

## Wykład 4. Schematy Identyfikacji

Wykładowca: Stefan Dziembowski

Skryba: Paulina Kania, Łukasz Osipiuk

*Streszczenie.*

Wprowadzamy pojęcie systemu identyfikacji. Przedstawiamy klasyczne metody stosowane do identyfikacji użytkowników w systemach komputerowych. W dalszej części zajmujemy się nowoczesnymi technikami wykorzystującymi dowody wiedzy z wiedzą zerową.

[notatki częściowo oparte na [3] i Rozdziale 10 [1] oraz wykładzie [2]]

### 1 Wstęp

Dzisiaj zajmiemy się *schematami identyfikacji* (ang.: *identification schemes*) zwanymi też *schematami uwierzytelniania podmiotu* (ang.: *entity authentication*). Jest to przykład zastosowania dowodów z wiedzą zerową.

Wykład będzie miał charakter bardziej praktyczny niż poprzednie.

Ogólnie: w identyfikacji mamy do czynienia z dwiema stronami:

- *podmiotem uwierzytelnianym*, którego dla wygody będziemy zwykle nazywać *użytkownikiem*
- *podmiotem weryfikującym*, czyli *weryfikatorem*.

W skrócie: celem takiego schematu jest to, aby użytkownik udowodnił weryfikatorowi swoją tożsamość.

Pojęcia tego nie należy mylić z MACami, podpisami cyfrowymi oraz z autentykowanym uzgadnianiem klucza, choć te pojęcia są sobie bliskie.

Typowe zastosowania schematów identyfikacji obejmują: logowanie się do systemu komputerowego, do banku internetowego, pobieranie pieniędzy z bankomatu, itp.

Podmiotem uwierzytelnianym może być np. klient banku, a weryfikatorem bankomat.

## 2 Tradycyjne metody

Tradycyjne metody korzystają z haseł lub PINów. Polegają one na tym że każdy użytkownik systemu ma swoje hasło dostępowe.

1. Aby się uwierzytelnić użytkownik  $U$  wysyła swoje hasło do weryfikatora  $W$ .
2. weryfikator  $W$  sprawdza, czy hasło się zgadza i akceptuje albo nie.

Wady tego rozwiązania:

1. Każdy, kto podsłucha rozmowę między  $U$  a  $W$  poznaje hasło i może się podszyć pod  $U$ .

Można temu zaradzić korzystając z szyfrowanego połączenia.

2. Każdy kto ma dostęp do pliku z hasłami, może się podszyć pod  $U$ .

Można temu zaradzić w następujący sposób: zamiast przechowywać plik z hasłami zapisanymi tekstem jawnym, przechowujemy je „zaszyfrowane”.

Tak na prawdę, to nie są one zaszyfrowane, lecz po prostu przechowujemy wartości jakiejś funkcji jednokierunkowej  $f$  na tych hasłach.

To cały czas nie jest do końca bezpieczne, bo:

3. Każdy kto ma dostęp do pliku z „zaszyfrowanymi” hasłami może próbować je łamać *atakiem słownikowym*, polegającym na zastosowaniu  $f$  do wszystkich słów w danym słowniku  $\mathcal{S}$  i porównaniu wyników z wartościami w pliku z hasłami.  $\mathcal{S}$  może zawierać wszystkie słowa w popularnych językach, słowa znalezione w internecie i ich proste modyfikacje.

Aby zapobiec atakom słownikowym stosuje się następujące metody:

- zmusza się użytkowników do wyboru trudnych haseł
- wybiera się funkcję  $f$ , której obliczenie zajmuje trochę czasu.
- „soli się” hasła Do otrzymanego hasła najpierw dopisujemy losowe bity, następnie przykładamy funkcję jednokierunkową i do otrzymanego w ten sposób wyniku ponownie dopisujemy te same losowe bity. Dzięki temu, dla każdego użytkownika hash hasła jest inny, nawet jeśli dwóch użytkowników ma takie same hasła. Takie podejście uniemożliwia stosowanie ataku słownikowego dla wszystkich użytkowników w bazie jednocześnie. Dla każdego zaszyfrowanego hasła w bazie użytkowników i dla każdego słowa ze słownika  $\mathcal{S}$  trzeba osobno liczyć wartość funkcji  $f$ .

Jest to metoda stosowana w uniksie. Funkcją  $f$  jest lekko zmodyfikowana funkcja DES, tzn. (w uproszczeniu)  $f$  jest 25-krotnym złożeniem funkcji:

$$g(x) := E_x(0, \dots, 0).$$

Metoda ta jest bezpieczniejsza od niezmodyfikowanego DES, ponieważ uniemożliwia wykorzystywanie przy atakach sprzętowego dekodera DES.

4. Nawet jeśli zastosuje się metodę z poprzedniego punktu, to i tak otrzymane hasło będzie przez jakiś czas znajdowało się w pamięci operacyjnej weryfikatora. Więc nie jest w pełni bezpieczne.

### 3 Nowoczesne metody uwierzytelniania

Nowoczesne metody uwierzytelniania polegają na zasadzie *wyzwanie-odpowiedź* (ang.: *challenge-response*).

W skrócie działa to tak:

1. Weryfikator generuje jakąś wiadomość (*wyzwanie*) i wysyła do Użytkownika.
2. Użytkownik wysyła do weryfikatora *odpowiedź*.
3. Na podstawie otrzymanej odpowiedzi weryfikator akceptuje albo nie.

#### 3.1 Identyfikacja oparta na kryptografii klucza publicznego

Przykładem protokołu wyzwanie-odpowiedź jest następujący protokół oparty na kryptografii klucza publicznego (zakładamy, że Weryfikator posiada parę  $(e, d)$ ):

1. Weryfikator wysyła do użytkownika losową (w miarę długą) wiadomość  $M$ .
2. Użytkownik podpisuje ją swoim kluczem prywatnym i wysyła podpis do Weryfikatora.
3. Weryfikator akceptuje wtedy i tylko wtedy gdy podpis jest poprawny.

Jest to w miarę bezpieczne, ale ma następujące wady:

- Jeśli weryfikator jest nieuczciwy, to może wybrać  $M$  dowolne (a nie losowe). W związku z tym może uzyskać podpis Użytkownika na dowolnie wybranej wiadomości (inaczej mówiąc: protokół nie ma właściwości „wiedzy zerowej”). Zatem:
  - klucze  $(e, d)$  nie mogą być wykorzystane w innych celach.

- musimy założyć, że dany kryptosystem jest bezpieczny w bardzo silnym sensie (tzn. względem ataku z wybraną wiadomością).
- Jest dość mało wydajne (np. w przypadku zastosowania RSA wymaga podnoszenia do potęgi w  $Z_n^*$ )

### 3.2 Protokoły uwierzytelniania z wiedzą zerową

Pomysł jest taki: zamiast przesyłać hasło, użytkownik udowodni w zero-knowledge, że zna to hasło.

Np.: w przypadku rozwiązania z funkcjami jednokierunkowymi: zamiast przesyłać  $x$  (t.ż.  $f(x) = y$ , dla jakiegoś  $y$  znanego Weryfikatorowi) użytkownik udowodni (w zero-knowledge), że zna  $x$ .

Na pierwszy rzut oka wydaje się to łatwe, bo język:

$$\{f(x) : x \in \{0, 1\}^*\}$$

jest w NP, więc istnieje dla niego generyczny dowód z wiedzą zerową.

Ale uwaga: *to, że prover może udowodnić, że istnieje  $x$ , t.ż.  $f(x) = y$  nie musi oznaczać, że zna to  $x$ .* Potrzebujemy więc silniejszego pojęcia, czyli *dowodu wiedzy z wiedzą zerową* (ang.: *zero-knowledge proof of knowledge*).

#### 3.2.1 Wiedza maszyny

Spróbujmy zdefiniować, co to znaczy, że maszyna „coś wie”.

Nieformalnie oznacza to, że istnieje wydajna maszyna  $B$ , która używając innej maszyny  $A$  jako wyroczeni, wypisuje na wyjście to co „wie” maszyna  $A$ .

Niech  $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$  będzie relacją binarną. Jeśli  $(x, s) \in R$  to  $s$  nazwiemy *rozwiązaniem* dla  $x$ .

Powiemy, że  $R$  jest *wielomianowo ograniczona*, jeśli istnieje wielomian  $p$  taki, że  $|s| \leq p(|x|)$  dla każdej pary  $(x, s) \in R$ .

Powiemy, że  $R$  jest *NP-relacją*, jeśli  $R$  jest wielomianowo ograniczona i dodatkowo istnieje algorytm działający w czasie wielomianowym, sprawdzający czy dana para należy do relacji  $R$  ( $R$  jest *sprawdzalna w czasie wielomianowym*).

Oznaczmy przez  $P_{x,y,r}(\bar{m})$  wiadomość wysłaną przez maszynę  $P$ , przy wejściu wspólnym dla provera i weryfikatora  $x$ , przy wejściu dodatkowym dla provera  $y$ , przy wejściu losowym  $r$  i po odebraniu wiadomości  $\bar{m} = (m_1, \dots, m_n)$ .

Maszyna z wyrocznią, mająca dostęp do funkcji  $P_{x,y,r}$  będzie reprezentować wiedzę maszyny  $P$  z wejściem wspólnym  $x$ , wejściem dodatkowym  $y$  i wejściem losowym  $r$ .

Ta maszyna zwana *knowledge extractor* będzie starała się znaleźć rozwiązanie dla  $x$ .

Czas działania extractora jest odwrotnie proporcjonalny do prawdopodobieństwa z jakim przekonuje nas  $P$ .

### 3.2.2 System dowodu wiedzy

**Definicja 1 (System dowodu wiedzy)** Niech  $R$  będzie binarną relacją i niech  $\kappa : \mathcal{N} \rightarrow [0, 1]$ . Mówimy, że  $V$  jest weryfikatorem wiedzy dla relacji  $R$  z błędem  $\kappa$ , jeśli zachodzą następujące warunki:

- Istnieje interaktywna maszyna  $P$  taka, że dla dowolnej pary  $(x, y) \in R$ , po interakcji  $P$  z  $V$  (z wejściem wspólnym  $x$  i dodatkowym  $y$ ),  $V$  akceptuje.
- Istnieje wielomian  $q$  i maszyna  $K$  z wyrocznią taka, że dla dowolnej funkcji  $P$  i dla dowolnego  $x$ , mającego rozwiązanie, maszyna  $K$  spełnia następujący warunek:

Niech  $p(x)$  będzie prawdopodobieństwem, że maszyna  $V$  zaakceptuje, na wejściu wspólnym  $x$ , podczas interakcji z proverem specyfikowanych przez  $P_{x,y,r}$ .

Wtedy, jeśli  $p(x) > \kappa(|x|)$  to, na wejściu wspólnym  $x$ , po dostępie do wyroczni  $P_{x,y,r}$ , maszyna  $K$  wypisuje rozwiązanie  $s$  takie, że  $(x, s) \in R$ , w czasie oczekiwanym ograniczonym przez

$$\frac{q(|x|)}{p(x) - \kappa(x)}$$

Maszyna z wyrocznią  $K$  jest nazywana *universal knowledge extractor*.

**Twierdzenie 2** Niech  $R$  będzie NP relacją oraz niech  $q$  będzie wielomianem takim, że z tego, że  $(x, y) \in R$  wynika  $|y| \leq q(|x|)$ . Niech  $(P, V)$  będzie systemem dowodu wiedzy dla  $R$  z błędem  $\kappa(n) = 2^{-q(n)}$ . Wtedy  $(P, V)$  jest systemem dowodu wiedzy z błędem zero.

**Dowód:** Rozważmy dwa przypadki:

- Jeśli  $p(x) \geq 2 \cdot \kappa(x)$ , wtedy:

$$\frac{q(|x|)}{p(x) - \kappa(x)} \leq \frac{2 \cdot q(|x|)}{p(x)} = \frac{q'(|x|)}{p(x) - 0}$$

- Jeśli  $p(x) \leq 2 \cdot \kappa(x)$ , wtedy dzieląc obustronnie przez  $2 \cdot p(x) \cdot \kappa(x)$  uzyskujemy szacowanie  $\frac{1}{p(x)} \geq \frac{2^{q(x)}}{2}$ . Oczekiwany czas działania:

$$\frac{q(|x|)}{p(x) - \kappa(x)} \geq \frac{2 \cdot q(|x|)}{p(x)} \geq q(|x|) \cdot 2^{q(x)}$$

□

Wyjaśnimy teraz dlaczego protokół dla 3-kolorowania jest dowodem wiedzy (z błędem  $\kappa = 1 - \frac{1}{|E|}$ ).

Aby to pokazać wystarczy zbudować knowledge extractor, który wielokrotnie uruchamia funkcję  $P_{x,y,r}$  specyfikującą naszego provera „zapytując ją o kolejne wierzchołki grafu”. W ten sposób jesteśmy w stanie przekonać się że prover rzeczywiście znał 3-kolorowanie danego grafu.

### 3.2.3 Protokół Fiat-Shamira

Przykładem dowodu wiedzy z wiedzą zerową jest protokół Fiat-Shamira, który jest pewnym ulepszeniem dowodu dla reszt kwadratowych, który poznaliśmy na poprzednich ćwiczeniach.

Najpierw przypomnijmy algorytm z poprzedniego wykładu.

Aby dokonać identyfikacji prover będzie dowodził w zero knowledge, że zna pierwiastek liczby  $r = s^2 \pmod{N}$ . Kluczem prywatnym jest  $(s, N)$ , a publicznym  $(r, N)$ .

1. Prover wybiera losowe  $q \pmod{N}$  i wysyła  $t = q^2 \pmod{N}$  do weryfikatora.
2. Weryfikator wybiera losowe  $\sigma \in \{0, 1\}$  i wysyła je do provera.  $\sigma = 0$  oznacza, że weryfikator prosi o pierwiastek z liczby  $t \pmod{N}$ , w przeciwnym przypadku weryfikator prosi o pierwiastek z  $t \cdot r \pmod{N}$ .
3. Prover odpowiada wysyłając  $p = q \cdot s^\sigma \pmod{N}$ .
4. Weryfikator akceptuje, jeśli  $p^2 = t \cdot r^\sigma \pmod{N}$ .

Powtarzając ten protokół  $k$  razy uzyskujemy dowód wiedzy z błędem  $2^{-k}$ . Czyli jest to dowód wiedzy z wiedzą zerową.

Pierwszą poprawką jaką możemy wprowadzić jest zmniejszenie ilości przesyłanych komunikatów pomiędzy proverem i weryfikatorem (z  $2 \cdot k$  do 2). Teraz kluczem prywatnym jest  $(\bar{s}, N)$ , a publicznym  $(\bar{r}, N)$ , gdzie  $\bar{s} = [s_1, \dots, s_k]$ ,  $\bar{r} = [r_1, \dots, r_k]$  i  $r_i = s_i^2 \pmod{N}$ .

- Prover wybiera losowe  $q_i \pmod{N}$ , po czym wysyła do weryfikatora wektor  $\bar{t} = [t_1, \dots, t_k]$  taki, że  $t_i = q_i^2 \pmod{N}$ .
- Weryfikator wybiera losowe  $\bar{\sigma} = [\sigma_1, \dots, \sigma_k]$  i wysyła je do provera.
- Prover odpowiada wysyłając iloczyn  $p = q_i \cdot s_i^{\sigma_i} \pmod{N}$ .
- Weryfikator akceptuje, jeśli

$$\left(\prod_{i=1}^k p\right)^2 \equiv \prod_{i=1}^k t_i \cdot r_i^{\sigma_i} \pmod{N}$$

Drugim usprawnieniem, zaproponowanym przez Fiata i Shamira, jest wyliczenie wektora  $\bar{\sigma}$  przy użyciu jednokierunkowej funkcji hashującej  $H$ . Weryfikator zamiast losować kolejne bity, wysyła do provera  $H(\bar{t})$ .

Dalsze usprawnienie protokołu polega na wyeliminowaniu interakcji pomiędzy weryfikatorem a proverem.

- Prover wybiera losowe  $q_i \pmod{N}$ , po czym wysyła do weryfikatora wektor  $\bar{t} = [t_1, \dots, t_k]$  taki, że  $t_i = q_i^2 \pmod{N}$ .
- Prover oblicza wektor  $\bar{\sigma} = [\sigma_1, \dots, \sigma_k]$  taki, że  $H(\bar{t}) = \bar{\sigma}$ .
- Prover wysyła iloczyn  $p = q_i \cdot s_i^{\sigma_i} \pmod{N}$ .
- Weryfikator akceptuje, jeśli

$$\left(\prod_{i=1}^k p\right)^2 \equiv \prod_{i=1}^k t_i \cdot r_i^{\sigma_i} \pmod{N}$$

(weryfikator może wyliczyć wektor  $\bar{\sigma}$  na podstawie otrzymanego  $\bar{t}$ ).

### Poprawność protokołu Fiata-Shamira (intuicja)

Przeciwnik, bez znajomości  $\bar{s}$ , nie jest w stanie wyprodukować takich  $t_1, \dots, t_k$ , które przyłożone do funkcji hashującej  $H$  dadzą w wyniku wektor  $\sigma_1, \dots, \sigma_n k$ , ponieważ  $k$  jest zbyt duże, a funkcja  $H$  zachowuje się jak funkcja losowa.

Protokół Fiata-Shamira jest stosowany w praktyce, na przykład w kartach chipowych.

#### 3.2.4 Działanie przeciwnika „jak drut”

Należy wspomnieć jeszcze o sytuacji, w której „przeciwnik działa jak drut”. W tej sytuacji proverowi  $P$  wydaje się, że rozmawia z weryfikatorem  $V$ , podczas gdy na drodze  $P - V$  stoi jeszcze wrogi komputer  $E$  przekazujący dane w obie strony.

Może to grozić atakami następującej postaci:

1. Stawiamy fałszywy bankomat.
2. Łączymy go drogą radiową ze specjalnie spreparowaną kartą bankową, którą wkładamy do prawdziwego bankomatu.
3. Użytkownik wkłada swoją kartę do naszego podstawionego bankomatu. Gdy użytkownik zautoryzuje się, możemy wyjąć pieniądze z prawdziwego bankomatu.

## 4 Przykład adowe inne zastosowanie dowodu wiedzy z wiedzą zerową

Aby ulepszyć dowód nieizomorfizmu grafów  $G_1$  i  $G_2$ , tak aby był z wiedzą zerową, prover wysyła do weryfikatora nie tylko permutację któregoś z grafów, ale również udowadnia, że zna izomorfizm między wysłanym grafem a którymś z grafów  $G_1$  lub  $G_2$  (czyli że wysłana permutacja jest rzeczywiście permutacją  $G_1$  lub  $G_2$ ).

## Literatura

- [1] *Handbook of Applied Cryptography*. dostępne pod: <http://www.cacr.math.uwaterloo.ca/hac>.
- [2] Benny Chor. Lecture "introduction to modern cryptography": Identification schemes. zero knowledge proofs. Dostępne pod adresem <http://www.cs.tau.ac.il/~bchor/Lecture12.pdf>.
- [3] Oded Goldreich. Zero-knowledge proof systems. Dostępne pod adresem <http://www.wisdom.weizmann.ac.il/~oded/frag.html>.