# Zerowa Wiedza

Franciszek Grzegorek

2 czerwca 2002 roku

## 1 Wprowadzenie

Pojęcie zerowej wiedzy polega na tym, aby przekonać inną osobę do tego, że się posiada jakąś wiedzę bez przekazywania samej wiedzy.
W problemie tym występują dwie osoby:
Prover - osoba posiadająca wiedzę
Verifier - osoba pragnąca sprawdzić, czy Prover posiada tą wiedzę.
Własność zerowej wiedzy polega na tym, że Verifier po przeprowadzeniu rozmowy z Proverem nie dowiaduje się więcej o sekrecie niż to, co wiedział wcześniej lub mógł sam obliczyć.

## 2 Przykłady dowodów z zerową wiedzą

### 2.1 Izomorfizm grafów

#### 2.1.1 Parametry problemu

Graf $G$
Graf $H$

#### 2.1.2 Czego dowód dotyczy

Prover przekonuje Verifiera, że umie pokazać izomorfizm grafów $G$ i $H$

#### 2.1.3 Protokół

Prover: ustala (np. poprzez wylosowanie) graf $J$ i przekazuje go Verifierowi
Verifier: wybiera (np. poprzez losowanie) graf $G$ lub $H$ i prosi Provera o pokazanie izomorfizmu pomiędzy wybranym grafem a grafem $J$

### 2.1.4 Poprawność protokołu

Prover ma tylko 50 procent szans na oszustwo. Gdyż gdy Prover nie zna izomorfizmu grafów $G$ i $H$, to dla dowolnego grafu $J$, który sam wybierze, będzie znał co najwyżej izomorfizm z jednym z grafów $G$ lub $H$. Po $t$ próbach szansa na oszustwo spada do $2^{-t}$
Z drugiej strony Verifier w każdej rundzie poznaje izomorfizm jednego z grafów $G$ i $H$ z coraz to innym grafem, co poza wyjątkowymi sytuacjami (których prawdopodobieństwo wystąpienia jest podobnego rzędu, co prawdopodobieństwo samodzielnego znalezienia izomorfizmu poprzez losowanie funkcji izomorfizmu) nie daje nic Verifierowi.
Oczywiście w przypadku gdy liczba prób będzie większa od liczby permutacji grafu $G$ (czyli też $H$), to graf $J$, który wybierze Prover będzie musiał się w końcu powtórzyć. Wtedy Verifier juz znając z wcześniejszych prób izomorfizm $J$ z $G$ lub $H$, wybierze ten drugi i pozna w ten sposób izomorfizm $G$ z $H$. Ale przeprowadzenie tylu (silnia!) prób jest w praktyce niewykonalne, chyba że grafy są małe, ale wtedy Verifier może szybko znaleźć izomorfizm samemu, zatem własność zerowej wiedzy jest zachowana.

## 2.2 Nieizomorfizm grafów

### 2.2.1 Parametry problemu

Graf $G$
Graf $H$

### 2.2.2 Czego dowód dotyczy

Prover przekonuje Verifiera, iż wie, że grafy $G$ i $H$ nie sa izomorficzne.

### 2.2.3 Protokół

Verifier: wybiera (np. poprzez losowanie) graf $G$ lub $H$ i daje Proverowi jego izomorficzna kopię $J$
Prover: Mówi, czy otrzymany graf powstał z $G$, czy z $H$

### 2.2.4 Poprawność protokołu

Prawdopodobieństwo oszukania, czyli prawdopodobieństwo podania poprawnej odpowiedzi w przypadku gdy grafy są izomorficzne, wynosi $\frac{1}{2}$, gdyż w tym przypadku $J$ będzie izomorficzny z każdym z grafów $G$ i $H$ i Prover nie będzie miał zadnych możliwości rozróżnienia.

Z drugiej strony Verifier nie dostaje żadnych informacji co do tego, w jaki sposób Prover rozróżnia te grafy.

## 2.3 Kostka Rubika

### 2.3.1 Parametry problemu

Kostka Rubika $K$

### 2.3.2 Czego dowód dotyczy

Prover przekonuje Verifiera, że umie ułożyć tą kostkę

### 2.3.3 Protokół

Prover: bierze drugą kostkę i układa na niej jakąś pozycję $K'$(zapamiętując wykonane obroty)
Verifier: wybiera (np. poprzez losowanie) wariant żądania do Provera:
a) prosi o ułożenie z $K'$ kostki poprawnie ułożonej
b) prosi o ułożenie z $K'$ kostki $K$

### 2.3.4 Poprawność protokołu

Szansa oszukania w jednej rundzie wynosi $\frac{1}{2}$, gdyż Prover może znać dla podanej kostki $K'$ co najwyżej przekształcenie do jenej z kostek $K$ i kostki ułożonej, chyba że zna metodę ułożenia kostki $K$.
Z drugiej strony Verifier nie poznaje metody ułożenia kostki $K$, gdyż albo poznaje ułożenie zupełnie innej kostki, albo poznaje metode przekształcenia zupełnie innej kostki w kostę $K$. Nic mu to jednak nie da, gdyż w każdej rundzie (przy założeniu rozsądnej liczby tur) kostka podawana przez Provera jest zupełnie inna.

# 3 Protokół zerowej wiedzy dla identyfikacji

Wadą prostych protokołów identyfikacji za pomocą hasła jest fakt, iż Verifier (osoba sprawdzająca znajomość hasła) po otrzymaniu hasła Provera może się pod niego podszywać, gdyż poznaje całą wiedzę (w tym przypadku po prostu hasło) sprawdzaną przy identyfikacji. Można to zmienić wprowadzając protokół w którym Prover podaje tylko informację pośrednią, która przekona Verifiera, o posiadaniu przez Provera odpowiedniej wiedzy, lecz nie umożliwi

Verifierowi możliwosci podszywania się.
Jako przykład omówię dwa takie protokoły:

## 3.1 Protokoły identyfikacji (Feige-Fiat-Shamir)

Prover udowadnia swoją osobowość Verifierowi w $t$ wykonaniach następującego protokołu z jednorazowym początkowym przygotowaniem:

### 3.1.1 Wybranie parametrów systemu

Zaufane centrum $T$ publikuje $n = pq$ dla wszystkich użytkowników, po wybraniu sekretnych liczb pierwszych przystających do $3 \bmod 4$ ($n$ jest odpowiednio duże, aby nie można go bylo latwo rozłożyć na czynniki pierwsze ($p$ i $q$))
liczby $n$ i $t$ są parametrami bezpieczeństwa.

### 3.1.2 Wybranie danych prywatnych

wybranie $k$ liczb naturalnych $s_1, s_2, ..., s_k$ z przedziału $1 \leq s_i \leq n - 1$ i $k$ losowych bitów $b_1, ..., b_k$. (wymaga się, aby $gcd(s_i, n) = 1$, gdyż brak tego warunku spowoduje natychmiastową faktoryzację $n$ - $s_i$ będzie po prostu równe $p$ lub $q$)
Policzenie $v_i = (-1)^{b_i}(s_i^2)^{-1} \bmod n$ dla $1 \leq i \leq k$
Prover identyfikuje się bez pomocy kryptografii (dowód osobisty) z $T$, które rejestruje jego publiczny klucz $(v_1, ..., v_k; n)$, Prover zapamiętuje $(s_1, ..., s_k)$ i $n$ jako swój klucz prywatny.

### 3.1.3 Przepływ komunikatów

Prover daje Verifierowi: $x = (-1)^b r^2 \bmod n$
gdzie $b$ jest losowym bitem, $r$ losową liczbą naturalną, $1 \leq r \leq n - 1$
Verifier daje Proverowi: $(e_1, ..., e_k), e_i \in \{0, 1\}$
gdzie $e_i$ są losowymi bitami
Prover daje Verifierowi: $y = r \prod_{e_j=1}^{k} s_j \bmod n$
czyli iloczyn $r$ i tych $s_j$, których zażąda Verifier
W tym momencie Verifier sprawdza poprawność odpowiedzi:
Liczy $z = y^2 \prod_{j=1}^{k} v_j^{e_j} \bmod n$ i sprawdza, czy $z = \pm x$ i czy $z \neq 0$ (wyklucza to trywialną metodę oszukiwania poprzez wybranie $r = 0$.)

### 3.1.4 Bezpieczeństwo

Najlepszy atak typu "wybrany tekst" ma prawdopodobieństwo skutecznego podszycia $2^{-kt}$, zakładając, że $n$ jest trudne do rozłożenia na czynniki pierwsze.
Wymaga się przy tym także założenia, że obliczenie pierwiastka kwadratowego modulo duża liczba pierwsza o nieznanej faktoryzacji jest trudne.

## 3.2 Protokół identyfikacji (Shnorr)

Jest to protokół oparty na trudności problemu licznia logarytmu dyskretnego. Jest tak zaprojektowany, aby możliwe były wstępne obliczenia, po których wykonaniu identyfikacja wymaga tylko jednego obliczenia modulo liczba pierwsza. Jest to badzo ważne w przypadku, gdy Prover dysponuje ograniczonymi zasobami (np. mała i tania w produkcji karta chipowa)

### 3.2.1 Wybieranie parametrów systemowych

Liczba piewsza $p$ taka aby $p - 1$ było podzielne przez inną liczbe piewszą $q$. (aby liczenie logarytmu dyskretnego było trudne, należy wybrać odpowiednio duże liczby $p$ i (pośrednio) $q$, np: $p \approx 2^{1024}$ oraz $q \geq 2^{160}$.
Element $\beta$ taki ,że $1 \leq \beta \leq p - 1$ o rzędzie $q$ (np. gdy $\alpha$ jest generatorem, $\beta = \alpha^{\frac{p-1}{q}}$)
Każda ze stron otrzymuje z zaufanego źródła $T$ dane $(p, q, \beta)$ oraz funkcję weryfikacji umożliwiającą sprawdzanie podpisu $T$ (podpis funkcją $S_T$.)
Liczba powtórzeń $t$ gwarantująca bezpieczeństwo $2^{-t}$.

### 3.2.2 Prywatne parametry Provera

Identyfikator $I$
Prywatny klucz $a$ taki, że $0 \leq a \leq q - 1$
$v = \beta^{-a} \bmod p$
Prover porozumiewa się z $T$ i ustala swoją tożsamość za pomocą np. dowodu osobistego, daje $A$ i $v$, otrzymuje swój klucz prywatny $cert = (I, v, S_T(I, v))$ łączący niepodważalnie (zakładając zaufanie do $T$) $I$ z $v$.

### 3.2.3 Przepływ komunikatów

Prover wybiera losową liczbę $r$, taką, że $1 \leq r \leq q - 1$, oblicza $x = \beta^r \bmod p$ i wysyła $cert$ i $x$ do Verifiera

Verifier najpierw sprawdza autentyczność certyfikatu, sprawdzając, czy funkcja $S_T$ zastosowana do pierwszych dwóch części certyfikatu da jego trzecią część. Następnie wysyła do Provera losową liczbę $e$ (wcześniej nie używaną) taką, że $1 \leq e \leq 2^t$
Prover sprawdza, czy rzeczywiście $1 \leq e \leq 2^t$ i wysyła odpowiedź do Verifiera $y = ae + r \bmod q$.
Verifier liczy $z = \beta^y v^e \bmod p$ i uznaje odpowiedź Provera o ile $z = x$.

### 3.2.4 Bezpieczeństwo tego protokołu

**prawdopodobieństwo oszustwa** Szansa na zgadnięcie $e$ wynosi $2^{-t}$, Więc Prover może sprobować je zgadnąć i wtedy wybrać dowolne $y$ i wysłać Verifierowi $x = \beta^y v^e \bmod p$, oraz $y$ jako końcową odpowiedź. Zatem paramer $t$ musi być odpowiednio duży (np. 40), aby prawdopodobieństwo oszustwa bylo znikome.
Liczba $q$ (powstająca pośrednio z wyboru liczby $p$) powinna wynsić przynajmniej $2^{160}$, aby nie tylko krotkie obliczenia wkonywane podczas identyfikacji byly trudne bez znajomości prywatnego klucza, ale także aby szansa na złamanie poprzez dlugie obliczenia off-line byla mała.

**solidność** Można pokazać, że poprawne przejście protokołu (nie licząc zgadywania z małym prawdopodobieństwem sukcesu) wymaga umiejętności obliczenia $a$, czyli posiadania wiedzy, którą w tym protokole się sprawdza.
Z drugiej strony Prover nie przekazuje prawie żadnych użytecznych przyobliczaniu $a$ danych, gdyż $x$ jest liczbą losową, oraz $y$ bardzo silnie zależy od innej losowej, znanej tylko Proverowi, liczby $r$

**własność zerowej wiedzy** dla bardzo dużych $e$ (czyli $t$ też musi być nienaturalnie duże) własność zerowej wiedzy nie zachodzi, gdyż poprzez interakcję Verifier znajduje rozwiązanie $(x, y, e)$ równania $x = \beta^y v^e \bmod p$, czego samodzielnie mógłby nie potrafić. (nie widać jednak jak z tego wywnioskować $a$)