

## Wykład 8. Protokoły głosowania elektronicznego

Wykładowca: Stefan Dziembowski

Skryba: Jarosław Bąbel, Rafał Linek

*Streszczenie.* Na dzisiejszym wykładzie skonstruujemy protokół głosowania elektronicznego. Najpierw wprowadzimy potrzebne narzędzia: pojęcie Tablicy Ogłoszeń, Szyfr ElGamala oraz jego progową wersję. Dalej opiszemy wstępną wersję protokołu. Na końcu uzupełnimy protokół tak, żeby działał poprawnie w środowisku z aktywnymi oszustami

## 1 Wstęp

Wykład powstał na podstawie [2, 3].

Na poprzednich wykładach poznawaliśmy generyczne metody tworzenia protokołów bezpiecznych obliczeń wielopodmiotowych.

Dzisiaj zajmiemy się konkretnym przypadkiem takiego protokołu. Będziemy konstruować protokół *głosowania elektronicznego*. Temat głosowania elektronicznego jest ostatnio coraz bardziej popularny, szczególnie w kontekście głosowania przy pomocy Internetu. Stąd prowadzone są prace nad efektywnymi protokołami do odbywania takich głosowań.

W protokole głosowania elektronicznego mamy dwa rodzaje uczestników. Są to:

- *wyborcy*  $V_1, \dots, V_m$  (ang.: *voters*) — głosują,
- *organizatorzy*  $A_1, \dots, A_n$  (ang.: *voting authorities*) — obliczają głosy. Kontaktują się z nimi wyborcy aby oddawać swoje głosy (o organizatorach możemy myśleć jako o serwerach obsługujących wybory).

Zakładamy, że liczba wyborców jest bardzo duża (np. może to być liczba mieszkańców danego kraju). Oczywiście organizatorów jest o wiele mniej.

Od protokołu do głosowania będziemy wymagać spełnienia następujących (nieformalnych) warunków:

1. Głosować mogą tylko uprawnieni wyborcy i każdy z nich co najwyżej jeden raz.
2. *uniwersalna weryfikowalność*: każdy (wyborca i organizator) może zweryfikować poprawność wyniku głosowania.

3. *prywatność*: głosy wyborców są tajne. Obowiązuje zasada *nepokwitowalności*: wyborca nie potrafi udowodnić jak głosował, nie może komuś oddać (sprzedać) swojego głosu. Nie chodzi tu o indywidualne przypadki, ale o zabezpieczenie przed przekupywaniem lub szantażowaniem większej ilości głosujących.
4. *odporność* (ang.: *robustness*): protokół ma działać nawet gdy jakaś koalicja uczestników działa niezgodnie z nim.
5. Obsługa protokołu po stronie wyborców powinna być prosta. Do głosowania mogą być używane urządzenia o małej mocy obliczeniowej np. telefony komórkowe.

Uwagi:

1. Ze względu na małą wydajność ogólne metody MPC (obliczeń wielopodmiotowych) nie mają tu zastosowania — liczba głosujących może być bardzo duża.
2. Jeśli jest tylko jeden organizator, to warunek 1 stoi w sprzeczności z prywatnością (nie zostanie spełniony warunek 3). Wówczas bowiem, wszystkie głosy spływają tylko do jednego organizatora. Może on więc, chcąc poznać głos konkretnego wyborcy, wymazać głosy oddane przez innych wyborców, obliczyć wynik głosowania i uzyskać potrzebną informację.

Przy tworzeniu protokołów głosowania będziemy zakładać, że wśród uczestników istnieje infrastruktura klucza publicznego. Mogą więc oni między sobą wymieniać zaszyfrowane wiadomości z możliwością weryfikacji nadawcy.

## 2 Narzędzia

### 2.1 Tablica Ogłoszeń

*Tablica ogłoszeń* (ang.: *bulletin board*) jest rodzajem kanału rozgłaszającego z pamięcią. Zakładamy, że wszyscy uczestnicy głosowania mają dostęp do tablicy ogłoszeń. Nieformalnie możemy założyć, że każdy uczestnik ma wyznaczone miejsce na tablicy (fragment pamięci), do której może pisać i wszyscy mogą to przeczytać (wszyscy widzą to samo).

### 2.2 Szyfr ElGamala

Kryptosystem ElGamala działa w podgrupie  $G_q$  rzędu  $q$  grupy  $Z_p^*$ , gdzie  $p$  i  $q$  są dużymi liczbami pierwszymi. Generatorem grupy  $G_q$  jest  $g$ . Bezpieczeństwo tego kryptosystemu oparte jest na trudności obliczania logarytmu dyskretnego w grupie  $G_q$ .

Parametry  $p$ ,  $q$  i  $g$  są publicznie znane.

Kluczem prywatnym jest losowe  $s \in \{0, \dots, q-1\}$ .

Kluczem publicznym jest  $h = g^s$ .

Aby zaszyfrować wiadomość  $m \in G_q$  wybieramy losowe  $\alpha \in \{0, \dots, q-1\}$ . Szyfrogramem  $E(\alpha, m)$  jest para  $(x, y) = (g^\alpha, h^\alpha m)$ . Parametr  $\alpha$  dodaje losowość, dzięki czemu szyfr staje się bardziej bezpieczny.

Aby odszyfrować szyfrogram  $(x, y)$  wystarczy obliczyć  $D((x, y)) = y/x^s$ .

Mamy:  $y = h^\alpha m$ ,  $x = g^\alpha$ ,  $h = g^s$ . Stąd  $y/x^s = h^\alpha m / (g^\alpha)^s = g^{\alpha s} * m * g^{-\alpha s}$ . Z tego wynika, że  $D((x, y)) = y/x^s = m$ .

Zauważmy, że szyfr ElGamala jest *homomorficzny*, w następującym sensie (mnożenie par wykonujemy „po współrzędnych”):

$$E(\alpha, m) \cdot E(\alpha', m') = E(\alpha + \alpha', m \cdot m').$$

### 2.3 Progowy szyfr ElGamala

Główną cechą kryptografii progowej jest wykorzystanie do szyfrowania protokołów wykonywanych wśród pewnej (dużej) liczby uczestników. *Szyfr progowy* ma następujące własności:

- zamiast *algorytmu* generacji klucza mamy *protokół* wykonywany w jakiejś grupie uczestników  $\{P_1, \dots, P_n\}$ . W wyniku działania tego protokołu obliczana jest para (klucz publiczny, klucz prywatny). Klucz prywatny nie jest nikomu znany, gdyż jest on podzielony pomiędzy uczestników protokołu (np. przy pomocy schematu Shamira).
- algorytm odszyfrowywania umożliwia odszyfrowanie kryptogramu bez konieczności ujawnienia wartości klucza prywatnego.

Teraz zajmiemy się progową wersją szyfru ElGamala. Wygląda ona następująco.

Najpierw gracze ustalają  $p$ ,  $q$  i  $g$ . Wartości te są obliczane na przykład z wykorzystaniem metod obliczeń wielopodmiotowych.

Następnie gracze generują parę kluczy: (publiczny, prywatny). Nie będziemy opisywać szczegółów tej procedury. Dla uproszczenia założmy, że zaufana strona trzecia  $T$  generuje tą parę  $(s, h = g^s)$  (patrz 2.2).

Następnie  $T$ :

1. Publikuje  $h$ .
2. Dzieli  $s$  pomiędzy graczy  $\{P_1, \dots, P_n\}$  za pomocą schematu Shamira z progiem  $t$ , gdzie  $t$  jest parametrem systemu. Niech  $s_j$  będzie udziałem gracza  $P_j$  w kluczu prywatnym.

Gracz nie może udowodnić, że coś otrzymał — brak ochrony przed aktywnymi oszustami. Rozwiązanie:  $T$  publikuje (na tablicy ogłoszeń) pary  $(j, h_j = g^{s_j})$ , gdzie  $1 \leq j \leq n$ . W ten sposób  $T$  nie ujawnia informacji o  $s_j$  (logarytm dyskretny jest trudny do policzenia) i chroni system przed aktywnymi oszustami.

Odszyfrowywanie szyfrogramu  $(x, y)$  wygląda następująco:

1. Każdy gracz  $P_j$  umieszcza  $w_j = x^{s_j}$  na tablicy ogłoszeń.  
Nieuczciwy gracz może podać wartość inną niż  $w_j$ . Problemem wykrywania aktywnych oszustw zajmiemy się później.
2. Załóżmy, że  $t$  graczy nie zostało złapanych na oszustwach w poprzednim kroku. Niech będą to gracze  $\{P_{i_1}, \dots, P_{i_t}\}$ .  
Sekret  $s$  jest podzielony schematem Shamira z progiem  $t$ . Stąd  $s$  można przedstawić jako kombinację liniową  $t$  udziałów  $s_{i_1} \dots s_{i_t}$  (patrz wykład 6., pkt. 10.) Istnieją więc, takie  $\lambda_1, \dots, \lambda_t$ , że:

$$s = \lambda_1 s_{i_1} + \dots + \lambda_t s_{i_t}.$$

Stąd:

$$x^s = x^{\lambda_1 s_{i_1} + \dots + \lambda_t s_{i_t}} = (x^{s_{i_1}})^{\lambda_1} \dots (x^{s_{i_t}})^{\lambda_t}$$

co jest równe  $w_{i_1}^{\lambda_1} \dots w_{i_t}^{\lambda_t}$ . Zatem każdy dysponujący dostępem do tablicy ogłoszeń może policzyć  $x^s$ . Może też więc policzyć  $y/x^s$  i odszyfrować szyfrogram.

### 3 Protokół (wstępna wersja)

W tym rozdziale dla uproszczenia założymy, że można głosować tylko „tak” lub „nie”, to znaczy, że każdy wyborca  $V_i$  ma jeden głos  $v_i \in \{-1, 1\}$ . Celem protokołu jest policzenie różnicy między głosami oddanymi na „tak” i na „nie”.

Głosowanie wygląda następująco:

1. Organizatorzy ustalają parę kluczy (publiczny, prywatny) w progowym szyfrze ElGamala (metodą jak wyżej). Niech  $h$  będzie kluczem publicznym (klucz prywatny dzielimy między organizatorów).
2. Każdy głosujący umieszcza na tablicy ogłoszeń swój zaszyfrowany głos:

$$(x_i, y_i) = (g^{\alpha_i}, h^{\alpha_i} g^{v_i})$$

Potem pokażemy w jaki sposób sprawdzić czy  $v_i \in \{-1, 1\}$ . Jeśli sprawdzenie dało wynik negatywny, to znaczy, że ktoś oddał głos nieważny.

Liczenie głosów przebiega następująco:

1. Każdy z uczestników może policzyć iloczyny:

$$(x, y) = (\prod x_i, \prod y_i),$$

gdzie mnożenie przebiega po wszystkich  $i = 1, \dots, m$ , takich że wyborca  $V_i$  oddał ważny głos. Ze względu na homomorfizm szyfru ElGamala szyfrogram  $(x, y)$  to zaszyfrowane  $g^d$ , gdzie  $d$ , to różnica głosów na „tak” i na „nie”.

2. Organizatorzy odszyfrowują  $(x, y)$  i zgodnie z protokołem umieszczają na tablicy wartości  $w_j = x^{s_j}$ .
3. Każdy kto ma dostęp do tablicy ogłoszeń może odszyfrować  $(x, y)$ , które jest szyfrogramem dla  $g^d$  (patrz 1). W celu uzyskania wyniku głosowania  $d$  wystarczy zlogarytmować  $g^d$ . Można to zrobić w czasie linowym ze względu na  $d$ .

Uwagi:

1. Wartość  $t$  jest możliwa do ustalenia.

Jeśli  $t$  jest bliskie  $n$ , to potrzebna jest duża koalicja oszustów żeby „zepsuć” głosowanie.

Jeśli jednak  $t = n$ , to odmowa współpracy lub „awaria” jednego z organizatorów powoduje niemożność przeprowadzenia głosowania.

W praktyce dobiera się  $t$  bliskie  $n$  i pozwalające na poprawne przeprowadzenie wyborów bez względu na „awarie” pewnej liczby organizatorów (odmowie współpracy można zapobiec nakładając duże kary na odmawiających).

2. Prywatność głosujących jest chroniona tylko obliczeniowo. Ma to tą wadę, że np. za 50 lat będzie można odczytać głosy wyborców. Istnieją protokoły chroniące prywatność bezwarunkowo.
3. Mała złożoność komunikacyjna protokołu pozwala na praktyczne zastosowanie:
  - Każdy głosujący wysyła jedną wiadomość wraz z dowodem, że zawiera ona ważny głos.
  - Każdy organizator musi przeczytać  $m$  (liczba ważnych głosów) wiadomości z tablicy ogłoszeń (wraz z dowodami poprawności) oraz umieścić tam jedną wiadomość.
  - Aby odczytać wynik trzeba odczytać  $t$  wiadomości (wraz z dowodami poprawności) z tablicy ogłoszeń.

## 4 Uzupelnienie

Pokażemy teraz jak organizatorzy i głosujący mogą udowodnić, że umieszczane przez nich na tablicy ogłoszeń wartości są zgodne z protokołem.

### 4.1 Dowód dla organizatora

W protokole deszyfrowania przedstawionym powyżej (faza obliczania głosów) każdy organizator musi udowodnić, że umieścił na tablicy prawdziwe  $w_j = x^{s_j}$ , gdzie  $s_j$  to jego udział w kluczu prywatnym  $s$ . Przypomnijmy, że  $h_j = g^{s_j}$  jest opublikowane na tablicy (patrz 2.3). Organizator musi udowodnić, że  $w_j$  i  $h_j$  dają ten sam logarytm o podstawach odpowiednio  $x$  i  $g$ .

Przedstawimy interaktywny dowód wiedzy o równości logarytmów  $y_1 = g_1^x$  i  $y_2 = g_2^x$  (bez ujawniania  $x$ ), gdzie  $x$  to losowy element należący do zbioru  $\{0, \dots, q-1\}$ .

Niech  $P$  to „Udowadniacz” (ang.: *prover*), a  $V$  to „Weryfikator” (ang.: *verifier*) (patrz wykład 3., pkt. 1.)

*Proof LogEq*( $g_1, y_1, g_2, y_2$ ):

1.  $P$  losuje  $r$  należące do  $\{0, \dots, q-1\}$ . Ustala  $a := (a_1, a_2) = (g_1^r, g_2^r)$  i wysyła  $a$  do  $V$ .
2.  $V$  losuje  $c$  należące do  $\{0, \dots, q-1\}$  i wysyła do  $P$ .
3.  $P$  oblicza  $b := r - c * x$  i wysyła  $b$  do  $V$ .
4.  $V$  akceptuje wtedy i tylko wtedy, gdy  $a_1 = g_1^b * y_1^c$  i  $a_2 = g_2^b * y_2^c$

**Pełność dowodu** : jeśli  $P$  zna wspólny logarytm  $y_1$  i  $y_2$  oraz  $P$  i  $V$  postępują zgodnie z protokołem, to  $V$  akceptuje.

**Poprawność dowodu** : załóżmy, że  $P$  oszukuje.  $P$  ma szansę  $\frac{1}{q}$  (prawdopodobieństwo zgadnięcia  $c$ ), że uda mu się oszukać tzn., że logarytmy są różne, a  $V$  zaakceptował. Chcemy pokazać, że to jest maksymalne prawdopodobieństwo z jakim  $P$  może oszukać.

Założmy, że logarytmy są różne i że  $P$  zna  $a = (a_1, a_2)$ , dla którego może odpowiedzieć na dwa różne wyzwania  $c$  i  $\tilde{c}$ . Stąd  $P$  może policzyć  $b$  i  $\tilde{b}$ , takie że:

$$a_1 = g_1^b * y_1^c, a_2 = g_2^b * y_2^c,$$

$$a_1 = g_1^{\tilde{b}} * y_1^{\tilde{c}}, a_2 = g_2^{\tilde{b}} * y_2^{\tilde{c}}.$$

Stąd można policzyć:

$$g_1^{\tilde{b}-b} = y_1^{c-\tilde{c}}, g_2^{\tilde{b}-b} = y_2^{c-\tilde{c}}$$

i dalej podnosząc równania stronami do potęgi  $c - \tilde{c}$ :

$$g_1^{\frac{\tilde{b}-b}{c-\tilde{c}}} = y_1, g_2^{\frac{\tilde{b}-b}{c-\tilde{c}}} = y_2.$$

Widać, że odpowiednie logarytmy  $y_1$  i  $y_2$  są sobie równe — sprzeczność z założeniem. Udowodniliśmy, że szansa oszukania przez  $P$  jest nie większa niż  $\frac{1}{q}$ .

Nie wiadomo, czy opisany dowód jest dowodem z wiedzą zerową (nie potrafimy pokazać odpowiedniego symulatora). Na pewno jest z wiedzą zerową jeżeli weryfikator  $V$  jest uczciwy (ang.: *honest-verifier*). Inaczej mówiąc dowód jest z wiedzą zerową jeżeli  $V$  dobiera  $c$  w sposób zupełnie losowy.

Pokażemy symulator, który potrafi (nieformalnie mówiąc) uzyskać to samo co weryfikator, który ma dostęp do  $P$  (stąd wniosek, że  $V$  nie wynosi żadnej wiedzy z komunikacji z  $P$ ).

Zakładamy, że wartości  $(g_1, y_1, g_2, y_2)$  są poprawne. Symulator produkuje transkrypt wykonania protokołu bez dostępu do  $P$ . Algorytm symulatora:

$S(g_1, g_2, y_1, y_2)$

1. wylosuj  $\tilde{b}$  należące do  $\{0, \dots, q-1\}$
2. wylosuj  $\tilde{c}$  należące do  $\{0, \dots, q-1\}$  (to jest zadanie weryfikatora)
3.  $\tilde{a}_1 := g_1^{\tilde{b}} * y_1^{\tilde{c}}, a_2 := g_2^{\tilde{b}} * y_2^{\tilde{c}}$
4. Zwróć  $(\tilde{a}_1, \tilde{a}_2, \tilde{c}, \tilde{b})$

Łatwo pokazać, że rozkład prawdopodobieństwa dla uzyskiwanych transkryptów jest taki sam jak rozkład dla uczciwego weryfikatora.

## 4.2 Dowód dla głosującego

Każdy głosujący musi udowodnić, że zaszyfrował głos  $g^v$  należący do zbioru  $\{g, g^{-1}\}$ . Inaczej mówiąc musi dowieść, że  $c = (c_1, c_2) = (g^\alpha, h^\alpha * m)$  i  $m$  należy do zbioru  $\{g, g^{-1}\}$ . W tym celu dowodzi, że zna  $\alpha$  dla  $c_1 = g^\alpha$  i  $c_2 * g^{-1} = h^\alpha$ , lub  $c_1 = g^\alpha$  i  $c_2 * g = h^\alpha$ . Każdego składnika alternatywy można dowieść tak jak w dowodzie dla organizatora. Tutaj jednak zadanie  $P$  jest trudniejsze: dowód nie może ujawniać, która część alternatywy jest prawdziwa.

## 4.3 Nieinteraktywne dowody wiedzy

Każdy uczestnik protokołu, który chce pokazać, że zachował się „dobrze” (zgodnie z protokołem) musi przedstawić dowód  $(c, b)$ . Kto spełnia rolę weryfikatora w naszym

protokole? Kto generuje  $c$ ? Mogą to robić wszyscy wspólnie, ale to jest niepraktyczne ze względu na dużą złożoność. W takim razie chcemy, żeby  $c$  zostało wygenerowane nieinteraktywnie (przy pomocy pewnej heurystyki). Nieformalnie mówiąc: pokażemy nieinteraktywny dowód dla dowodu interaktywnego.

Potrzebujemy funkcji haszującej „dobrej” w sensie kryptograficznym tzn. takiej, że trudno znaleźć dwie różne wartości „haszujące się” do jednej. Tą własność można nazwać odpornością na kolizje (ang.: collision-resistance).

Za pomocą funkcji  $h$  symulujemy niezależność  $V$ . Funkcję  $h$  nazywa się czasem po angielsku *random-oracle*.

Zmiana w dowodzie polega na tym, że  $P$  po wylosowaniu  $r$  i ustaleniu  $a$  oblicza  $c$ :

$$c := h(g_1, y_1, g_2, y_2, a_1, a_2)$$

i ustala  $b := r - c * x$ .

Sprawdzenie, że para  $(c, b)$  jest poprawnym dowodem polega na sprawdzeniu równości:

$$c = h(g_1, y_1, g_2, y_2, g_1^b * y_1^c, g_2^b * y_2^c).$$

Weryfikator sprawdzający poprawność dowodu nie musi znać  $a$ , żeby sprawdzić warunek. Ze względu na „dobre” własności funkcji  $h$ , na podstawie  $h(u) = h(v)$  możemy wywnioskować, że  $u = v$ .

Jeżeli zamienimy nasze interaktywne dowody wiedzy w nieinteraktywne, to dowód z wiedzą zerową przy założeniu uczciwości weryfikatora jest wystarczający, ponieważ nasz weryfikator jest zawsze uczciwy ( $c$  generuje funkcja  $h$ , a sprawdzenia dokonuje każdy zainteresowany).

W protokole elektronicznych wyborów każdy organizator i każdy głosujący uzupełniają swoje wiadomości nieinteraktywnymi dowodami, co pozwala przekonać innych uczestników, że działali zgodnie z protokołem.

## Literatura

- [1] J. Cohen. Improving privacy in cryptographic elections, 1986.
- [2] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997.
- [3] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. Springer Verlag, 2002.

- [4] Claus Peter Schnorr and Markus Jakobsson. Security of signed ElGamal encryption. *Lecture Notes in Computer Science*, 1976:73–??, 2000.
- [5] Yiannis Tsiounis and Moti Yung. On the security of ElGamal-based encryption. *Lecture Notes in Computer Science*, 1431:117–??, 1998.