie i, x są odpowiednio inputami użytkownika i bazy danych a r_U i r_D odpowiednio inputami zrandomizowanymi.

Definicja 2 Transkrypt C jest *możliwy dla (x, i)* jeśli istnieją r_U i r_D takie, że $C = C(r_U, i, r_D, x)$.

Nieformalnie: Prawdopodobieństwo pojawienia się transkryptu C dla danych i, x, r_U i r_D nie jest zerowe.

Definicja 3 Transkrypt C jest *możliwy dla i* , jeśli istnieje taki x , że transkrypt C jest możliwy dla (x, i) .

Dowód:

1. Ustalamy indeks i oraz zakładamy, że istnieje nietrywialny IT-PIR.
2. Dla dowolnego indeksu i z nietrywialności wynika, że możliwych transkryptów dla i jest mniej niż 2^n , gdzie n to rozmiar bazy danych (ponieważ długość transkryptu może być co najwyżej n).
3. Zatem dla ustalonego indeksu i istnieją takie x, y i transkrypt C , że $x \neq y$ oraz C jest możliwy dla (x, i) i (y, i) .
4. Ze względu na założenie o prywatności i o tym, że przeciwnik jest nieograniczony obliczeniowo mamy, iż C musi być możliwe dla (x, j) i (y, j) .
5. Nieformalnie: Baza danych to przeciwnik (z punktu widzenia bazy $i \neq j$). Jeśli przeciwnik zobaczył C to wie, że istnieją $x \neq y$ takie, że (x, i) i (y, i) są możliwe. Co by było, gdyby dla pewnego j albo (x, j) albo (y, j) było niemożliwe? Jeśli C byłoby niemożliwe dla (x, j) natomiast możliwe dla (x, i) i (y, i) , to jeśli baza zobaczyła komunikację C , to wie, że inputem użytkownika nie mogło być j . Czyli jeśli C jest możliwy dla (x, i) , to też musi być możliwy dla dowolnego (x, j) . Analogicznie rozumujemy z y .
6. W szczególności, ponieważ $x \neq y$, istnieje takie j , że $x_j \neq y_j$.
7. A teraz ponieważ output użytkownika zależy od transkryptu C oraz od j , to uzyskujemy sprzeczność. (np. gdy $x_j = 0$ i $y_j = 1$, to użytkownik zwraca różne wyniki).

□

Uwaga: W pracy [4] pokazano m.in. IT-PIR z dwiema bazami danych (które nie mogą się komunikować) i złożonością komunikacyjną $\mathcal{O}(n^{1/3})$

4 Protokół podstawowy

Przedstawione podejście do problemu PIR polega na zastosowaniu reszt kwadratowych i ich własności. Na początku przypomnimy podstawowe narzędzia przydatne do sformułowania implementacji protokołów.

4.1 Quadratic Residuosity Assumption

Niech $N \in \mathbf{N}$.

fakt: Zdefiniujmy:

$$Z_N^* = \{x \mid 1 \leq x \leq N, \text{NWD}(N, x) = 1\}$$

fakt: $Q_N(y)$ (ang.: *Quadratic Residuosity Predicate*) definiujemy następująco:

$$Q_N(y) = \begin{cases} 0 & \text{gdy } y \text{ jest resztą kwadratową modulo } n \\ 1 & \text{wpp.} \end{cases}$$

Dokładniej:

$$Q_N(y) = \begin{cases} 0 & \text{gdy } \exists w \in Z_N^*, \text{ takie że } w^2 = y(\text{mod}N) \\ 1 & \text{wpp.} \end{cases}$$

Mówimy, że y jest resztą kwadratową ((ang.: *quadratic residue*) - QR) modulo N , jeśli $Q_N(y) = 0$ oraz że y jest nie-resztą kwadratową ((ang.: *quadratic non-residue*) - QNR) jeśli $Q_N(y) = 1$.

fakt: Problem sprawdzania czy liczba jest resztą kwadratową modulo N jest określany jako najtrudniejszy ("hardest"), jeśli N jest iloczynem dwóch różnych liczb pierwszych długości $k/2$. Zdefiniujmy tzw. "hard set" (zbiór dobrych N) generowany przez k jako:

$$H_k = \{N \mid N = p_1 \cdot p_2, \text{ gdzie } p_1, p_2 \text{ są } k/2\text{-bitowymi liczbami pierwszymi}\}$$

fakt: Jeśli faktoryzacja $N \in H_k$ jest znana, obliczenie $Q_N(y)$ można wykonać w czasie $\mathcal{O}(|N^3|)$. Oznaczmy przez $(\frac{y}{N})$ symbol Jakobiego. Dla wszystkich $N \in H_k$ wartość $(\frac{y}{N})$ może być wyznaczona w czasie wielomianowym względem $|N|$, nawet bez informacji o faktoryzacji N . Dla $N \in H_k$, jeśli $(\frac{y}{N}) = -1$, wtedy y jest zawsze QNR, jeśli natomiast $(\frac{y}{N}) = +1$, to dokładnie połowa y -ków to QNR, a druga połowa to QR. Zdefiniujmy zatem:

$$Z_N^{+1} = \{y \in Z_N^* \mid (\frac{y}{N}) = 1\}$$

fakt: Warto dodać, że aby otrzymać losową QR ze zbioru Z_N^{+1} wystarczy wziąć losowe $r \in Z_N^*$ i obliczyć $r^2(\text{mod}N)$, do czego nie jest potrzebna wiedza o faktoryzacji N .

Definicja 4 Quadratic Residuosity Assumption (QRA):

Dla dowolnej stałej c i dowolnej wielomianowej rodziny obwodów logicznych $C_k(\cdot, \cdot)$ istnieje stała $K \in \mathbf{Z}$, taka że dla wszystkich $k > K$ zachodzi:

$$P[C_k(N, y) = Q_N(y)] < \frac{1}{2} + \frac{1}{k^c}$$

gdzie $N \in H_k$, $y \in \mathbf{Z}_N^{+1}$. Albo inaczej mówiąc, różnica:

$$P[C_k(N, y) = Q_N(y)] - \frac{1}{2}$$

jest zaniedbywalna w k .

4.2 Pierwszy (zły) pomysł

Rozważmy następujący protokół.

1. \mathcal{U} wybiera losowe liczby pierwsze p_1 i p_2 o długości $k/2$ i wysyła $N = p_1 \cdot p_2$ do \mathcal{DB} .
2. \mathcal{U} tworzy wektor $\vec{y} \in (\mathbf{Z}_N^{+1})^n$ taki, że dla każdego j :
 jeśli $j \neq i$, to y_j jest losowym elementem takim, że $Q_N(y_j) = 0$
 a jeśli $j = i$, to y_j jest losowym elementem takim, że $Q_N(y_j) = 1$
 (zatem wektor \vec{y} składa się z n losowych reszt kwadratowych oprócz i -tego elementu) i wysyła \vec{y} do \mathcal{DB} .
3. Dla każdego j \mathcal{DB} oblicza

$$w_j := \begin{cases} y_j^2 & \text{jeśli } x_j = 0; \\ y_j & \text{jeśli } x_j = 1. \end{cases}$$

a następnie iloczyn

$$\prod_{j=1}^n w_j$$

i wysyła go do \mathcal{U} .

Nieformalnie: Chodzi o to, że \mathcal{U} wysyła w wektorze \vec{y} prawie same reszty kwadratowe. \mathcal{DB} go dostaje i po kolei bierze jego elementy patrząc na swój wektor \vec{x} . Liczy iloczyn wartości w_j i wysyła do \mathcal{U} . Jeżeli na i -tej pozycji wektora \vec{x} stała jedynka ($x_i = 1$), to iloczyn jest QNR, w przeciwnym przypadku iloczyn jest QR (nie-reszta kwadratowa podniesiona do kwadratu da resztę kwadratową).

4. Użytkownik \mathcal{U} zna faktoryzację N , zatem jest w stanie wyznaczyć czy otrzymana wartość to QR czy QNR i może wyliczyć poszukiwany bit:

$$x_i = \begin{cases} 0 & \text{dla QR;} \\ 1 & \text{dla QNR.} \end{cases}$$

Złożoność komunikacyjna:

- co prawda $a(x, q(i))$ to tylko jeden element ciała (k bitów), ale
- $q(i)$ to aż n elementów ($n \cdot k$ bitów) (z grubsza)
- Zatem użytkownik wysłał $n \cdot k$ i odebrał k bitów. W sumie przesłanych zostało $(n + 1)k$ bitów, co jest zwykle dużo większe niż poszukiwane n bitów.

Lemat 1 *Powyższy protokół spełnia warunek prywatności z definicji PIR.*

Dowód:(przez zaprzeczenie)

Załóżmy przez zaprzeczenie, że dla pewnych indeksów i, j baza danych \mathcal{DB} potrafi rozróżnić zapytania dotyczące elementu (bitu) i od zapytania dotyczącego j .

Oczywistym jest, że z opisu protokołu wynika że $i \neq j$ - gdyby było inaczej ($i = j$), to ponieważ protokół zachowałby się identycznie na tych zapytaniach nie byłoby możliwości ich rozróżnienia.

Fakt, że baza danych potrafi rozróżnić te zapytania oznacza, że istnieje algorytm (bardziej precyzyjnie - wielomianowa rodzina obwodów) D , taki że jeśli D otrzyma od użytkownika zapytanie (N, \vec{y}) (liczba $N \in H_k$, \vec{y} - wektor QR/QNR długości n) wygenerowane dla indeksu i , to wtedy wykonanie D zakończy się wynikiem 1 z pewnym prawdopodobieństwem, powiedzmy $p + \epsilon$, gdy tymczasem D na zapytaniu wygenerowanym dla indeksu j zwróci 1 z prawdopodobieństwem p . inaczej mówiąc:

$$P[D(q(i)) = 1] - P[D(q(j)) = 1] = \epsilon$$

dla pewnego $\epsilon > 0$.

Zgodnie z konstrukcją zapytania, składa się ono z liczby N wybranej ze zbioru H_k oraz wektora \vec{y} złożonego z liczb należących do Z_N^{+1} , spośród których wszystkie są QR (resztami kwadratowymi $mod N$), za wyjątkiem liczby znajdującej się na pozycji i w przypadku pierwszego zapytania lub pozycji j dla drugiego zapytania. Na tych wyróżnionych pozycjach znajdują się liczby QNR (nie-reszty kwadratowe $mod N$).

Opierając się na powyższych założeniach konstruujemy obwód logiczny C (wielomianowej wielkości względem $|D|$), który na wejściu otrzymuje dwie liczby (N, y) , z których $N \in H_k$ a $y \in Z_N^{+1}$. Wynika z tego, że y jest QNR z prawdopodobieństwem $1/2$. Zadaniem obwodu C jest próba wyznaczenia $Q_N(y)$ (ang.: *quadratic residuosity predicate*) z prawdopodobieństwem co najmniej $1/2 + \epsilon/2$.

Obwód C jest jakby otoczką D , a jego zadaniem jest skonstruowanie sekwencji liczb, na którą oczekuje algorytm rozróżniający D . Obwód C działa w sposób następujący:

1. Wybierz $n - 2$ losowe QR i umieść je na wszystkich pozycjach sekwencji $\vec{y} = \{y_1, \dots, y_n\}$ za wyjątkiem i oraz j .
2. Wylosuj jedną z pozycji i lub j i umieść na niej y (parametr wejściowy obwodu C), a na pozostałej pozycji umieść inną wylosowaną wartość QR.
3. Zapuć D na utworzonej sekwencji. Jeśli wybraną w kroku drugim pozycją było i zwróć $D(\vec{y})$ (taką samą wartość jaką zwraca D), jeśli natomiast wybraną pozycją było j zwróć $1 - D(\vec{y})$.

Najpierw obliczmy prawdopodobieństwo, że C zwróci 1 na wejściu (N, y) , takim że y jest QR modulo N . W tym przypadku, nie ma znaczenia którą pozycję wybierzemy w kroku drugim, ponieważ wejście do algorytmu D jest ciągiem N losowych QR ze zbioru Z_N^{+1} . Oznaczmy przez q pewne prawdopodobieństwo, że D zwróci 1 na zadanym wektorze \vec{y} . Prawdopodobieństwo, że C zwróci 1 w tym przypadku wynosi:

$$\frac{1}{2}q + \frac{1}{2}(1 - q) = \frac{1}{2}$$

Teraz obliczmy prawdopodobieństwo, że C zwróci 1 na wejściu (N, y) , takim że y jest QNR modulo N . W tym przypadku wektor \vec{y} zawiera pojedynczą wartość QNR na pozycji i lub pozycji j . Zatem prawdopodobieństwo, że C zwróci 1 w tym przypadku wynosi:

$$\frac{1}{2}P[D(q(i)) = 1] + \frac{1}{2}P[D(q(j)) = 0]$$

co jest równe:

$$\frac{1}{2}(P[D(q(i)) = 1] + 1 - P[D(q(j)) = 1])$$

a to z kolei jest równe:

$$\frac{1}{2}(p + \epsilon) + \frac{1}{2}(1 - p) = \frac{1}{2} + \frac{\epsilon}{2}.$$

Czyli prawdopodobieństwo, że na losowym $y \in Z_N^{+1}$ obwód C zwróci $Q_N(y)$ wynosi:

$$P\left[\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2}\left(\frac{1}{2} + \frac{\epsilon}{2}\right)\right]$$

co jest ostatecznie równe:

$$\frac{1}{2} + \frac{\epsilon}{4}$$

Przeczy to założeniu, że obwód C wylicza $Q_N(y)$ z prawdopodobieństwem co najmniej $1/2 + \epsilon/2$. □

Uczyńmy jeszcze następującą obserwację: jeśli \mathcal{DB} ma wiele różnych ciągów \vec{x} i w każdym z nich \mathcal{U} chce poznać i -ty bit, to nie trzeba za każdy razem generować nowego N i nowego wektora \vec{y} . Można użyć tego samego.

Wiedza bazy danych po wykonaniu zapytania na wektorze \vec{y} nie zmienia się - baza nie może nic stwierdzić o poszczególnych elementach wektora ani o elemencie poszukiwanym przez użytkownika – wynika to bezpośrednio z poprawności algorytmu oraz powyższego dowodu prywatności użytkownika.

4.3 Poprawna konstrukcja

4.3.1 Opis schematu

Zmodyfikujemy poprzednią konstrukcję w taki sposób, że co prawda

$$|a(x, q(i))| \cdot |q(i)| \approx n \text{ elementów ciała}$$

(czyli tak jak poprzednio), ale $|a(x, q(i))| + |q(i)| = 2 \cdot \sqrt{n}$ elementów. Konkretnie, będziemy mieć

$$|a(x, q(i))| \approx |q(i)| \approx \sqrt{n} \text{ elementów.}$$

Czyli: skracamy $q(i)$ kosztem wydłużenia $a(x, q(i))$, ale i tak się opłaca.

Zauważmy także, że protokół z Rozdziału [4.2] chronił także prywatność bazy danych (w każdym razie w modelu pasywnym). Poniższy protokół nie spełnia tego warunku (ale to nie szkodzi).

Poniższy schemat jest bardzo podobny do poprzedniego. W tym przypadku za pomocą prostego triku zmniejsza się złożoność komunikacyjną do $\mathcal{O}(n^{0.5+c})$ dla dowolnego $c > 0$.

Spójrzmy na bazę danych przechowującą wektor \vec{x} jak na macierz $s \times t$ bitów (oznaczymy tę macierz przez M). Użytkownik jest zainteresowany wydobyciem z bazy bitu x_i przy zachowaniu prywatności zgodnie z założeniami protokołu PIR. Zauważmy, że bit x_i znajduje się na pewnej pozycji (a, b) w macierzy M . Schemat działania protokołu jest następujący:

Krok 1 Użytkownik zaczyna od wyboru k -bitowego numeru $N \in H_k$ (wybiera dwie liczby pierwsze długości $k/2$ bitów i mnoży przez siebie). Użytkownik wysyła N do \mathcal{DB} (k bitów).

Krok 2 Użytkownik wybiera niezależnie t losowych liczb $y_1, y_2, \dots, y_t \in \mathbb{Z}_N^{+1}$ w taki sposób, że y_b jest QNR oraz y_j dla każdego $j \neq b$ jest QR. Użytkownik wysyła tych t liczb do \mathcal{DB} (w sumie $t \cdot k$ bitów).

Krok 3 Baza danych wylicza dla każdego wiersza $1 \leq r \leq s$ liczbe $z_r \in Z_N^*$ w następujący sposób:

◇ najpierw oblicza (w Z_N^*):

$$w_{r,j} := \begin{cases} y_j^2 & \text{jeśli } M_{r,j} = 0 \\ y_j & \text{jeśli } M_{r,j} = 1 \end{cases}$$

◇ następnie oblicza s iloczynów:

$$z_r = \prod_{j=1}^t w_{r,j}$$

Faktem jest, że jeśli $j \neq b$ to $w_{r,j}$ jest zawsze QR, natomiast jeśli $j = b$ to $w_{r,j}$ jest QR wtw $M_{r,j} = 0$ i jest QNR w przeciwnym przypadku. Dlatego z_r jest QR wtw $M_{r,b} = 0$ i jest QNR w przeciwnym przypadku.

Krok 4 DB przesyła z_1, z_2, \dots, z_s użytkownikowi (w sumie $s \cdot k$ bitów).

Krok 5 Użytkownik przetwarza tylko liczbę z_a , odpowiadającą wierszowi macierzy M , który zawiera bit interesujący użytkownika. Wartość ta jest QR wtw $M_{a,b} = 0$ i jest QNR w przeciwnym razie. Ponieważ \mathcal{U} zna faktoryzację liczby N może efektywnie sprawdzić czy z_a jest QR i dzięki temu odszyfrować bit $M_{a,b}$.

4.3.2 Złożoność komunikacyjna

Komunikacja w powyższym schemacie składa się z $(s + t + 1)$ k -bitowych liczb $(N, y_1, \dots, y_t, z_1, \dots, z_s)$. Przyjmując $s = t = \sqrt{n}$ otrzymujemy złożoność komunikacyjną:

$$\begin{aligned} |q(i)| &= (\sqrt{n} + 1) \cdot k \\ |a(x, q(i))| &= \sqrt{n} \cdot k \\ CC(n, k) &= (2\sqrt{n} + 1) \cdot k \end{aligned}$$

Jeśli zatem przyjmiemy, że parametr $k = n^c$ dla pewnej stałej $c > 0$ otrzymamy złożoność komunikacyjną równą:

$$n^{\frac{1}{2}+c}$$

Dla $c < 1/2$ powyższa wartość jest mniejsza od n .

4.3.3 Wnioski

Zatem mamy następujące twierdzenie:

Twierdzenie 2 Zakładając QRA istnieją nietrywialne protokoły PIR (z jednym serwerem).

Istnieją też konstrukcje protokołów PIR oparte na innych założeniach. Np. w pracy [2] pokazano protokół o złożoności polilogarytmicznej (ale oparty na specjalnie dobranym założeniu złożonościowym).

Jeśli chodzi o wyniki teoretyczne, to wiadomo, że założenie o istnieniu PIR (nietrywialnego i z jednym serwerem) jest równoważne założeniu o istnieniu transferu utajnionego [5].

4.4 Uzupełnienie

4.4.1 Założenia

Twierdzenie 3 *Dla każdego $c > 0$ istnieje model z pojedynczą bazą danych, w którym działa protokół PIR ze złożonością komunikacyjną pomiędzy użytkownikiem a bazą danych $O(n^c)$ oparty na trudności wyznaczania reszt kwadratowych (ang.: Quadratic Residuosity Assumption (QRA)).*

Protokół ten ma poniższe własności:

- ◇ Dane są przechowywane w bazie danych w ich naturalnej postaci (tzn. każdy napis x jest trzymany jako napis x w nieprzekształconej formie). Dzięki temu nie jest konieczny żaden pre-procesing (wstępne przetwarzanie danych), przechowywanie dodatkowych informacji lub koordynacja pomiędzy użytkownikami. W tym modelu nie jest możliwe jednak prywatne pisanie do bazy danych - baza może obserwować bezpośrednio zmiany zawartości x .
- ◇ Protokół działa jako schemat jednorundowy tak jak zakładaliśmy do tej pory), czyli *zapytanie(query) \longleftrightarrow odpowiedź(answer)*, co ma duże znaczenie dla bazy danych (przepustowość, obciążenie).

4.4.2 Opis schematu

W kroku 4 przedstawionego w poprzednim paragrafie schematu \mathcal{DB} wysyła s k -bitowych liczb z_1, \dots, z_s do użytkownika, mimo iż Użytkownik jest zainteresowany tylko jedną z tych liczb z_a . Możemy spojrzeć na ten ciąg liczb jak na słowo długości $s \cdot k$ bitów. Jeśli dodatkowo ustawimy liczby z_i jedna pod drugą otrzymamy k nowych liczb (słów), złożonych z kolejnych bitów każdej z liczb z_i . Teraz do każdego z tych słów możemy zastosować zmodyfikowaną wersję podanego wcześniej schematu, aby wydobyć z każdego z k -słów poszukiwany bit z pozycji a .

Bardziej formalnie - oznaczmy przez S_1 przedstawiony powyżej schemat. Następnie zdefiniujemy rekurencyjnie schemat S_ℓ . Oznaczmy przez n_ℓ liczbę bitów które przetwarzamy na poziomie rekursji S_ℓ (na najwyższym poziomie, oznaczanym przez

L , mamy $n_L = n$ bitów). Oznaczmy przez t_ℓ liczbę kolumn w macierzy używanej w schemacie S_ℓ do reprezentowania napisu (oraz oznaczmy $s_\ell = n_\ell/t(\ell)$).

Realizujemy S_ℓ poprzez zamianę kroku 4 powyższego schematu następującym:

Krok 4 \mathcal{U} i \mathcal{DB} wykonują k razy schemat $S_{\ell-1}$. Po każdym wykonaniu użytkownik dostaje jeden z bitów liczby z_a pobrany z napisu długości $n_{\ell-1} = k \cdot s_\ell$ przechowywanego w bazie danych. Po wykonaniu schematu k razy użytkownik posiada całą liczbę z_a , która pozwala mu na rekonstrukcję poszukiwanego bitu (identyczną metodą co w kroku 5-tym podanego poprzednio schematu).

Pomimo tego, że kolejne sekwencje bitów w poszczególnych krokach rekursji stają się coraz mniejsze, nie można zmniejszyć parametru bezpieczeństwa 1^k . Z drugiej strony krok 1 wystarczy wykonać tylko raz, a uzyskaną w ten sposób liczbę N można używać na wszystkich poziomach rekursji.

Powyższą nową wersję kroku 4 schematu komunikacji można jeszcze usprawnić. Ponieważ kiedy wykonywany jest k razy schemat $S_{\ell-1}$ aby pozyskać liczbę z_a oczywistym jest, że podczas pierwszego wykonania $S_{\ell-1}$ użytkownik uzyskuje najbardziej znaczący bit z_a , podczas drugiego wykonania - drugi najbardziej znaczący bit z_a , itd. Dlatego po każdym wykonaniu baza danych może stać się mniejsza o czynnik k . Dostajemy drugą wersję kroku 4:

Krok 4 \mathcal{U} i \mathcal{DB} wykonują k razy schemat $S_{\ell-1}$. W d -tym wykonaniu użytkownik dostaje d -ty najbardziej znaczący bit liczby z_a pobrany z napisu długości $n_{\ell-1} = s_\ell$ przechowywanego w bazie danych (napis ten zawiera d -ty bit każdej z liczb z_1, \dots, z_{s_ℓ}). Po wykonaniu schematu k razy użytkownik posiada już całą liczbę z_a , która pozwala mu na rekonstrukcję poszukiwanego bitu (identyczną metodą co w kroku 5-tym podanego poprzednio schematu).

Jeśli teraz spojrzymy na dowolny poziom j schematu rekursji, to w każdym wykonaniu S_j użytkownika interesuje ten sam indeks napisu, z którego są pobierane dane (indeks ten zależy od i - pozycji szukanego bitu w słowie x z najwyższego poziomu rekursji oraz od bieżącego poziomu rekursji j). Dlatego wykonujemy następującą modyfikację:

◊ Dla każdego poziomu j rekursji, \mathcal{U} wysyła pojedyncze zapytanie (złożone z t_j liczb). To zapytanie posłuży we wszystkich wykonaniach S_j (pełni ono rolę zapytania obliczonego w kroku 2 pierwotnego schematu, tylko że dla j -tego poziomu rekursji).

4.4.3 Złożoność komunikacyjna

Na początku założymy, że dla każdego ℓ mamy taką samą wartość $t_\ell = n^{\frac{1}{L+1}}$ jako parametr.

Wynika z tego, że jeśli na najwyższym poziomie rekursji używamy schematu S_L na bazie danych rozmiaru $n_L = n$ to przez indukcję (po $\ell = L, L - 1, \dots, 1$) otrzymamy:

$$n_\ell = n^{\frac{\ell+1}{L+1}}$$

Rozważmy teraz kolejne kroki rekursji. Na najwyższym poziomie \mathcal{U} wysyła zapytanie do \mathcal{DB} , która przygotowuje k słów (napisów, stringów) przetwarzając to zapytanie, ale nie wysyła żadnego bitu do użytkownika. Potem, na kolejnym poziomie, \mathcal{U} znowu przesyła zapytanie - tym razem \mathcal{DB} znów przygotowuje z każdego stringa z poprzedniego poziomu rekursji nowe k stringów (co razem daje k^2 słów na drugim poziomie) i tak dalej. W końcu, na ostatnim poziomie rekursji użytkownik znów przesyła zapytanie, przetwarzając które, baza danych przygotowuje k nowych stringów dla każdego ze słów przechowywanych w bazie powstałych w poprzednim kroku. Daje to w sumie k^L napisów długości $n^{\frac{1}{L+1}}$ każdy.

Dzięki takiemu podejściu widać, że powyższy schemat może zostać zaimplementowany jako schemat pojedynczy typu zapytanie-odpowiedź. Można również łatwo zanalizować złożoność komunikacyjną schematu:

Na każdym z L poziomów rekursji \mathcal{U} wysyła $n^{\frac{1}{L+1}}$ k -bitowych liczb. Baza danych odpowiada tylko na ostatnim poziomie, kiedy wysyła do Użytkownika k^L napisów długości $n^{\frac{1}{L+1}}$ każdy. W sumie złożoność komunikacyjna wynosi:

$$\text{CC}(n, k) = n^{\frac{1}{L+1}} \cdot (k^L + L \cdot k)$$

Jeśli teraz przyjmiemy, że $L = \mathcal{O}(\sqrt{\log n / \log k})$ (jest to wartość dla której $n^{\frac{1}{L+1}}$ oraz k^L stają się równe) dostaniemy złożoność równą:

$$\text{CC}(n, k) = 2^{\mathcal{O}(\sqrt{\log n \cdot \log k})}$$

Jeśli na przykład $k = n^c$ dostaniemy złożoność $n^{\mathcal{O}(\sqrt{c})}$, a jeśli $k = \log^c n$ dostaniemy złożoność: $2^{\mathcal{O}(\sqrt{\log n \cdot \log \log n})}$.

4.4.4 Prywatność

Dowód jest podobny do dowodu z poprzedniej sekcji. Znowu założymy przez zaprzeczenie, że istnieje algorytm rozróżniający, który dla pewnych i, j odróżnia rozkład dla zapytania $q(i)$ od rozkładu dla zapytania $q(j)$ z pewnym prawdopodobieństwem ϵ . Jeśli rozważymy rekursję w powyższym schemacie zobaczymy, że zapytanie wysyłane przez użytkownika jest liczbą N , za którą występuje sekwencja liczb z_1, z_2, \dots, z_β ze zbioru Z_N^{+1} , gdzie $\beta = L \cdot n^{\frac{1}{L+1}}$.

Oznaczmy dla indeksu i przez I_i podzbiór $\{1, \dots, \beta\}$, wszystkich tych pozycji, które zawierają QNR (zgodnie z protokołem dla każdego indeksu i podzbiór I_i jest wyznaczony i jest niezależny od losowych wyborów dokonywanych przez użytkownika).

Tak jak poprzednio $i \neq j$, gdyż w innym przypadku protokół dla obu wartości byłby identyczny i nierozróżnialny.

Tak jak poprzednio oznaczmy przez D obwód logiczny (algorytm) rozróżniający $q(i)$ od $q(j)$. Jego definicja jest analogiczna do definicji z poprzedniego paragrafu.

Tak jak poprzednio konstruujemy obwód C , którego zdaniem jest obliczenie "quadratic residuosity predicate" Q_N . C na wejściu dostaje (N, y) i działa następująco:

- Wybierz losowo jeden z podzbiorów I_i lub I_j . Na każdej pozycji nie należącej do wybranego zbioru umieść $QR \in Z_N^{+1}$. Na każdej pozycji należącej do zbioru umieść iloczyn y i losowej liczby QR .
- Uruchom D na utworzonej w ten sposób sekwencji. Jeśli wybranym zbiorem jest I_i na wyjściu zwróć tą samą wartość co $D(q(i))$, jeśli wybraną pozycją jest I_j zwróć $1 - D(q(j))$.

Jeśli y jest QR to cała sekwencja złożona z β elementów jest złożona z samych QR. Jeśli natomiast y jest QNR to otrzymana sekwencja ma rozkład taki jak $q(i)$ lub $q(j)$. Pozostała część dowodu jest identyczna jak w poprzednim paragrafie.

4.4.5 Wnioski

Twierdzenie 4 Zakładając trudność obliczeniową QRA , dla każdego $c > 0$ istnieje protokół PIR o złożoności obliczeniowej $\mathcal{O}(n^c)$.

5 Prace o zbliżonej tematyce

5.1 Symetryczny PIR ($SPIR$)

W pracy [6] wprowadzono pojęcie (oraz podano implementacje) Symetrycznego PIR u, to znaczy takiego, w którym zachowana jest także prywatność bazy danych (to znaczy \mathcal{U} poznaje tylko jedną wartość w bazie).

Ma to znaczenie wtedy, kiedy np. dostęp do informacji jest płatny.

Łatwo zauważyć, że definicja $SPIR$ u jest wzmocnieniem definicji transferu utajonego.

5.2 Prywatne Przechowywanie Informacji

W pracy [8] rozważano problem prywatnego przechowywania informacji, tzn. rozszerzano protokół o możliwość *pisania do bazy danych*.

5.3 *Pozyskiwanie bloków i wyszukiwanie słów kluczowych*

Rozważano także problem pozyskiwania całych bloków bitów oraz pytania do bazy danych postaci: „czy dany string występuje w bazie danych”. Patrz np. [3].

5.4 *Praktyczne implementacje*

Prowadzone są badania nad praktycznymi implementacjami protokołów PIR [1].