Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Dominik Kamiński
Nr albumu: 189183

Adaptywność w wielopodmiotowych obliczeniach funkcji

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dra Stefana Dziembowskiego
Instytut Informatyki

Maj 2004

Pracę przedkładam do oceny

Data Podpis autora pracy:

Praca jest gotowa do oceny przez recenzenta

Data Podpis kierującego pracą:

Streszczenie

Inspiracją dla tej pracy jest definicja bezpieczeństwa wielopodmiotowego obliczenia funkcyjnego podana przez Canettiego oraz osiągnięcia Canettiego, Damgaarda, Dziembowskiego, Ishaia i Malkina w określeniu przewag przeciwnika adaptywnego nad nieadaptywnym w świetle tej definicji w różnych modelach obliczeń.

Wielopodmiotowe obliczenie funkcji jest wykonywane przez pewien z góry określony zbiór graczy, którzy przy pomocy wzajemnej komunikacji obliczają pewną funkję ich wejść. Dla każdego takiego obliczenia można rozważać przeciwnika, który - przez gromadzenie informacji i korumpowanie graczy - stara się je zakłócić lub zgromadzić pewne niebezpieczne informacje. Oprócz rozważania innych modeli, można rozróżniać dwa typy przeciwników. Takich, którzy na początku obliczenia korumpują z góry określony zbiór graczy nazywamy nieadaptywnymi, tych zaś którzy korumpują graczy w trakcie trwania obliczenia - na podstawie już zdobytych informacji - adaptywnymi.

Cytuję definicję zawartą w pracy Canettiego ilustrując ją przykładami. Przedstawiam własne rozważania na temat pozostawionego bez odpowiedzi w pracy [2] pytania czy istnieje protokól obliczenia funkcji dla dwóch graczy w modelu otwartych kanałów, który jest bezpieczny nieadaptywnie, ale niebezpieczny adaptywnie, czyli inaczej mówiąc, czy w tym przypadku bezpieczeństwo protokołu w sensie nieadaptywnym gwarantuje bezpieczeństwo w sensie adaptywnym.

Słowa kluczowe

obliczenia wielopodmiotowe, protokół kryptograficzny, bezpieczeństwo protokołu, adaptywność

Klasyfikacja tematyczna

94A60

Spis treści

1. Wprowadzenie 5

2. Wielopodmiotowe obliczenie funkcji 7
2.1. Wstęp 7
2.2. Definicja 9
2.3. Modele 10

3. Bezpieczeństwo wielopodmiotowego obliczenia funkcji 11
3.1. Pojęcie bezpieczeństwa 11
3.2. Bezpieczeństwo nieadaptywne 12
3.2.1. Proces rzeczywisty 12
3.2.2. Proces idealny 13
3.2.3. Definicja bezpieczeństwa 14
3.3. Bezpieczeństwo adaptywne 15
3.3.1. Proces rzeczywisty 15
3.3.2. Proces idealny 17
3.3.3. Definicja bezpieczeństwa 18

4. Wpływ adaptywności na bezpieczeństwo obliczenia 19
4.1. Wprowadzenie 19
4.1.1. Początkowa adaptywność 20
4.2. Rozróżnienie dla aktywnego przeciwnika i więcej niż dwóch graczy 22
4.3. Obliczenia dwustronne 24
4.3.1. Tajne kanały komunikacyjne 24
4.3.2. Otwarte kanały komunikacyjne 26

5. Podsumowanie 35

A. Zespoły rozkładów prawdopodobieństwa 37

Rozdział 1

Wprowadzenie

W pracy tej pragnę przedstawić problem wielopodmiotowych obliczeń funkcyjnych w kontekście adaptywności przeciwnika. Adaptywność, rozumiana jako zdolność przeciwnika do korumpowania uczestniczących w obliczeniu graczy, stanowi ważne rozróżnienie. Dowodzenie bezpieczeństwa protokołu wobec przeciwników nieadaptywnych są zazwyczaj dużo łatwiejsze. Dlatego twierdzenia ogólne mówiące o równoważności bezpieczeństwa adaptywnego i nieadaptywnego protokołów w różnych modelach są bardzo pożyteczne. Pomijam tu zagadnienie łączenia wielu obliczeń w jedno obliczenie złożone, co pozwala na praktyczne zastosowanie definicji do konstrukcji dowodów bezpieczeństwa protokołów złożonych[1].

Definicja bezpieczeństwa Canettiego wykorzystująca podejście Goldreicha do dowodów z wiedzą zerową polega na porównaniu obliczenia rzeczywistego z innym obliczeniem, zwanym idealnym, na które nakłada się odpowiednie warunki dodatkowe. Od obliczenia idealnego żąda się, by przeciwnik (zwany symulatorem) nie miał dostępu do rzeczywistej komunikacji między graczami, która jest ukryta w mechanizmie dodatkowego niekorumpowalnego gracza. Protokół jest bezpieczny, jeśli ogólny wynik obliczenia graczy i przeciwnika rzeczywistego jako rozkład funkcji losowej, można uzyskać podczas obliczenia idealnego. Porównywanie takich dwóch obliczeń stanowi dobrą formalizację intuicji bezpieczeństwa.

Bezpieczeństwo obliczenia wielopodmiotowego rozpatruje się w zależności od wielu różnych czynników. Oprócz adaptywności przeciwnika rozważa się oczywiście jego aktywność i moc obliczeniową. Ważną rolę odgrywa rozróżnienie przypadków tajności i otwartości kanałów komunikacyjnych między graczami (domyślnie nie zakłada się istnienia wspólnego kanału rozgłoszeniowego, ale wiele prywatnych kanałów dla par graczy) oraz mechanizm wymazywania informacji przez graczy, gdy te zostały już wykorzystane w obliczeniu. Wobec definicji postawionej przez Canettiego ([1]) rodzą się jeszcze inne rozróżnienia, a mianowicie stosunek mocy przeciwnika realnego i symulującego go przeciwnika idealnego oraz podobieństwo rozkładów obliczenia rzeczywistego i idealnego.

Za pracą Canettiego, Damgaarda, Dziembowskiego, Ishaia i Malkina ([2]) rozpatrzę kilka możliwych modeli obliczeń, popierając je przykładami i dowodami. Na koniec zajmę się pozostawionym przypadkiem dla dwóch graczy przy otwartych kanałach komunikacyjnych. Przedstawię kilka faktów i twierdzeń z dowodami. Rozstrzygnięcie tego przypadku wydaje się trudne i pozostawiam je jako otwarte.

W następnym rozdziale, poczynając od przykładów, opiszę, co rozumiem pod pojęciem wielopodmiotowego obliczenia funkcji i przedstawię różnorodność modeli ze względu na przeciwnika. W rozdziale trzecim przedstawię nieco obszerniej definicję bezpieczeństwa dla przypadku przeciwnika nieadaptywnego wraz z potrzebnymi pojęciami, po czym przejdę do bez-

[1] Zagadnienie jest opisane w [1].

pieczeństwa adaptywnego. W kolejnych podrozdziałach rozdziału czwartego podam kilka faktów na temat bezpieczeństwa i omówię ciekawsze ze względu na postawione pytanie przypadki obliczeń porównujące przeciwnika adaptywnego z nieadaptywnym, przy czym ostatnia część będzie poświęcona przypadkowi dwóch graczy przy otwartych kanałach komunikacyjnych. Podrozdział 4.1.1, motywowany przez [1], jest dyskusją nad początkową adaptywnością przeciwnika. W ostatnim rozdziale zamieszczam podsumowanie tej pracy. Dla przejrzystości, dodatek A zawiera wszelkie konieczne definicje i twierdzenia, których znajomość jest potrzebna dla zrozumienia tej pracy.

Rozdział 2

Wielopodmiotowe obliczenie funkcji

2.1. Wstęp

Przed formalnym postawieniem problemu, rozważmy prosty przykład pokazujący istotę obliczeń wielopodmiotowych i prostą intuicję bezpieczeństwa.

Przykład 1 *Dwie osoby - Alicja i Bob - chcą sprawdzić, czy wzajemnie się kochają. Jednocześnie każda z nich chce jak najmniej ujawnić na temat swojego uczucia.*

Niech:

$$a = \begin{cases} 1 & \textit{jeśli Alicja kocha Boba} \\ 0 & \textit{w przeciwnym przypadku} \end{cases} \quad b = \begin{cases} 1 & \textit{jeśli Bob kocha Alicję} \\ 0 & \textit{w przeciwnym przypadku} \end{cases}$$

Alicja i Bob obliczają więc dwuargumentową funkcję ich wejść:

$$f(a,b) = a \wedge b.$$

Jeśli okaże się, że $f(a,b) = 1$, to oczywiście każde będzie wiedzieć o uczuciu drugiego. Podobnie, jeśli Alicja kocha Boba i okaże się, że nie kochają się nawzajem, to będzie ona wiedzieć, że Bob jej nie kocha.

Jeśli jednak Alicja nie kocha Boba ($a = 0$), ponieważ $f(a,0) = 0 = f(a,1)$, to jeśli Alicja poznałaby tylko wartość f, nie wiedziałaby o uczuciu Boba i takiego zachowania oczekuję się od bezpiecznego obliczenia podanej funkcji.

W przykładzie tym mamy do czynienia z pewną funkcją dwuargumentową. Nie widać jednak jeszcze sposobu obliczenia tej funkcji. Oczywiście różnych obliczeń może być wiele. Najbardziej prymitywnym jest następujące:

1. Alicja wysyła bit a do Boba.
2. Bob wysyła wartość $f(a,b)$ do Alicji.
3. Każde zwraca wartość $f(a,b)$.

Tutaj Bob „za darmo" - niezależnie od własnego uczucia - poznaje uczucie Alicji. Obliczenie takie jest w naszej intuicji niebezpieczne. Po pierwsze nie jest spełniony postulat podany w przykładzie. Co prawda jeśli Alicja Boba nie kocha, to nie pozna jego uczucia, ale Bob zawsze poznaje uczucie Alicji (obliczenie jest niesymetryczne). Poza tym Bob może tu łatwo oszukać Alicję, wysyłając jej nieprawdziwą wartość. Jeśli jednak dopuścilibyśmy do obliczenia Ewę jako trzecią zaufaną stronę:

1. Alicja wysyła bit a do Ewy.
2. Bob wysyła bit b do Ewy.
3. Ewa odsyła każdemu wartość $f(a, b)$.
4. Alicja i Bob zwracają $f(a, b)$.
5. Ewa nic nie zwraca.

Teraz, przy tym rygorystycznym założeniu, że Ewa nie oszukuje, obliczenie nabiera żądanych cech. Jeśli jednak Ewa chciałaby oszukać, to może ona dowolnie pokierować obliczeniem.

Dla opisania „złych" cech obliczenia, wprowadza się do protokołu przeciwnika. Odpowiada on za próby oszukiwania podjęte przez uczestników oraz za wyciek informacji. Przeciwnikowi pozwala się na korumpowanie graczy. I tak w pierwszym z powyższych obliczeń, przeciwnik może doprowadzić do sytuacji nieprawidłowej, jeśli skorumpuje Boba. W drugim przypadku, przeciwnik korumpujący Alicję lub Boba, nie dowiaduje się nic ponad ich uczucia, dlatego zakłada się, że Ewa jest graczem zaufanym. Rodzi się więc naturalne rozróżnienie na graczy korumpowalnych i zaufanych i mówi się o strukturze $\mathcal{B}$ przeciwnika, jako rodzinie podzbiorów zbioru wszystkich graczy. Struktura przeciwnika musi mieć taką własność, że wszystkie podzbiory zbioru do niej należącego muszą także do niej należeć. Obrazuje to założenie, że jeśli pewien zbiór graczy jest korumpowalny, to każdy z tych graczy także. Na przykład o drugim obliczeniu można powiedzieć, że „wygląda" na bezpieczne wobec przeciwników ograniczonych strukturą $\mathcal{B} = 2^{\{A,B\}}$, ale nie dla przeciwników takich, że $\{E\} \in \mathcal{B}$.

Na przykładzie tej prostej funkcji widać też inne aspekty obliczeń. Jednym z nich jest możliwość zmiany komunikacji przez przeciwnika. Gdyby na przykład, w pierwszym obliczeniu, Alicja została skorumpowana, to przeciwnik, który „kazałby" jej zawsze mówić, że kocha Boba, łatwo dowiadywałby się o uczuciu tego drugiego. Jednocześnie, gdy taki przeciwnik może tylko podsłuchiwać, to skorumpowanie Alicji pozwala daje mu tylko znajomość jej uczucia do Boba.

Kolejnym rozróżnieniem może być to, czy przeciwnik ma dostęp do całej komunikacji między graczami, czy tylko do części dotyczącej graczy skorumpowanych. Jeżeli w drugim przypadku przeciwnik może podsłuchać wszystko, to bez korumpowania kogokolwiek, dowiadywałby się o uczuciach Alicji i Boba. Wobec takich przeciwników można stosować szyfrowanie wiadomości.

Na podstawie następnego przykładu pokażę, jak na obliczenie może wpływać dynamiczne korumpowanie graczy przez przeciwnika.

Przykład 2 *([3])*
Dzielenie sekretu w ogólności polega na rodzieleniu pewnej informacji (sekretu) pomiędzy m graczy, tak by każdych $k < m$ graczy umiało tę informację odtworzyć, a jednocześnie nie umiało tego dokonać żadnych $k-1$ graczy. Można tego dokonać przez wykorzystanie interpolacji wielomianu.

Rozważmy dzielenie sekretu przez gracza P_1 na zbiorze $\{P_1, \ldots P_n\}$. Funkcją jaką pragną obliczyć gracze, jest funkcja pusta. Niech przeciwnik może skorumpować co najwyżej $t = \mathcal{O}(n)$ graczy i gracz P_1 jest niekorumpowalny (struktura przeciwnika $\mathcal{B} = \{B \in 2^{\{P_2,\ldots P_n\}} : |B| \leq t\}$).

1. *P_1 dostaje na wejściu pewien sekret secret.*
2. *P_1 wybiera losowo dowolny zbiór S o liczebności $\omega(\log n) < m = |S| < t$ (np. $m = [\sqrt{t}]$) graczy, którym swój sekret powierzy.*

3. *P_1 publikuje S i dzieli secret pomiędzy graczy z S tak, że dopiero udziały wszystkich m graczy pozwalają go odtworzyć.*

4. *Wyjście każdego z graczy jest puste.*

Popatrzmy najpierw na sytuację, gdy przeciwnik nie ma możliwości dynamicznego korumpowania graczy. Odpowiada to założeniu, że zbiór graczy oszukujących lub kontatkujących się ze sobą, by powierzony sekret odtworzyć jest zdeterminowany na początku. Przeciwnik taki ma niewielką szansę odtworzenia sekretu, bo to, między kogo będzie rozdzielony sekret, gracz P_1 ustala w czasie trwania obliczenia. Aby więc przeciwnik mógł poznać sekret, gracz P_1 musiałby wylosować dokładnie zbiór graczy skorumpowanych. Szansa takiego zdarzenia jest odwrotnie wykładnicza ze względu na m.

Wyobraźmy sobie teraz przeciwnika, który w czasie obliczenia może korumpować graczy, w zależności od zebranych informacji. Odpowiada to sytuacji, gdy gracze mogą zacząć oszukiwać dopiero w trakcie obliczenia, a na początku wszyscy są prawi. Taki przeciwnik może zawsze odkryć podzielony sekret. Wystarczy, że skorumpuje jakiegokolwiek gracza i poczeka na rozgłoszenie zbioru S powierników sekretu. Teraz już z łatwością, korumpując wszystkich powierników, sekret odtwarza.

Można więc powiedzieć, że przedstawione obliczenie nie jest bezpieczne wobec przeciwnika dynamicznie korumpującego graczy, a jednocześnie zachowuje pozory bezpieczeństwa, jeśli zbiór graczy oszukujących jest zdeterminowany na samym początku[1].

2.2. Definicja

Przez wielopodmiotowe obliczenie π funkcji (często zwane dalej po prostu protokołem π) rozumiem obliczenie pewnej z góry zadanej funkcji $f : D^n \times \{0,1\}^* \to D^n$, gdzie D to ustalony wcześniej język nad alfabetem Σ, zaś n to moc zbioru graczy $\mathbf{G} = \{G_1, \ldots G_n\}$ obliczających tę funkcję[2]. Ponieważ przypadek dowolnego alfabetu skończonego można sprowadzić do liczb binarnych, niech dalej $\Sigma = \{0,1\}$.

Na początku obliczenia każdy gracz G_i, modelowany przez probablistyczną maszynę Turinga dostaje na wejściu pewne słowo in_i, element losowości $rand_i$ oraz współczynnik złożoności k. Następnie dochodzi do komunikacji między graczami, którzy mogą przesłać do siebie wielomianową względem k liczbę komunikatów, będących słowami nad Σ o wielomianowej długości względem parametru k. Zakładam, że każda para graczy jest połączona oddzielnym kanałem komunikacyjnym i cała wymiana informacji jest w pełni autentykowana. Po zakończeniu fazy komunikacji każdy gracz i oddaje na wyjściu słowo $out_i = f(k, \overrightarrow{in}, \overrightarrow{rand})_i$. W przypadku obliczania funkcji interesuje nas więc całościowy wynik $f(k, \overrightarrow{in}, \overrightarrow{rand})$.

Do takiego obliczenia dochodzi dodatkowa probablistyczna maszyna Turinga, zwana przeciwnikiem, która dla rzeczywistych protokołów kryptograficznych może modelować robaka internetowego lub intruza. Celem przeciwnika jest, przez gromadzenie informacji i korumpowanie graczy (przejmowanie kontroli nad graczami), zebranie jakiś konkretnych informacji lub doprowadzenie do niepoprawnego obliczenia funkcji. Aby zaznaczyć wpływ przeciwnika na działanie protokołu, rozszerza się funkcję i wynikiem obliczenia staje się $f(k, \overrightarrow{in}, aux, \overrightarrow{rand})$,

[1] Według definicji przedstawionej w tej pracy, protokół ten nie jest bezpieczny nawet w takim przypadku.

[2] Nie wszystkie protokoły kryptograficzne są obliczeniami funkcji. Istnieją protokoły, które nie obliczają żadnej z góry zadanej funkcji, jak również takie, od których, mimo, że obliczają funkcję, żąda się nie indywidualnego bezpieczeństwa, ale bezpieczeństwa zespołu protokołów (np. wielokrotne kodowanie przy pomocy tego samego klucza prywatnego i publicznego, które ma mieć własność tajności klucza prywatnego). Większość tych zagadnień można jednak sprowadzić do obliczania funkcji.

gdzie aux jest pewnym wejściem dodatkowym przeciwnika modelującym wiedzę z poprzednich wykonań protokołu lub jakąś dodatkową informację, zaś $rand_0$ jest losowym wejściem przeciwnika.

2.3. Modele

Można rozpatrywać dwa typy przeciwników: pasywnych i aktywnych. Przeciwnik pasywny zbiera informacje i korumpuje graczy, by dowiedzieć się jakiegoś sekretu. Zaś przeciwnik aktywny przez zmianę wejść graczy i zaburzenie komunikacji pragnie doprowadzić do nieprawidłowego obliczenia funkcji lub w skrajnym przypadku do jego zatrzymania w pewnych niekorzystnych warunkach. Korumpowanie graczy polega na przejęciu całkowitej kontroli nad ich maszynami w przypadku przeciwnika aktywnego, zaś dla pasywnego jedynie wejście w posiadanie pełnej informacji o ich stanie (w tym komunikacji z innymi graczami).

Dalej można rozróżniać przypadek kanałów tajnych, gdzie komunikaty są ściśle tajne i przeciwnik może je poznać tylko przez korumpowanie graczy, do lub od których zostały one wysłane oraz przypadek kanałów otwartych, gdy cała komunikacja jest przez przeciwnika widziana. Można też zakładać istnienie kanału rozgłoszeniowego, dzięki któremu wysłany przez gracza komunikat zostaje dostarczony do wszystkich pozostałych graczy.

Następne kryterium to trwałość informacji określająca, czy część komunikacji dotycząca danego gracza jest przez niego pamiętana. Jest to istotne podczas korumpowania gracza przez przeciwnika w przypadku tajnych kanałów. W rzeczywistych obliczeniach bowiem, oprócz mechanizmu „czyszczenia pamięci" gracza, istnieje pewne ryzyko utraty zapamiętywanych, już niepotrzebnych danych.

Najważniejszą dla tej pracy jest kwestia adaptywności przeciwnika. Niech $\mathcal{B} \subseteq 2^{\mathbf{G}}$ będzie niepustą strukturą monotoniczną nad zbiorem graczy, czyli taką, że $\forall_{B \in \mathcal{B}} B' \subset B \Rightarrow B' \in \mathcal{B}$. Przeciwnik nieadaptywny jest zdeterminowany do skorumpowania na samym początku obliczenia ustalonego na początku zbioru graczy $B \in \mathcal{B}$, podczas gdy przeciwnik adaptywny może korumpować graczy kolejno, zgodnie z pewną strategią, w miarę obliczenia, byle tylko po każdym skorumpowaniu podzbiór graczy skorumpowanych należał do $\mathcal{B}$. Oczywiście intuicyjnie przeciwnik adaptywny jest mocniejszy. Większość przypadków pod tym względem rozstrzyga praca [2].

Rozdział 3

Bezpieczeństwo wielopodmiotowego obliczenia funkcji

3.1. Pojęcie bezpieczeństwa

Definicja bezpieczeństwa cytowana za Canettim [1] adaptuje do obliczeń wielopodmiotowych funkcji podejście z pracy Goldreicha na temat dowodów z wiedzą zerową [4]. Polega ona na porównaniu danego rzeczywistego obliczenia funkcji z innym obliczeniem tej samej funkcji, na które nakłada się dodatkowe założenia. Przez takie podejście można przez jawne wyodrębnienie odpowiednich cech precyzyjnie sformalizować intuicję bezpieczeństwa.

Od bezpiecznego protokołu żąda się, by był sekretny i zapewniał poprawne obliczenie. Sekretność ma się przejawiać w tym, iż przeciwnik nie będzie mógł wydobyć z protokołu żadnych istotnych informacji poza wejściami i wyjściami graczy skorumpowanych. Oczywiście nie można ochronić się przed poznaniem przez przeciwnika wejść graczy, których ten skorumpuje, ale żąda się, by między innymi wejścia i - w jakimś sensie - wyniki pozostałych graczy pozostały tajne. Poprawność oznacza niemożliwość zakłócenia obliczeń przez przeciwnika, by wyniki graczy nieskorumpowanych były błędne lub błędnie rozłożone, gdy mamy do czynienia z funkcją probablistyczną. Tu, podobnie jak przy sekretności, nie możne obronić się przed niektórymi działaniami przeciwnika. Aktywny przeciwnik może tak pokierować komunikacją, by obliczenia graczy nieskorumpowanych były błędne, ale żąda się, by były jakoś zgodne z założeniami protokołu.

Prosty przykład z pracy [1] pokazuje nierozdzielność tych dwóch własności:

Przykład 3 *([1], Rozdział 2.1)*
(obliczanie xor dwóch słów)

1. *Każdy z dwóch graczy dostaje in_i. Wejście losowe nie jest potrzebne.*
2. *P_1 wysyła do gracza P_2 swoje wejście.*
3. *P_2 oblicza funkcję $in_1 \oplus in_2$ i odsyła wynik do P_1.*
4. *Każdy z graczy oddaje na wyjściu $in_1 \oplus in_2$.*

Gdzie funkcja $\oplus$ dla dwóch słów zerojedynkowych $a = (a_1, \ldots a_n)$, $b = (b_1, \ldots b_n)$ ($a_i, b_i \in \{0, 1\}$) jest zdefiniowana jako:

$$a \oplus b = (a_1, \ldots a_n) \oplus (b_1, \ldots b_n) = ((a_1 + b_1) \bmod 2, \ldots (a_n + b_n) \bmod 2).$$

Oczywiście protokół ten nie jest bezpieczny, bo przeciwnik korumpujący P_2 może poznać in_1 i wpłynąć nieoczekiwanie na out_1. Jeśli jednak rozważać sekretność i poprawność oddzielnie, to protokół trzyma sekretność w sposób pusty (bo i tak jeśli tylko przeciwnik pozna $out_1 = out_2$ i in_i, to może wyliczyć prosto in_{3-i}), a wynik uzyskany przez gracza P_1 jest w pewnym sensie poprawny przy zmianach komunikatu wysyłanego przez skorumpowanego wcześniej gracza P_2. Jednocześnie protokół ten nie jest bezpieczny w obliczu podanych dalej definicji nawet dla przeciwnika nieadaptywnego.

Od obliczenia idealnego należy więc żądać jednoczesnego zachowania sekretności i poprawności. Rozważa się więc obliczenie, w którym istnieje dodatkowy, zaufany (niekorumpowalny) gracz, do którego wszyscy wysyłają swoje wkłady w obliczenie funkcji, a on po obliczeniu, odsyła do wszystkich odpowiednie wyniki. Ponieważ działanie protokołu jest nieco różne w zależności od adaptywności przeciwnika, więc dokładniej omówię je w następnych dwóch podrozdziałach.

3.2. Bezpieczeństwo nieadaptywne

3.2.1. Proces rzeczywisty

Przez obliczenie wielopodmiotowe π funkcji f z przeciwnikiem nieadaptywnym rozumie się obliczenie zgodne z następującym schematem:

1. (a) Każdy z graczy P_i dostaje na wejściu parametr bezpieczeństwa k, wejście in_i i słowo losowe $rand_i$.

   (b) Przeciwnik $\mathcal{A}$ rozpoczyna z wejściem k, aux i $rand_0$. $\mathcal{A}$ dostaje też zbiór $B \in \mathcal{B}$ określający graczy skorumpowanych oraz $\{(i, in_i, rand_i) : \quad i \in B\}$[1]. Bez zmniejszenia ogólności zakłada się, że $aux = (aux_0, i_1, aux_{i_1}, \ldots i_m, aux_{i_m})$, gdzie i_j to wszystkie numery skorumpowanych graczy.

2. Inicjalizacja liczby rund komunikacji $r := 0$.

3. Dopóki istnieje nieskorumpowany gracz, który nie zakończył komunikacji, wykonywane są następujące kroki:

   (a) $r := r + 1$

   (b) Każdy nieskorumpowany gracz P_i, $i \notin B$ generuje komunikaty $m_{i,j,r}$ (ewentualnie puste) przeznaczone pozostałych graczy w bieżącej rundzie.

   (c) Przeciwnik $\mathcal{A}$ poznaje $\{m_{i,j,r} : j \in B\}$ i generuje (ewentualnie puste) odpowiedzi $m_{i,j,r}$, $i \in B$, $j \notin B$.

   (d) Każdy nieskorumpowany gracz dostaje przeznaczoną dla niego część komunikacji.

4. (a) Każdy z graczy P_i generuje wynik $\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_i$, przy czym wynikiem działania graczy skorumpowanych jest znak specjalny $\perp$[2].

   (b) Przeciwnik $\mathcal{A}$ generuje swój wynik $\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_0 = \mathrm{ADVR}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})$.

[1] Równoważnie można zakładać, że $\mathcal{A}$ dostaje tylko zbiór B i sam korumpuje graczy poznając ich wejścia.

[2] Można równoważnie zakładać, że skorumpowani gracze oddają na wyjściu swój wynik, a w wyniku przeciwnika jest ukryty zbiór graczy skorumpowanych, co jednak prowadzi do późniejszych komplikacji przy formalizacji.

Punkt 3 zaznacza, że w przyjętym modelu, o ile to tylko możliwe, żąda się, by komunikaty graczy nieskorumpowanych były przesłane przed komunikatami graczy nieskorumpowanych.

Wynikiem działania protokołu π jest:

$$\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand}) = \mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_0, \ldots \mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_n$$

co przez wyodrębnienie losowości można rozumieć jako rozkład prawdopodobieństwa:

$$\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux),$$

gdzie $\overrightarrow{rand}$ jest wybrane w sposób jednostajny i dalej rozumiane jako zespół rozkładów prawdopodobieństwa[3]:

$$\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*},$$

indeksowanym współczynnikiem bezpieczeństwa i możliwymi wejściami wszyskich graczy i przeciwnika.[4]

3.2.2. Proces idealny

Obliczenie idealne zaś jest zgodne z wcześniejszymi założeniami o sekretności i poprawności, i przebiega według schematu:

1. (a) Każdy z graczy P_i dostaje na wejściu in_i.
   (b) Przeciwnik idealny $\mathcal{S}$ rozpoczyna z parametrem bezpieczeństwa k, wejściem dodatkowym aux, słowem losowym $rand_S$ i zbiorem $B \in \mathcal{B}$ określającym graczy skorumpowanych oraz $\{(i, in_i) : \quad i \in B\}$.
   (c) Dodatkowy, zaufany (niekorumpowalny) gracz T dostaje na wejściu $rand_T$ i parametr bezpieczeństwa k.

2. (a) Każdy nieskorumpowany gracz P_i przesyła swoje wejście do zaufanego gracza T.
   (b) Przeciwnik idealny $\mathcal{S}$, jeśli jest aktywny, generuje komunikaty, które mają być wysłane do T od graczy skorumpowanych. Jeśli zaś jest pasywny, gracze P_i, $i \in B$ wysyłają do T swoje wejścia in_i.

3. Gracz T, oblicza wartość funkcji $f : \mathbb{N} \times D^n \times \{0,1\}^* \to D^n$ (pierwszy argument to parametr bezpieczeństwa, dalej wejścia graczy i wejście losowe T) i odsyła każdemu graczowi P_i wartość $f(k, \overrightarrow{in}, rand_T)_i$.

4. (a) Każdy nieskorumpowany gracz P_i oddaje na wyjściu $\mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_i$, gdzie $\overrightarrow{rand} = (rand_T, rand_S)$ - wynik nadesłany przez T, zaś wynikiem graczy skorumpowanych jest znak specjalny $\perp$.
   (b) Przeciwnik $\mathcal{S}$ generuje swój wynik $\mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_0 = \mathrm{ADVR}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})$.

[3] Definicje i twierdzenia dotyczące zespołów rozkładów prawdopodobieństw zamieszczone są w Dodatku A.

[4] Ponieważ definicja działa przez porównanie wyników dwóch obliczeń, oczywiście żądanie, by wyniki działania dwóch obliczeń lub by ich rozkłady tylko ze względu na losowe wejścia były równe jest zbyt wymagające. Dlatego od tej pory będzie mowa o zespołach rozkładów związanych z wynikami obliczenia w zależności od parametru bezpieczeństwa i wejść wszystkich graczy i przeciwnika.

Wynikiem działania procesu idealnego jest zespół rozkładów prawdopodobieństwa:

$$\mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*},$$

po wyborze jednostajnym
$\overrightarrow{rand} = (rand_T, rand_S)$, gdzie:

$$\mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand}) = \mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_0, \ldots \mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_n.$$

Jak widać cała komunikacja jest tu ukryta w mechanizmie działania gracza zaufanego. Przeciwnik idealny jest więc pozbawiony wielu informacji, do których ma dostęp przeciwnik rzeczywisty.

W odróżnieniu od poprzednich podejść (jak [6]), w których zakładano, że symulator jest ograniczony do jednokrotnego odpytania przeciwnika rzeczywistego bez znajomości jego mechanizmu (jednorazowa wyrocznia), tutaj nie stawiamy symulatorowi takich ograniczeń. Jeśli jednak symulator zna mechanizm każdego gracza i jednocześnie może wielokrotnie użyć przeciwnika rzeczywistego, trzeba założyć, że ma on ograniczony dostęp do losowości graczy. Zakłada się więc, że symulator może wielokrotnie losować wejścia graczy, ale tylko tych których skorumpował. Pozwala to na wielokrotne odpytywanie przeciwnika na różnych, ale sensowych wejściach. Dodatkowo, w zależności od modelu bezpieczeństwa, zmienne losowe, którymi dysponuje symulator, w zależności od parametru bezpieczeństwa i wejść, tworzą zespoły rozkładów odpowiednio równe (dokładnie, statystycznie lub obliczeniowo nierozróżnialne) rozkładom rzeczywistych zmiennych, z których pochodzą wejścia losowe graczy. Jednocześnie, rozważając niedeterministycznych przeciwników, nie ma sensu założenie, że przeciwnik idealny ma dostęp do losowości, której sam używa, by uruchamiać przeciwnika rzeczywistego na różnych danych, bo zdaje się nie dawać mu to dodatkowych informacji, a równocześnie można by nie przekazywać mu żadnej losowości, bo sam mógłby ją uzyskać. Stąd przeciwnik idealny, wykonując algorytm przeciwnika rzeczywistego, powinien dać mu własne wejście losowe.

W pracy [2] rozważa się dodatkowo w modelu idealnym możliwość dostarczania do zaufanego gracza wkładu przeciwnika (wtedy rozszerza się funkcję do $f : \mathbb{N}\times D^{n+1}\times\{0,1\}^* \to D^n$) oraz pobieranie przez przeciwnika wyniku od T ($f : \mathbb{N}\times D^n\times\{0,1\}^* \to D^{n+1}$) albo obydwa te podejścia naraz. Dostarczanie wkładu do T ma modelować wpływ symulatora na wynik graczy nieskorumpowanych, zaś pobranie wyniku przez $\mathcal{S}$ - możliwy wyciek informacji z protokołu do przeciwnika rzeczywistego wynikający z komunikacji.

3.2.3. Definicja bezpieczeństwa

Definicja bezpieczeństwa nieadaptywnego opiera się na porównaniu dwóch obliczeń przedstawionych w poprzednich rozdziałach. Aby powiedzieć, że dane obliczenie jest bezpieczne trzeba będzie dla każdego przeciwnika rzeczywistego $\mathcal{A}$ znaleźć przeciwnika idealnego $\mathcal{S}$, który mimo nałożonych restrykcji, będzie dobrze symulował zachowanie przeciwnika $\mathcal{A}$ i doprowadzał do wyprodukowania przez protokół podobnego wyniku całościowego w postaci zespołu rozkładów prawdopodobieństw po parametrze bezpieczeństwa i możliwych wejściach.

Widać, że przy porównywaniu tych dwóch przebiegów narzucają się różne modele odzwierciedlające wzajemny stosunek złożoności obliczeniowej przeciwnika rzeczywistego i idealnego oraz podobieństwo wynikowych zespołów rozkładów prawdopodobieństw. I tak w zależności od tego, czy wynikowe zespoły prawdopodobieństw są równe, statystycznie czy obliczeniowo nierozróżnialne, będę mówił o emulacji dokładnej, statystycznej albo obliczeniowej. Ze względu zaś na złożoność przeciwników rozróżnia się bezpieczeństwo informacyjno-teoretyczne (IT), gdy obydwaj przeciwnicy są obliczeniowo nieograniczeni, bezpieczeństwo uniwersalne, gdy

symulator jest oczekiwanie wielomianowy względem (nieograniczonej) złożoności przeciwnika rzeczywistego i bezpieczeństwo obliczeniowe[5], gdy obydwaj przeciwnicy działają w oczekiwanym czasie wielomianowym względem bezpieczeństwa. Należy tu zaznaczyć, że w oczywisty sposób bezpieczeństwo uniwersalne pociąga za sobą bezpieczeństwo IT oraz obliczeniowe, jakkolwiek bezpieczeństwo IT i obliczeniowe są nieporównywalne.

Definicja 1 *Niech π będzie wielopodmiotowym obliczeniem funkcji f dla n graczy i $\mathcal{B}$ strukturą ograniczającą przeciwnika. Protokół π bezpiecznie oblicza f względem przeciwników nieadaptywnych ze strukturą $\mathcal{B}$ i emulacja jest dokładna (odpowiednio statystyczna, obliczeniowa), jeśli dla każdego przeciwnika rzeczywistego $\mathcal{A}$ i każdego $B \in \mathcal{B}$ istnieje symulator $\mathcal{S}$ taki, że*

$$\mathrm{EXEC}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*} =_d \mathrm{IDEAL}_{f,\mathcal{S}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*}$$

(odpowiednio $\mathrm{EXEC}_{\pi,\mathcal{A}} =_s \mathrm{IDEAL}_{f,\mathcal{S}}$, $\mathrm{EXEC}_{\pi,\mathcal{A}} =_c \mathrm{IDEAL}_{f,\mathcal{S}}$). Jeśli dodatkowo dla każdego $\mathcal{A}$ można znaleźć $\mathcal{S}$ działający w oczekiwanym czasie wielomianowym względem $\mathcal{A}$, to mówi się o bezpieczeństwie uniwersalnym. Jeśli obydwaj przeciwnicy mają złożoność wielomianową względem k jest to bezpieczeństwo obliczeniowe.

Jeśli o przeciwniku $\mathcal{A}$ i symulatorze $\mathcal{S}$ założy się, że są pasywni, to mówi się, że protokół π $\mathcal{B}$-tajnie oblicza funkcję f.

Powyższa definicja spełnia nałożone wymaganie połączenia sekretności i poprawności z przykładu 3. Załóżmy, że $in_1, in_2 \in \{0,1\}$ i gracze nie dostają wejścia losowego oraz niech $\mathcal{B} = \{\emptyset, \{P_2\}\}$. Rozważmy przeciwnika nieadaptywnego $\mathcal{A}$ dla $B = \{P_2\}$, którego celem jest sprawienie, by P_1 zawsze zwracał jedynkę. $\mathcal{A}$ korumpuje P_2 i, niezależnie od bitu przysłanego od P_1, odsyła do P_1 jedynkę. Przeciwnik idealny $\mathcal{S}$, który miałby symulować $\mathcal{A}$, aby wpłynąć na wynik P_1 musiałby, przed wysłaniem w imieniu P_2 bitu do zaufanego gracza, poznać bit P_1.

3.3. Bezpieczeństwo adaptywne

3.3.1. Proces rzeczywisty

Ponieważ przeciwnik adaptywny może korumpować graczy w dowolnym momencie trwania obliczenia (byle by zgodnie ze strukturą ograniczającą $\mathcal{B}$) należy położyć tu większą wagę na kontakt protokołu ze środowiskiem zewnętrznym. W przypadku nieadaptywnym polegał on na pobraniu na początku obliczenia wejść przez graczy, pobraniu dodatkowego wejścia przez przeciwnika i oddaniu przez wszystkich wyników. W przypadku adaptywnym takie podejście jest niewystarczające. Komunikacja ze środowiskiem jak w przypadku nieadaptywnym nie pozwala bowiem na pełne przedstawienie współistnienia i przenikania się protokołów w czasie, ani na ich składanie ze sobą[6]. Gdy bowiem przeciwnik korumpuje gracza, widzi jego dane z różnych obliczeń - tych zakończonych i niezakończonych, więc należy pozwolić na dodatkowy przepływ danych z protokołu do środowiska i odwrotnie. Do tego celu wprowadza się dodatkową nieograniczoną maszynę $\mathcal{Z}$ zwaną środowiskiem. Dodatkową informację, jaką przeciwnik nieadaptywny dostawał na wejściu, przenosi się teraz do $\mathcal{Z}$. Z tego względu trzeba

[5] W pracy Canettiego [1] wyłącznie ten model jest używany w przypadku otwartych kanałów komunikacyjnych.

[6] Operacja składania protokołów, jak powiedziałem we wprowadzeniu, jest w tej pracy pomijana. Jest ona jednak bardzo istotna z praktycznego punktu widzenia i leży u podstaw teorii obliczeń wielopodmiotowych funkcji.

też zmienić definicję nie rozważając już samego przeciwnika, ale też dowolność środowiska, w jakim trwa obliczenie.

Dodatkową różnicą jest możliwość korumpowania graczy przez przeciwnika po zakończeniu działania protokołu. Oczywiście w praktycznym działaniu, przeciwnik może chcieć skorumpować gracza już po obliczeniu, by otrzymać nie tylko dodatkowe informacje o tym obliczeniu, których nie udało mu się dowiedzieć w trakcie komunikacji, ale też o innych obliczeniach, w których dany gracz uczestniczy. Mechanizm ten będzie widoczny w schematach protokołów, ale będzie on pomijany jako nieistotny dla tej pracy.

Rzeczywisty przeciwnik adaptywny działa zgodnie ze schematem:

1. (a) Każdy z graczy P_i dostaje na wejściu parametr bezpieczeństwa k, wejście in_i i słowo losowe $rand_i$.
   (b) Przeciwnik $\mathcal{A}$ rozpoczyna z wejściem k i losowym słowem $rand_0$. $\mathcal{A}$ dostaje również strukturę ograniczającą $\mathcal{B}$.
   (c) Środowisko $\mathcal{Z}$ dostaje na wejściu aux i $rand_{\mathcal{Z}}$.
2. Inicjalizacja liczby rund komunikacji $r := 0$. $\mathcal{A}$ dostaje od $\mathcal{Z}$ wstępną informację aux_0.
3. Dopóki istnieje nieskorumpowany gracz, który nie zakończył komunikacji, wykonywane są następujące kroki:
   (a) $r := r + 1$
   (b) Dopóki istnieje gracz nieaktywowany w tej rundzie:
      i. Dopóki $\mathcal{A}$ chce skorumpować jakiegoś gracza: $\mathcal{A}$ korumpuje wybranego P_i, po czym wysyła wszystkie uzyskane w ten sposób informacje do środowiska, zaś $\mathcal{Z}$ odsyła pewien komunikat aux_i do przeciwnika.
      ii. Przeciwnik aktywuje wybranego P_i. P_i dostaje wszystkie skierowane do niego komunikaty z poprzedniej rundy (jeśli $r > 1$), po czym generuje ewentualnie puste komunikaty do pozostałych graczy. Przeciwnik dostaje $\{m_{i,j,r} : P_j \quad jest \quad skorumpowany\}$.
   (c) Przeciwnik $\mathcal{A}$ generuje $\{m_{i,j,r} : P_i \quad jest \quad skorumpowany\}$.
4. (a) Każdy z graczy P_i generuje wynik $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_i$, przy czym wynikiem działania graczy skorumpowanych jest znak specjalny $\perp$.
   (b) Przeciwnik $\mathcal{A}$ generuje swój wynik $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})_0 = \mathrm{ADVR}_{\pi,\mathcal{A}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})$.
   (c) $\mathcal{Z}$ dostaje wszystkie wyniki.
5. Dopóki $\mathcal{Z}$ nie kończy działania:
   (a) $\mathcal{Z}$ wysyła do $\mathcal{A}$ żądanie skorumpowania P_i.
   (b) $\mathcal{A}$ korumpuje P_i, jeśli to możliwe.
   (c) $\mathcal{A}$ wysyła do $\mathcal{Z}$ informacje wynikające ze skorumpowania P_i.
6. $\mathcal{Z}$ kończy działanie z jakimś wynikiem.

Punkt 3. jest dokładniej określony niż ten sam w przypadku nieadaptywnym, gdyż przeciwnik adaptywny może na bieżąco wykorzystywać wszystkie zgromadzone informacje - i te zgromadzone w czasie śledzenia komunikacji, i te uzyskane przez korumpowanie graczy.

Wydawać by się mogło, że przeciwnik zawsze zdąży skorumpować wybranego gracza, ale nie jest to prawdą, bo liczba rund obliczenia może być zmienna i zależeć od losowości graczy.

Wynikiem działania protokołu jest zespół rozkładów prawdopodobieństw:

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,\overrightarrow{in},aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*},$$

po losowym wyborze $\overrightarrow{rand} = (rand_{\mathcal{Z}}, rand_0, \ldots rand_n)$.

3.3.2. Proces idealny

Obliczenie idealne jest zgodne ze schematem:

1. (a) Każdy z graczy P_i dostaje na wejściu in_i.
   (b) Przeciwnik $\mathcal{S}$ rozpoczyna z wejściem k i losowym słowem $rand_0$. $\mathcal{S}$ dostaje również strukturę ograniczającą $\mathcal{B}$.
   (c) Środowisko $\mathcal{Z}$ dostaje na wejściu aux i $rand_{\mathcal{Z}}$.
   (d) Dodatkowy, zaufany (niekorumpowalny) gracz T dostaje na wejściu $rand_T$ i parametr bezpieczeństwa k.
2. $\mathcal{S}$ dostaje od $\mathcal{Z}$ wstępną informację.
3. Dopóki $\mathcal{S}$ chce skorumpować jakiegoś gracza: $\mathcal{S}$ korumpuje wybranego P_i, po czym wysyła wszystkie uzyskane w ten sposób informacje do środowiska, zaś $\mathcal{Z}$ odsyła pewien komunikat do przeciwnika.
4. (a) Każdy nieskorumpowany gracz P_i przesyła swoje wejście do zaufanego gracza T.
   (b) Przeciwnik idealny $\mathcal{S}$, jeśli jest aktywny, generuje komunikaty, które mają być wysłane do T od graczy skorumpowanych. Jeśli zaś jest pasywny, skorumpowani gracze P_i wysyłają do T swoje wejścia in_i.
5. Gracz T, oblicza wartość funkcji $f : \mathbb{N} \times D^n \times \{0,1\}^* \to D^n$ i odsyła każdemu graczowi P_i wartość $f(\overrightarrow{in}, rand_T)_i$.
6. Po poznaniu $\{f(\overrightarrow{in}, rand_T)_i : P_i \quad jest \quad skorumpowany\}$, przeciwnik może skorumpować dodatkowo dowolną liczbę graczy, komunikując się przy tym ze środowiskiem.
7. (a) Każdy nieskorumpowany gracz P_i oddaje na wyjściu $\mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(k,\overrightarrow{in},aux,\overrightarrow{rand})_i$, gdzie $\overrightarrow{rand} = (rand_T, rand_S)$ - wynik nadesłany przez T, zaś wynikiem graczy skorumpowanych jest znak specjalny $\perp$.
   (b) Przeciwnik $\mathcal{S}$ generuje swój wynik $\mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(k,\overrightarrow{in},aux,\overrightarrow{rand})_0 = \mathrm{ADVR}_{f,\mathcal{S},\mathcal{Z}}(k,\overrightarrow{in},aux,\overrightarrow{rand})$.
8. Dopóki $\mathcal{Z}$ nie kończy działania:
   (a) $\mathcal{Z}$ wysyła do $\mathcal{S}$ żądanie skorumpowania P_i.
   (b) $\mathcal{S}$ korumpuje P_i, jeśli to możliwe.
   (c) $\mathcal{S}$ wysyła do $\mathcal{Z}$ informacje wynikające ze skorumpowania P_i.
9. $\mathcal{Z}$ kończy działanie z jakimś wynikiem.

Aby zrównoważyć większe niż w przypadku nieadaptywnym możliwości przeciwnika, pozwala się, oprócz fazy korumpowania po zakończeniu protokołu, na drugą fazę po uzyskaniu wyników od gracza zaufanego. Rozważmy następujący przykład przy założeniu tajności kanałów komunikacyjnych:

Przykład 4 *Gracze $P_1, \ldots P_{n-1}$ wybierają lidera wysyłając swoje wejścia do P_n. Liderem zostaje P_i o największym in_i (jeśli kilka takich samych, to rozstrzygnięcie następuje na podstawie $rand_n$). Gracz P_n odsyła numer lidera do wszystkich graczy. Każdy gracz zwraca jako wynik numer lidera.*

Załóżmy, że pasywny przeciwnik adaptywny może skorumpować najwyżej dwóch graczy i P_n jest niekorumpowalny (obliczenie rzeczywiste będzie bardzo podobne do idealnego). Obliczenie takie powinno być bezpieczne, gdyż jedynymi informacjami, jakie dostaje przeciwnik, są wejścia graczy przez niego skorumpowanych i numer lidera, który jest znany przez każdego gracza. Jeśli symulator byłby pozbawiony drugiej fazy korumpowania, to protokół ten byłby niebezpieczny, bo nie dałoby się zasymulować przeciwnika, który na początku korumpuje dowolnego gracza, a potem korumpuje lidera P_l i oddaje na wyjściu in_l.

3.3.3. Definicja bezpieczeństwa

Definicja bezpieczeństwa adaptywnego różni się od wersji nieadaptywnej jedynie tym, że działania przeciwnika są uwzględniane przy dowolnym zachowaniu środowiska, przy czym złożoność czasowa środowiska nie ma znaczenia przy porównywaniu złożoności przeciwników.

Definicja 2 *Niech π będzie wielopodmiotowym obliczeniem funkcji f dla n graczy i $\mathcal{B}$ strukturą ograniczającą przeciwnika. Protokół π bezpiecznie oblicza f względem przeciwników adaptywnych ze strukturą $\mathcal{B}$ i emulacja jest dokładna (odpowiednio statystyczna, obliczeniowa), jeśli dla każdego przeciwnika rzeczywistego $\mathcal{A}$ i każdego środowiska $\mathcal{Z}$ istnieje symulator $\mathcal{S}$ taki, że*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*} =_d \mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(k, \overrightarrow{in}, aux)_{k\in\mathbb{N},<\overrightarrow{in},aux>\in\{0,1\}^*\times\{0,1\}^*}$$

(odpowiednio $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} =_s \mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}$, $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} =_c \mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}$). Jeśli dodatkowo dla każdego $\mathcal{A}$ można znaleźć $\mathcal{S}$ działający w oczekiwanym czasie wielomianowym względem $\mathcal{A}$, to mówi się o bezpieczeństwie uniwersalnym. Jeśli obydwaj przeciwnicy mają złożoność wielomianową względem k jest to bezpieczeństwo obliczeniowe.

Jeśli o przeciwniku $\mathcal{A}$ i symulatorze $\mathcal{S}$ założy się, że są pasywni, to mówi się, że protokół π $\mathcal{B}$-tajnie oblicza funkcję f.

Rozdział 4

Wpływ adaptywności na bezpieczeństwo obliczenia

4.1. Wprowadzenie

Oczywiście, intuicyjnie, przeciwnik adaptywny może więcej niż nieadaptywny, więc bezpieczeństwo nieadaptywne powinno być warunkiem koniecznym dla bezpieczeństwa adaptywnego. Jest tak w rzeczywistości:

Fakt 1 *([1], Rozdział 5.2)*
Jeśli protokół π obliczający funkcję f jest bezpieczny wobec przeciwnika adaptywnego dla struktury $\mathcal{B}$, to (w tym samym modelu bezpieczeństwa) jest bezpieczny wobec przeciwnika nieadaptywnego dla tej samej struktury $\mathcal{B}$.

Dowód Rozważmy dowolnego przeciwnika nieadaptywnego $\mathcal{A}_{na}$ ograniczonego strukturą $\mathcal{B}$ i jego działanie na dowolnym $B \in \mathcal{B}$. Rozpatrzmy przeciwnika adaptywnego $\mathcal{A}_a$ i środowisko $\mathcal{Z}$ takie, że $\mathcal{Z}$ dostaje jako *aux* zbiór B, wejście dodatkowe $\mathcal{A}_{na}$ i po ich przekazaniu do gracza pozostaje bierne, zaś $\mathcal{A}_a$ - znając B - korumpuje wszystkich $P_i \in B$ przed rozpoczęciem komunikacji i dalej działa dokładnie tak samo jak $\mathcal{A}_{na}$. Oczywiście dla $\mathcal{A}_a$ istnieje symulator $\mathcal{S}_a$. Ponieważ $\mathcal{S}_a$ dostaje B przed rozpoczęciem działania, to można wymagać, by $\mathcal{S}_a$ skorumpował wszystkich $P_i \in B$ przed komunikacją z zaufanym graczem, czyli ograniczyć się do pierwszej fazy korumpowania. Taki symulator jest więc nieadaptywny (jeśli pominiemy środowisko) i symuluje działanie $\mathcal{A}_{na}$. □

Jest jasne, że przeciwnik rzeczywisty widzący całą komunikację ma większe możliwości niż taki, który jest ograniczony do komunikacji dotyczącej tylko graczy, których skorumpował, a jednocześnie obydwaj przeciwnicy idealni mają takie same możliwości. Stąd:

Fakt 2 *Bepieczeństwo protokołu przy otwartych kanałach komunikacyjnych pociąga za sobą bezpieczeństwo tego protokołu w modelu kanałów tajnych.*

Dowód Dla każdego przeciwnika przy tajnych kanałach można rozważyć takiego samego przy kanałach otwartych, który po prostu nie podsłuchuje części komunikacji nie dotyczącej graczy, których skorumpował. □

Kolejny fakt, choć oczywisty, jest nieco ciekawszy:

Fakt 3 *W przypadku adaptywnym, w dowolnym modelu obliczeń, bezpieczeństwo protokołu wobec przeciwników, którzy w wyniku oddają parę, w której na pierwszym miejscu jest lista*

kolejno korumpowanych graczy, pociąga za sobą bezpieczeństwo adaptywne tego protokołu w tym samym modelu.

Dowód Weźmy dowolnego przeciwnika rzeczywistego $\mathcal{A}_a$, który na danych wejściach protokołu zwraca jako wynik $out_{\mathcal{A}_a}$. Rozważmy przeciwnika adaptywnego $\mathcal{A}'_a$, który działa dokładnie jak $\mathcal{A}_a$, ale jako wynik daje $(L, out_{\mathcal{A}_a})$, gdzie L jest listą wszystkich skorumpowanych graczy (ewentualnie pustą jeśli do skorumpowania nie doszło). Jeśli przeciwnik idealny $\mathcal{S}'_a$ dobrze symuluje $\mathcal{A}'_a$, to $\mathcal{S}_a$, który uruchamia symulator $\mathcal{S}'_a$ i daje jego obcięty wynik dobrze symuluje $\mathcal{A}_a$. □

Trzeba zauważyć, że zwracanie informacji o kolejności skorumpowanych graczy może już być zawarte w strategii przeciwnika. Na przykład może on próbować zyskać przewagę nad przeciwnikiem idealnym przez podawanie w wyniku przekłamanej informacji na ten temat. Nie zmienia to jednak powyższego rozumowania. Fakt ten może pomóc przy dowodzeniu bezpieczeństwa, gdyż zawsze można wymagać, by przeciwnik rzeczywisty zwracał taką listę, co może upraszczać rozumowanie.

4.1.1. Początkowa adaptywność

Nie jest oczywiste, gdzie zaczyna się realna przewaga adaptywności. Canetti w pracy [1] rozważa przeciwnika początkowo adaptywnego $\mathcal{A}_{ia}$. Przeciwnik taki może tylko przed fazą komunikacji dynamicznie - w sposób adaptywny - korumpować graczy. $\mathcal{A}_{ia}$ może więc przy korumpowaniu sugerować się informacjami uzyskanymi z poprzednich korumpowań. Od wejścia w fazę komunikacji, protokół przebiega zgodnie ze schematem dla przeciwnika nieadaptywnego, tyle że wejście przeciwnika zamiast zbioru B zawiera strukturę $\mathcal{B}$ i przeciwnik zaraz po pierwszej fazie schematu może korumpować graczy zgodnie z $\mathcal{B}$. Taki $\mathcal{A}_{ia}$ ma, wydawać by się mogło, pewną przewagę nad przeciwnikiem nieadaptywnym. Canetti twierdzi jednak, że bezpieczeństwo przeciw przeciwnikom początkowo adaptywnym jest równoważne bezpieczeństwu nieadaptywnemu i jasne jest przy tym, że przeciwnik początkowo adaptywny jest co najmniej tak mocny jak nieadaptywny.

Fakt 4 *([1], Rozdział 4.2, Uwaga 5)*
Jeśli protokół π obliczający funkcję f jest bezpieczny wobec przeciwnika nieadaptywnego dla struktury $\mathcal{B}$, to (w tym samym modelu bezpieczeństwa) jest bezpieczny wobec przeciwnika początkowo adaptywnego dla tej samej struktury $\mathcal{B}$.

Autor pracy [1] podaje następujący szkic dowodu faktu 4:

Niech $\mathcal{A}'$ będzie nieadaptywnym rzeczywistym przeciwnikiem, który jako dodatkowe wejście dostaje stan wewnętrzny przeciwnika początkowo adaptywnego $\mathcal{A}$ w momencie, gdy ten kończy korumpować graczy (*), i dalej naśladuje dokładnie działanie $\mathcal{A}$. Niech $\mathcal{S}'$ będzie gwarantowanym przez definicję symulatorem dla $\mathcal{A}'$. Konstruuje się $\mathcal{S}$ jak następuje. Na początku $\mathcal{S}$ w idealny sposób naśladuje żądania skorumpowania gracza przez $\mathcal{A}$. Niech σ będzie stanem $\mathcal{A}$ w momencie, gdy jest gotowy do wejścia w interakcję z graczami. $\mathcal{S}$ uruchamia $\mathcal{S}'$ z dodatkowym wejściem σ. Jest jasne, że $\mathcal{S}$ jest poprawnym początkowo adaptywnym przeciwnikiem idealnym symulującym $\mathcal{A}$.

Zauważmy, że szkic ten posiada pewną lukę, gdyż w momencie (*) zmienia się protokół. Przeciwnik nieadaptywny jest bowiem rozpatrywany przy innej dziedzinie wejść dodatkowych aux. Istnienie $\mathcal{S}'$ nie jest więc pewne. W przedstawionym szkicu należałoby żądać, by wejście dodatkowe symulatora nieadaptywnego było jednakowe z aux początkowo adaptywnego. Nie rozważa się też możliwość pomyłki symulatora $\mathcal{S}'$, co zawęża rozumowanie do przypadku emulacji dokładnej.

Wydaje się, że da się udowodnić fakt 4, jeśli założyć, że przeciwnik początkowo adaptywny nie sugeruje się przy kolejnych żądaniach skorumpowania wejściami losowymi już skorumpowanych graczy, a tylko ich wejściami *in* (np. $\mathcal{A}$ poznaje losowość skorumpowanych przez siebie graczy dopiero po wejściu w fazę komunikacji).

Można natomiast wyobrazić sobie protokół, którego działanie i wynik są mocno uzależnione od losowości, będący bezpiecznym nieadaptywnie, ale niebezpiecznym wobec przeciwnika początkowo adaptywnego. Demonstruje to przykład:

Przykład 5 *Niech n i m będą odpowiednio ze sobą związanymi liczbami (np. $n = 61$, $m = 365$) i niech przeciwnik pasywny może korumpować wszystkich graczy prócz P_n ($\mathcal{B} = 2^{\{P_1,\dots P_{n-1}\}}$).*

1. *Każdy z graczy P_i, $i \neq n$ dostaje na wejściu pewną liczbę naturalną jako in_i, (ewentualna nagroda gracza) oraz losowość $rand_i$ (los na loterii) wybraną w sposób jednostajny z przedziału $\{0, \dots m\}$. P_n ma puste wejście.*

2. *Każdy z graczy P_i, $i \neq n$ wysyła swoją losowość do P_n, który sprawdza, czy jest gracz z taką samą losowością i, jeśli nie ma, odsyła do P_i true, a w przeciwnym razie false (P_n jest arbitrem, który daje nagrodę graczom z losem jedynym na loterii i każe płacić graczom z losem konfliktowym).*

3. *Każdy gracz P_i, $i \neq n$ zwraca $rand_i$ oraz in_i jeśli dostał true i $-in_i$ w przeciwnym przypadku. Wyjście P_n stanowi zbiór graczy, którzy brali udział w jakiejkolwiek kolizji.*

Fakt 5 *Przedstawiony protokół przy tajnych kanałach jest nieadaptywnie bezpieczny wobec przeciwników pasywnych.*

Dowód Rozważmy przeciwnika nieadaptywnego $\mathcal{A}_{na}$ ze zbiorem $B \in \mathcal{B}$. Zauważmy, że jedynymi informacjami przeciwnika $\mathcal{A}_{na}$ są wejścia i losowość graczy skorumpowanych oraz ich wyniki. Symulator $\mathcal{S}_{na}$ oddaje wszystkie in_i graczy $P_i \in B$ do gracza zufanego i czeka na wyniki, potem zaś uruchamia przeciwnika rzeczywistego, dając mu na wejściu B i odpowiednie in_i. Losowe wejścia skorumpowanych graczy symulator wybiera przy pomocy (nieefektywnego) algorytmu:

(niech dla uproszczenia zbiór graczy skorumpowanych to $\{P_1, \dots P_k\}$, $k < n$ oraz niech out_i będą wynikami działania zaufanego gracza)

$B_1 := \emptyset$, $B_2 := \emptyset$.

$x := \sum_{i \leq k} [out_i > 0]$.

for($i := 1$; $i < k + 1$; $i++$)

jeśli $out_i > 0$ to losuj $rand_i$, aż $rand_i \notin B_1 \cup B_2$, po czym $B_1 := B_1 \cup \{rand_i\}$

jeśli $out_i < 0$ to losuj $rand_i$, aż $rand_i \notin B_1$ oraz $(|B_2| + [rand_i \notin B_2] \leq n - 1 - x)$, po czym $B_2 := B_2 \cup \{rand_i\}$

Gdy przeciwnik wyjdzie z fazy komunikacji (która nie daje mu żadnych dodatkowych informacji), $\mathcal{S}$ przekazuje mu wyniki graczy skorumpowanych i oddaje na wyjściu wynik, jaki zwraca $\mathcal{A}$. Protokół jest więc bezpieczny wobec pasywnego przeciwnika nieadaptywnego. □

Fakt 6 *Przedstawiony protokół przy tajnych kanałach nie jest początkowo adaptywnie bezpieczny wobec przeciwników pasywnych.*

Dowód Celem przeciwnika początkowo adaptywnego $\mathcal{A}_{ia}$ jest by dowiedzieć się jak najwięcej o uczestnikach loterii i jak najmniej stracić (korumpując gracza, zawiera on umowę, że opłaci jego ewentualną przegraną). $\mathcal{A}_{ia}$, wykorzystując paradoks urodzinowy, zyska przewagę nad przeciwnikiem idealnym. Przeciwnik korumpuje graczy począwszy od P_1 aż do P_{n-1}, chyba że wcześniej zauważy, iż ostatnio skorumpowany gracz P_i ma losowość $rand_i$ taką jak jeden z już skorumpowanych. Z prostych obliczeń wynika, że dla $n = 61$ oraz $m = 365$, szansa że $\mathcal{A}$ skorumpuje wszystkich $n - 1 = 60$ możliwych graczy i nie napotka kolizji jest bliska 0:

$$\frac{364}{365}\frac{363}{365}\ldots\frac{365-60+1}{365} \approx 0{,}01.$$

Więc prawie zawsze przeciwnik nie musi korumpować wszystkich graczy, a co więcej ostatni przez niego skorumpowany gracz P_l powoduje prawie zawsze kolizję. W ten sposób prawie zawsze gracz P_l jest w zbiorze wyjściowym gracza P_n. Niech więc wynikiem działania $\mathcal{A}_{ia}$ będzie numer l ostatniego ze skorumpowanych graczy P_l (o największym numerze). Przypuśćmy, że przeciwnik idealny $\mathcal{S}_{ia}$ chciałby zasymulować działanie $\mathcal{A}_{ia}$. Po pierwsze musi skorumpować przynajmniej dwóch graczy, po drugie zaś jego wynikiem końcowym musi być największy numer skorumpowanego gracza l. Szansa, że gracz zaufany tak wybierze losy graczom, że P_l dostanie los kolizyjny wynosi tylko:

$$1 - \underbrace{\frac{364}{365}\frac{364}{365}\ldots\frac{364}{365}}_{59} \approx 0{,}15...$$

□

4.2. Rozróżnienie dla aktywnego przeciwnika i więcej niż dwóch graczy

Prosty przykład pokazuje, że dla $n \geq 3$ bezpieczeństwo adaptywne nie jest równoważne bezpieczeństwu nieadaptywnemu dla przeciwników aktywnych.

Przykład 6 *([2], Rozdział 2.6)*
Niech $n = 3$, obliczana funkcja: $f_{act}(k, \perp, \perp, (in_1, in_2), rand_T) = (in_1, in_2, \perp)$ (wprowadzony symbol $\perp$ oznacza puste wejście lub wyjście funkcji odpowiednio) i struktura przeciwnika $\mathcal{B} = 2^{\{P_1,P_2,P_3\}}$. Protokół π_{act} działa według schematu:

1. *Gracze P_1 i P_2 mają puste wejścia. Wejście P_3 składa się z dwóch bitów $b_1, b_2 \in \{0, 1\}$. P_3 nie ma wejścia losowego.*
2. *P_3 wysyła b_1 do P_1.*
3. *P_3 wysyła b_2 do P_2.*
4. *Każdy P_i zwraca b_i dla $i \in \{1, 2\}$. P_3 zwraca $\perp$.*

Fakt 7 *([2], Rozdział 2.6, Twierdzenie 25)*
Protokół π_{act} oblicza bezpiecznie funkcję f_{act} wobec nieadaptywnego przeciwnika aktywnego ograniczonego strukturą $\mathcal{B}$ i bezpieczeństwo jest uniwersalne, a emulacja dokładna.

Dowód Rozważmy nieadaptywnego przeciwnika $\mathcal{A}$, który korumpuje P_3. Przeciwnik idealny $\mathcal{S}$ korumpuje P_3 i przekazuje do $\mathcal{A}$ bity b_1, b_2. $\mathcal{S}$ przekazuje ewentualnie zmienione przez $\mathcal{A}$ bity b'_1, b'_2 do gracza zaufanego jako wejście P_3. $\mathcal{S}$ zwraca wynik $\mathcal{A}$ i kończy. Jasno widać, że globalny wynik działania procesu idealnego jest identyczny do wyniku procesu rzeczywistego.

Powyższy symulator może być w łatwy sposób rozszerzony na przypadek, gdy $\mathcal{A}$ korumpuje P_3 i P_1 (lub P_3 i P_2) przez przesłanie do gracza zaufanego bitów 0 i b'_2 (lub odpowiednio b'_1, 0) jako wejście P_3, gdzie b'_2 (b'_1) jest bitem zmienionym przez $\mathcal{A}$ przeznaczonym dla P_2 (P_1).

Teraz niech $\mathcal{A}$ korumpuje P_1 (i/lub P_2). Symulator $\mathcal{S}$ korumpuje P_1 (i/lub P_2) w procesie idealnym, przekazuje do zaufanego gracza puste wejście (lub wejścia) $\perp$ jako P_1 (i/lub P_2) i dostaje od niego b_1 (i/lub b_2). $\mathcal{S}$ dostarcza do $\mathcal{A}$ b_1 (i/lub b_2) jako wiadomość przesyłaną przez P_3 do P_1 (i/lub P_2), zwraca to co $\mathcal{A}$ i kończy działanie. Znowu łatwo zobaczyć, że wynik działania procesu idealnego jest taki sam jak rzeczywistego.

Na koniec, jeśli $\mathcal{A}$ korumpuje wszystkich graczy, symulator korumpuje wszystkich w procesie idealnym, wręcza $\mathcal{A}$ ich wejścia i postępuje zgodnie z instrukcjami $\mathcal{A}$, co znowu prowadzi proces idealny do zwrócenia jednakowego zespołu prawdopodobieństw jak proces rzeczywisty. □

Jednocześnie zachodzi:

Fakt 8 *([2], Rozdział 2.6, Twierdzenie 26)*
Protokół π_{act} obliczenia f_{act} nie jest bezpieczny w sposób uniwersalny, statystyczny ani obliczeniowy, adaptywnie przy strukturze gracza $\mathcal{B}$.

Dowód Pokażę przeciwnika adaptywnego, który nie może być zasymulowany przez przeciwnika idealnego nieograniczonego obliczeniowo nawet z emulacją obliczeniową. Celem $\mathcal{A}$ jest zapewnienie, że zawsze, gdy $b_1 = 1$, wynikiem działania P_2 będzie 0, przy czym gracz P_3 będzie korumpowany tylko gdy $b_1 = 1$, a P_2 nigdy (warto zauważyć, że każdy z tych celów może być osiągnięty osobno przez aktywnego, adaptywnego przeciwnika idealnego). $\mathcal{A}$ korumpuje P_1 i jeśli $b_1 = 0$, kończy a w przeciwnym przypadku korumpuje P_3 i przekazuje do P_2 wartość $b'_2 = 0$. Oczywiście $\mathcal{A}$ osiąga wszystkie postawione cele. Niech $\mathcal{S}$ będzie nieograniczonym czasowo symulatorem, który nigdy nie korumpuje P_2 (skorumpowanie P_2 może nastąpić tylko z zaniedbywalnym prawdopodobieństwem). Jeśli $\mathcal{S}$ powoduje, że P_2 zwraca 0 z przeważającym prawdopodobieństwem zawsze gdy $b_1 = 1$, to P_3 dla $b_1 = 1$ musi także być skorumpowane z przeważającym prawdopodobieństwem w pierwszej fazie korumpowania w procesie idealnym. $\mathcal{S}$ jednak przed interakcją z graczem zaufanym, przed skorumpowaniem P_3 nie ma żadnym informacji o b_1 i b_2, więc P_3 jest korumpowane z jednakowym prawdopodobieństwem dla wszystkich możliwych wejść b_1 i b_2. Stąd $\mathcal{S}$ korumpuje P_3 z przeważającym prawdopodobieństwem, gdy $b_1 = 0$, podczas gdy w tym przypadku P_3 nigdy nie jest korumpowany w procesie rzeczywistym. □

Na podstawie powyższych dwóch faktów można powiedzieć:

Twierdzenie 1 *Dla przeciwników aktywnych bezpieczeństwo adaptywne jest mocniejsze od bezpieczeństwa nieadaptywnego w dowolnym modelu bezpieczeństwa, przy dowolnej emulacji, jeśli w obliczeniu uczestniczy przynajmniej trzech graczy.*

Protokół z przykładu 6 nie jest bezpieczny nieadaptywnie przy otwartych kanałach komunikacyjnych w dowolnym modelu bezpieczeństwa nawet dla przeciwników pasywnych i powyższe rozważania należy ograniczyć do kanałów tajnych, bowiem pasywny przeciwnik nieadaptywny, który nie korumpuje żadnego z graczy może zwrócić $b_1 \oplus b_2$, czego nie może

uczynić przeciwnik idealny. Jeśli jednak pozwoli się na zwrot obliczanej funkcji dla przeciwnika idealnego, zmieniając funkcję z przykładu na

$$f'_{act}(k, \perp, \perp, (in_1, in_2), rand_T) = (in_1, in_2, \perp, (in_1, in_2)),$$

obydwa fakty i drugi dowód pozostają w mocy, a dowód pierwszego faktu jest prosty (jeśli $P_3 \in B$, to symulator ma oczywiście takie same możliwości jak przeciwnik rzeczywisty, w przeciwnym przypadku przeciwnicy są de facto pasywni, a symulator dostaje całą komunikację po zakończeniu protokołu).

4.3. Obliczenia dwustronne

Zajmę się w tym rozdziale motywowanym przez rozdział 2.7 pracy [2] obliczeniami w różnych modelach przy udziale dwóch graczy. Przedstawię najpierw dowód równoważności bezpieczeństwa adaptywnego i nieadaptywnego przy założeniu tajnych kanałów, po czym na jego podstawie - przechodząc przez różnicę dla zmienionej definicji z wyjściem funkcji dla przeciwnika idealnego - przejdę do dyskusji przy otwartych kanałach.

4.3.1. Tajne kanały komunikacyjne

Autorzy [2] udowodnili następujące twierdzenie:

Twierdzenie 2 *([2], Rozdział 2.7) Ograniczając się do przeciwników deterministycznych (bez losowego wejścia), przy tajnych kanałach komunikacyjnych, jeśli protokół π $\mathcal{B}$-bezpiecznie nieadaptywnie oblicza funkcję f (uniwersalnie, obliczeniowo) i emulacja jest dokładna (statystyczna, obliczeniowa), to w tym samym modelu π jest bezpieczny adaptywnie.*

Dowód (szkic) Dla dokładnej emulacji dowód jest nieco prostszy. Dla dowolnego przeciwnika adaptywnego $\mathcal{A}_a$ konstruuje się następujący symulator $\mathcal{S}_a$:

1. $\mathcal{S}_a$ dostaje parametr bezpieczeństwa k i dodatkowe wejście aux_0 od środowiska $\mathcal{Z}$.
2. $\mathcal{S}_a$ uruchamia $\mathcal{A}_a$, przekazując mu k i aux (symulując środowisko). Jako że komunikacja między P_1 i P_2 jest tajna, $\mathcal{A}_a$ nie spodziewa się dostać żadnych informacji, póki nie skorumpuje jednego z graczy. $\mathcal{S}_a$ kontynuuje wykonanie $\mathcal{A}_a$ aż:
   (a) $\mathcal{A}_a$ zechce skorumpować P_i lub
   (b) $\mathcal{A}_a$ zakończy z wynikiem out.

   Jeśli zdarzy się 2.(b), $\mathcal{S}_a$ kończy zwracając out.
3. $\mathcal{S}_a$ korumpuje P_i, poznaje in_i i dostaje aux_i od $\mathcal{Z}$.
4. $\mathcal{S}_a$ konstruuje symulator $\mathcal{S}_{na}$ następującego przeciwnika nieadaptywnego $\mathcal{A}_{na}$ korumpującego P_i:
   (a) $\mathcal{A}_{na}$ dostaje na wejściu $B = \{P_i\}$, (i, in_i) oraz dodatkową informację (aux_0, i, aux_i).
   (b) $\mathcal{A}_{na}$ uruchamia $\mathcal{A}_a$ na swoich danych i czeka, aż ten zechce skorumpować P_i. Jeśli $\mathcal{A}_a$ nie skorumpuje żadnego gracza lub zażąda skorumpowania P_{3-i}, to kończy i zwraca $error_1$ (wiemy, że się to nie zdarzy, bo $\mathcal{A}_a$ jest deterministyczny). Przekazuje do $\mathcal{A}_a$ informacje o P_i oraz aux_i jako środowisko. Kontynuuje wykonanie $\mathcal{A}_a$ dopóki:

i. $\mathcal{A}_a$ skończy z wynikiem *out* lub

ii. $\mathcal{A}_a$ zażąda skorumpowania P_{3-i}.

$\mathcal{A}_{na}$ kończy i jeśli zdarzyło się i., to wynikiem jego działania jest *out*, a w przeciwnym przypadku $error_2$.

$\mathcal{S}_a$ uruchamia symulator $\mathcal{S}_{na}$ korumpujący P_i i przekazuje mu potrzebne informacje, po czym pobiera od niego (jako zaufany gracz) komunikat, który $\mathcal{S}_{na}$ chce wysłać do gracza zaufanego i wysyła. $\mathcal{S}_a$ odbiera od zaufanego gracza wynik dla P_i i przekazuje go do $\mathcal{S}_{na}$. Jeśli $\mathcal{S}_{na}$ nie zwraca błędu, to $\mathcal{S}_a$ kończy z jego wynikiem. W przeciwnym przypadku kontynuuje działanie.

5. Korumpuje P_{3-i} i dostaje aux_{3-i} od $\mathcal{Z}$.

6. $\mathcal{S}_a$ wykonuje następującą procedurę *sim_both*:

   (a) Symuluj działanie $\mathcal{A}_a$ ze znanymi wejściami in_1, in_2, wejściami losowymi oraz aux_1, aux_2 (jeśli potrzeba).

   (b) Jeśli $\mathcal{A}_a$ korumpuje obydwu graczy, to $\mathcal{S}_a$ zwraca jego wynik. W przeciwnym przypadku (gracz P_{3-i} nie został skorumpowany) wraca do (a).

$\mathcal{S}_a$ dobrze symuluje $\mathcal{A}_a$ (pomijam szczegóły), a jego czas oczekiwany jest wielomianowy w stosunku do czasu $\mathcal{A}_a$, bo średnia liczba restartów *sim_both*, jeśli prawdopodobieństwo skorumpowania obydwu graczy jest niezerowe:

$$P(\mathcal{S}\text{ dojdzie do 6.})\sum_{j=1}^{\infty} jP(sim\_both\text{ zostanie restartowane dokładnie j razy}) =$$
$$p\sum_{j=1}^{\infty} jp(1-p)^j = p^2(p-1)\sum_{j=1}^{\infty} -j(1-p)^{j-1} = p^2(p-1)\sum_{j=1}^{\infty}((1-p)^j)' =$$
$$p^2(p-1)(\sum_{j=1}^{\infty}((1-p)^j))' = p^2(p-1)(\tfrac{1-p}{p})' = p^2(p-1)\tfrac{-1}{p^2} = 1-p < 1.$$

Niestety w przypadku emulacji statystycznej lub obliczeniowej nie można opierać się na powyższym symulatorze, bo może on nigdy nie zakończyć działania. Symulator $\mathcal{S}_{na}$ konstruowany w punkcie 2. może się z małym prawdopodobieństwem pomylić i dać sygnał do skorumpowania P_i, choć w rzeczywistości $\mathcal{A}_{na}$ (i $\mathcal{A}_a$) wcale nie korumpuje P_i, a wtedy *sim_both* wejdzie w nieskończoną pętlę. Okazuje się jednak, że wystarczy zmienić tę procedurę, by otrzymać odpowiedni symulator (pomijam szczegóły), który ma odpowiedni czas działania:

(a) $r := 1$

(b) Symuluj działanie $\mathcal{A}_a$ ze znanymi wejściami in_1, in_2, wejściami losowymi oraz aux_1, aux_2 (jeśli potrzeba).

(c) Jeśli $\mathcal{A}_a$ korumpuje obydwu graczy, to $\mathcal{S}_a$ zwraca jego wynik. W przeciwnym przypadku (gracz P_{3-i} nie został skorumpowany) $r := r + 1$.

(d) Jeśli $r > k$ zakończ z wynikiem *loop*, w przeciwnym razie wróć do (a).

□

4.3.2. Otwarte kanały komunikacyjne

Zajmijmy się teraz przypadkiem otwartych kanałów, w którym przeciwnik widzi całą komunikację nawet bez korumpowania jakiegokolwiek gracza. Rozpatrzmy zmienioną definicję, kiedy pozwala się na zwrot obliczanej funkcji do przeciwnika idealnego. Poniższy przykład, naśladujący przykład 6 z poprzedniego podrozdziału, pokazuje przewagę adaptywności w tym przypadku:

Przykład 7 *([2], Rozdział 2.7.2)*
Niech $\mathcal{B} = \{\emptyset, \{P_1\}\}$. *Obliczenie funkcji* $f_2(in_1, \perp, \perp) = (\perp, in_1, in_1)$, *gdzie* $in_1 \in \{0,1\}$ *przebiega następująco:*

1. *Jedynym wejściem jakie dostają gracze jest bit* x *dla* P_1.
2. P_1 *wysyła* x *do* P_2.
3. P_1 *wysyła* x *do* P_2.
4. P_2 *zwraca bit otrzymany od* P_1 *w kroku 3. Obydwaj gracze kończą działanie.*

Protokół powyższy jest nieadaptywnie bezpieczny. Jeśli bowiem $B = \emptyset$, symulator $\mathcal{S}$ przeciwnika rzeczywistego $\mathcal{A}$ może po prostu poczekać na wynik otrzymany od zaufanego gracza i wręczając ten wynik $\mathcal{A}$, oddać na wyjściu wynik zwracany przez przeciwnika rzeczywistego. Jeśli zaś $B = \{P_1\}$, symulator już na początku dowiaduje się bitu x i łatwo doprowadza proces idealny do wyniku globalnego jednakowego do wyniku procesu rzeczywistego.

Natomiast przeciwnik adaptywny, którego celem jest, by P_2 zawsze zwracał 1, przy jednoczesnym korumpowaniu P_1 tylko jeśli to konieczne, łatwo zyskuje przewagę nad przeciwnikiem idealnym (podobnie jak w przykładzie 6) i przeczy bezpieczeństwu adaptywnemu protokołu.

Zajmijmy się teraz przypadkiem otwartych kanałów w oryginalnej definicji Canettiego. Po pierwsze należy zauważyć, że problem porównania bezpieczeństwa adaptywnego i nieadaptywnego dostarcza w porównaniu z twierdzeniem 2 dodatkowego kłopotu. Przedstawione w dowodzie tego twierdzenia podejście rzeczywiście zawodzi[1] w przypadku przeciwników niedeterministycznych (z wejściem losowym), bo $\mathcal{S}_a$ uruchamiając kolejne symulacje (2., 4., 6.(a)) musiałby przekazywać własne wejście losowe (lub korzystać z jakiegoś dodatkowego generatora pseudolosowego), zaś symulator $\mathcal{S}_{na}$ może dawać całkiem inny wynik niż $\mathcal{A}_{na}$ na konkretnym wejściu (co nie przeczy odpowiedniej relacji uzyskiwanych wynikowych zespołów rozkładów prawdopodobieństwa protokołu przez $\mathcal{S}_{na}$ i $\mathcal{A}_{na}$):

Przykład 8 *Dla dowolnego protokołu dwuosobowego można rozważać prostego przeciwnika niedeterministycznego* $\mathcal{A}_p$, *który przed fazą komunikacji:*

1. $\mathcal{A}_p$ *dostaje, obok pozostałych danych, wejście losowe* $rand_0 \in \{0,1,2,3\}$ *(wybrane w sposób jednostajny).*
2. *Jeśli* $rand_0 = 0$, *przeciwnik nie korumpuje żadnego gracza i kończy.*
3. *Jeśli* $rand_0 \in \{1,2\}$, *to* $\mathcal{A}_p$ *korumpuje* P_i *i zwraca* in_i.
4. *Jeśli* $rand_0 = 3$, *przeciwnik korumpuje obydwu graczy i zwraca parę ich wejść.*

[1] W oczywisty sposób nie da się też przy jego pomocy dowodzić przypadku przeciwników deterministycznych w otwartych kanałach, ze względu na krok 2 przedstawionego tam schematu symulatora, bo trzeba by w nim dostarczyć przeciwnikowi rzeczywistemu komunikacji.

Oczywiście przeciwnika takiego da się w łatwy sposób symulować, niezależnie od dalszej części protokołu. Rozważmy jednak symulator $\mathcal{S}_{nap}$ (niedeterministyczny) przeciwnika nieadaptywnego $\mathcal{A}_{nap}$ (z punktu 4. rozważanego dowodu), który zachowuje się jak $\mathcal{A}_{nap}$, ale zamiast $rand_0$ kierując się wartością $3 - rand_0$. $\mathcal{S}_{nap}$ dobrze symuluje przeciwnika nieadaptywnego - to znaczy z równym prawdopodobieństwem korumpuje i zwraca to, co $\mathcal{A}_{nap}$. Jednocześnie uruchamiając $\mathcal{A}_{nap}$ i $\mathcal{S}_{nap}$ na tym samym wejściu losowym dostajemy inne wyniki.

Prowadzi to do sytuacji, w której $\mathcal{S}_{na}$ może zwrócić $error_1$ i trzeba by jakoś dodatkowo wielokrotnie uruchamiać ten symulator.

Rzeczywiście w przypadku otwartych kanałów komunikacyjnych to, czy przeciwnik jest deterministyczny, czy też nie, nie stanowi dodatkowego rozróżnienia niezależnie od adaptywności przeciwnika. Dla każdego protokołu i przeciwnika niedeterministycznego można bowiem rozważać podobny protokół i podobnego przeciwnika, ale deterministycznego. Wystarczy rozszerzyć odpowiednio wejście losowe jednego z graczy, aby zawierało wejście losowe przeciwnika i wymagać, by ten gracz jako pierwszy wysyłał do drugiego komunikat w postaci tej dodatkowej losowości specjalnie, by przeciwnik się o niej dowiedział.

Mimo tej różnicy w porównaniu do tajnych kanałów, z rozumowania użytego w dowodzie twierdzenia 2 można wysnuć pewne ogólne wnioski w postaci następującego twierdzenia:

Twierdzenie 3 *Jeśli dla każdego $X \in \{P_1, P_2\}$ protokół π oblicza $\{\emptyset, \{X\}\}$-bezpiecznie funkcję f, to π oblicza $2^{\{P_1,P_2\}}$-bezpiecznie funkcję f wobec przeciwników deterministycznych niezależnie od adaptywności przeciwnika, kanałów komunikacyjnych, modelu obliczeń i emulacji.*

Zauważmy, że w twierdzenie rzeczywiście jest w oczywisty sposób prawdziwe dla przeciwników nieadaptywnych. Jeśli bowiem symulator nieadaptywny korumpuje obydwu graczy, to symulacja przeciwnika rzeczywistego jest trywialna. Kolejną ważną rzeczą jest, że twierdzenie to jest prawdziwe niezależnie od tego, czy kanały komunikacyjne są tajne, czy otwarte, bo jak zobaczymy w poniższym dowodzie, nie odgrywa to żadnej roli.

Ograniczmy się więc do przeciwników adaptywnych i nie zakładajmy niczego o kanałach. Tak jak podczas dowodzenia twierdzenia 2, dowody dla emulacji dokładnej oraz statystycznej i obliczeniowej są nieco różne. Znacznie łatwiej jest dowieść twierdzenia, gdy zakłada się emulacje dokładną.

Dowód (bezpieczeństwo adaptywne, dokładna emulacja)
Rozważmy przeciwnika $\mathcal{A}$ ograniczonego strukturą $\mathcal{B} = 2^{\{P_1,P_2\}}$ (nie tracimy w ten sposób ogólności). Niech $\mathcal{A}_i$ dla $i \in \{1, 2\}$ będzie przeciwnikiem adaptywnym działającym według schematu:

1. $\mathcal{A}_i$ postępuje identycznie jak $\mathcal{A}$ i jeśli ten nie zdecyduje się na skorumpowanie żadnego z graczy, to kończy zwracając wynik $out_{\mathcal{A}_i} = out_{\mathcal{A}}$.

2. $\mathcal{A}$ zażądał skorumpowania gracza P_j.

   (a) Jeśli $i \neq j$, to $\mathcal{A}_i$ kończy działanie, zwracając *error*.
   Ważne jest, że $\mathcal{A}_i$ nie korumpuje wtedy gracza.

   (b) $i = j$, $\mathcal{A}_i$ korumpuje P_i.

3. $\mathcal{A}_i$ postępuje identycznie jak $\mathcal{A}$ i jeśli ten nie zdecyduje się na skorumpowanie drugiego gracza, to kończy zwracając wynik $out_{\mathcal{A}_i} = out_{\mathcal{A}}$.

4. $\mathcal{A}$ zażądał skorumpowania gracza P_{3-i}, $\mathcal{A}_i$ kończy, zwracając *continue*.

Przypuśćmy, że dla obydwu tych przeciwników istnieją symulatory $\mathcal{S}_i$, $i \in \{1, 2\}$. Należy teraz zauważyć, że ponieważ żaden z nich nie ma jakichkolwiek informacji przed skorumpowaniem gracza, to można żądać, by korumpowały graczy przed interakcją z graczem zaufanym. Wtedy można zasymulować działanie $\mathcal{A}$ w następujący sposób:

1. $\mathcal{S}$ dostaje parametr bezpieczeństwa k i dodatkowe wejście aux_0 od środowiska $\mathcal{Z}$.

2. $\mathcal{S}$ uruchamia symulator $\mathcal{S}_1$, dostarczając mu k i aux_0, i spełnia jego ewentualne żądanie skorumpowania gracza P_1. Jeśli $\mathcal{S}_1$ skorumpuje P_1, to $\mathcal{S}$ wysyła odpowiedni komunikat do gracza zaufanego i przekazuje uzyskaną wartość do $\mathcal{S}_1$. Niech $out_{\mathcal{S}_1}$ oznacza uzyskane wyjście.

   (a) Jeśli wynikiem powyższej symulacji nie jest *error* ani *continue*, to $\mathcal{S}$ kończy działanie, zwracając $out_{\mathcal{S}_1}$.

   (b) Jeśli $out_{\mathcal{S}_1} = continue$, to $i := 1$ i $\mathcal{S}$ wykonuje punkt 3.

   (c) ($out_{\mathcal{S}_1} = error$) Zaczynamy wielokrotną symulację $\mathcal{S}_2$:
   W tym momencie jeszcze żaden gracz nie został skorumpowany, bo $\mathcal{S}_1$ nie może skorumpować żadnego gracza i zwrócić error w przypadku dokładnej emulacji. Nie doszło też do interakcji z graczem zaufanym, bo bez korumpowania P_1, $\mathcal{S}_1$ nie może liczyć na zwrot funkcji.

      i. $\mathcal{S}$ uruchamia symulator $\mathcal{S}_2$, dostarczając mu k i aux_0, i spełnia jego ewentualne żądanie skorumpowania P_2.

      ii. Jeśli P_2 nie został skorumpowany, to $\mathcal{S}$ wraca do punktu i. (nie doszło do interakcji z graczem zaufanym).

      iii. Jeśli $out_{\mathcal{S}_2} \neq continue$, to $\mathcal{S}$ kończy, zwracając tą wartość.

      iv. ($out_{\mathcal{S}_2} = continue$) $i := 2$, $\mathcal{S}$ wykonuje punkt 3.
      P_2 został skorumpowany, więc (przy dokładnej emulacji) wynikiem działania symulatora $\mathcal{S}_2$ nie może być error.

3. W wyniku działania $\mathcal{S}_i$ otrzymaliśmy *continue*, gracz P_i został skorumpowany i jest już po interakcji z graczem zaufanym. $\mathcal{S}$ korumpuje drugiego gracza i wykonuje procedurę *sim_both* z dowodu twierdzenia 2:

   (a) Symuluj działanie $\mathcal{A}$ ze znanymi wejściami in_1, in_2, zalosowanymi wejściami $rand_1$, $rand_2$ oraz odpowiednimi aux_1 i aux_2 (jeśli potrzeba).

   (b) Jeśli gracz P_{3-i} nie został skorumpowany, wróć do (a).

   (c) Jeśli $\mathcal{A}$ skorumpował obydwu graczy, to $\mathcal{S}$ zwraca jego wynik.

Po pierwsze należy udowodnić, że jest to poprawna symulacja $\mathcal{A}$. Nie jest to jednak trudne, bo:

- jeśli $\mathcal{A}$ nie korumpuje żadnego gracza, symulacja jest dokonywana przez poprawny symulator $\mathcal{S}_1$,

- jeśli $\mathcal{A}$ korumpuje tylko jednego gracza, symulacja jest dokonywana przez odpowiedni symulator $\mathcal{S}_i$,

- jeśli $\mathcal{A}$ korumpuje obydwu, symulacja jest dokonywana przez procedurę *sim_both*.

Ważne jest przy tym, że w kroku 2.(c)ii. nie pozwala się na wyjście z pętli nawet jeśli $\mathcal{S}_2$ nie skorumpuje gracza P_2, ale odda wynik różny od *error* i *continue*. W przeciwnym przypadku wyniki działania $\mathcal{S}_i$, gdy nie korumpują graczy mogłyby się „źle" nakładać, nie dając w wyniku odpowiedniego zezpołu rozkładów.

Ponieważ powyższe zdarzenia są rozłączne, kończy to prosty dowód poprawności.

Pozostaje więc tylko udowodnić, że czas trwania symulacji jest odpowiedni. Tak jak w twierdzeniu 2 dowodzę bezpieczeństwa uniwersalnego (które implikuje dwa pozostałe).

Najpierw zauważmy, że jeśli $\mathcal{S}_1$ zwraca *error* z dodatnim prawdopodobieństwem, to oczekiwana liczba restartów symulacji w kroku 2.(c) wynosi:

$$P(\mathcal{S}_1 \text{ zwraca } error) \sum_{j=1}^{\infty} jP(\mathcal{S}_2 \text{ dokładnie } j \text{ razy nie korumpuje } P_2) =$$
$$= p \sum_{j=1}^{\infty} jp(1-p)^j = 1 - p < 1,$$

bo przy dokładnej emulacji $\mathcal{S}_1$ zwraca *error* z takim samym prawdopodobieństwem, co $\mathcal{S}_2$ korumpuje P_2.

Obliczenie liczby restartów procedury *sim_both* jest podobne:

$$P(\mathcal{S} \text{ dochodzi do kroku 3.}) \sum_{j=1}^{\infty} jP(sim\_both \text{ restartowane dokładnie } j \text{ razy}) =$$
$$(P(\mathcal{S} \text{ dochodzi do kroku 3.}, i = 1) + P(\mathcal{S} \text{ dochodzi do kroku 3.}, i = 2))$$
$$\sum_{j=1}^{\infty} jP(sim\_both \text{ restartowane dokładnie } j \text{ razy}) =$$
$$(P(\mathcal{S}_1 \text{ zwraca } continue) + P(\mathcal{S}_2 \text{ zwraca } continue))$$
$$\sum_{j=1}^{\infty} jP(sim\_both \text{ restartowane dokładnie } j \text{ razy}) =$$
$$(P(\mathcal{A} \text{ korumpuje } P_1 \text{ potem } P_2) + P(\mathcal{A} \text{ korumpuje } P_2 \text{ potem } P_1))$$
$$\sum_{j=1}^{\infty} jP(sim\_both \text{ restartowane dokładnie } j \text{ razy}) =$$
$$P(\mathcal{A} \text{ korumpuje obydwu graczy}) \sum_{j=1}^{\infty} jP(sim\_both \text{ restartowane dokładnie } j \text{ razy}) =$$
$$p \sum_{j=1}^{\infty} jp(1-p)^j = 1 - p < 1,$$

Udowodniliśmy więc, że symulacja jest poprawna i czas jej działania jest wielomianowy w stosunku do czasu działania przeciwnika rzeczywistego. □

Dowód ten nie działa jednak w przypadku emulacji statystycznej i obliczeniowej, bo kroki 2.(c) i 3. mogłyby się nie zakończyć lub trwać zbyt długo. Rzeczywiście na przykład symulator $\mathcal{S}_1$ może mylić się z pewnym zaniedbywalnym prawdopodobieństwem i doprowadzić do 2.(c), choć przeciwnik rzeczywisty nigdy nie korumpuje gracza P_2. Kolejną przeszkodą jest fakt, że nie można jak do tej pory zakładać, że $\mathcal{S}_1$ nie skorumpuje P_1 zwracając *error*. Należy więc zmienić nieco przedstawioną powyżej symulację, uzależniając te wrażliwe momenty od współczynnika k.

Dowiodę tu jedynie przypadku emulacji obliczeniowej, bo dowód dla emulacji statystycznej, jest bardzo podobny.

Dowód (bezpieczeństwo adaptywne, emulacja obliczeniowa)
Podobnie jak w poprzednim dowodzie rozważmy przeciwnika $\mathcal{A}$ ograniczonego strukturą $\mathcal{B} = 2^{\{P_1,P_2\}}$ i niech $\mathcal{A}_i$ dla $i \in \{1,2\}$ będą takimi jak tam przeciwnikami adaptywnymi, dla których istnieją symulatory $\mathcal{S}_i$ (tak jak wcześniej symulatory te korumpują tylko przed interakcją z zaufanym graczem). Konstruujemy symulator $\mathcal{A}$ w następujący sposób:

1. $\mathcal{S}$ dostaje parametr bezpieczeństwa k i dodatkowe wejście aux_0 od środowiska $\mathcal{Z}$.

2. $\mathcal{S}$ uruchamia symulator $\mathcal{S}_1$, dostarczając mu k i aux_0, i spełnia jego ewentualne żądanie skorumpowania gracza P_1. Jeśli $\mathcal{S}_1$ skorumpuje P_1, to $\mathcal{S}$ wysyła odpowiedni komunikat do gracza zaufanego i przekazuje uzyskaną wartość do $\mathcal{S}_1$. Niech $out_{\mathcal{S}_1}$ oznacza uzyskane wyjście.

(a) Jeśli wynikiem powyższej symulacji nie jest *error* ani *continue*, to $\mathcal{S}$ kończy działanie, zwracając $out_{\mathcal{S}_1}$.

(b) Jeśli $out_{\mathcal{S}_1} = continue$, to $i := 1$ i $\mathcal{S}$ wykonuje punkt 3.

(c) Jeśli $out_{\mathcal{S}_1} = error$, ale gracz P_1 został skorumpowany, $\mathcal{S}$ kończy zwracając $error_1$.

(d) ($out_{\mathcal{S}_1} = error$ i żaden z graczy nie jest skorumpowany) Zaczynamy wielokrotną symulację $\mathcal{S}_2$:

i. $r := 1$.

ii. $\mathcal{S}$ uruchamia symulator $\mathcal{S}_2$, dostarczając mu k i aux_0, i spełnia jego ewentualne żądanie skorumpowania P_2.

iii. Jeśli P_2 został skorumpowany i wynikiem działania nie jest *continue* ani *error*, to $\mathcal{S}$ zwraca tą wartość.

iv. Jeśli P_2 został skorumpowany i wynikiem działania jest *continue*, to $i := 2$ i $\mathcal{S}$ wykonuje punkt 3.

v. Jeśli P_2 został skorumpowany i wynikiem działania jest *error*, to $\mathcal{S}$ zwraca $error_2$.

vi. (P_2 nie został skorumpowany) $\mathcal{S}$ uruchamia symulator $\mathcal{S}_1$, dostarczając mu k i aux_0. Jeśli ten zażąda skorumpowania P_1 lub zwróci wartość różną od *error*, $\mathcal{S}$ nic nie robi. W przeciwnym przypadku $r := r + 1$.

vii. Jeśli $r > k$, $\mathcal{S}$ kończy, zwracając $loop_1$, w przeciwnym przypadku $\mathcal{S}$ idzie do punktu ii.

3. W wyniku działania $\mathcal{S}_i$ otrzymaliśmy *continue*, gracz P_i został skorumpowany i jest już po interakcji z graczem zaufanym. $\mathcal{S}$ korumpuje drugiego gracza i wykonuje zmodyfikowaną procedurę *sim_both*:

(a) $r := 1$.

(b) Symuluj działanie $\mathcal{A}$ ze znanymi wejściami in_1, in_2, zalosowanymi wejściami $rand_1$, $rand_2$ oraz odpowiednimi aux_1 i aux_2 (jeśli potrzeba).

(c) Jeśli $\mathcal{A}$ skorumpował obydwu graczy, to $\mathcal{S}$ zwraca jego wynik.

(d) (jeden z graczy nie został skorumpowany) $\mathcal{S}$ uruchamia symulator $\mathcal{S}_1$, dostarczając mu k i aux_0. Jeśli potrzeba, symuluje zachowanie gracza zaufanego (ma wszystkie potrzebne informacje). Podobnie uruchamia $\mathcal{S}_2$. Jeśli którykolwiek $\mathcal{S}_i$ skorumpuje P_i i zwróci *continue*, $r := r + 1$.

(e) Jeśli $r > k$, $\mathcal{S}$ kończy, zwracając $loop_2$, w przeciwnym przypadku $\mathcal{S}$ idzie do (a).

Podobnie jak poprzednio, zacznijmy od dowiedzenia poprawności powyższej symulacji. Dla przejrzystości wprowadźmy zapis $\mathcal{X} \rightsquigarrow P$ na oznaczenie, że $\mathcal{X}$ korumpuje gracza P ('$\not\rightsquigarrow$' odpowiednio) oraz $\mathcal{X} \to x$ na oznaczenie, że $\mathcal{X}$ zwraca w wyniku x ('$\not\to$' odpowiednio).

Na początku zauważmy, że $P(\mathcal{S} \to error_i)$ jest funkcją zaniedbywalną względem k. Rzeczywiście:

$$P(\mathcal{S} \to error_1) = P(\mathcal{S}_1 \rightsquigarrow P_1, \mathcal{S}_1 \to error)$$
$$P(\mathcal{S} \to error_2) \leq P(\mathcal{S}_2 \rightsquigarrow P_2, \mathcal{S}_2 \to error),$$

a prawdopobieństwa po prawej stronie są zaniedbywalne w k, bo $\mathcal{S}_i$ są symulatorami.

Następnym krokiem jest udowodnienie, że $P(\mathcal{S} \to loop_i)$ dla $i \in \{1,2\}$ są zaniedbywalne względem k. Zajmijmy się najpierw przypadkiem $loop_1$. Zauważmy:

$$
\begin{array}{l}
P(\mathcal{S} \to loop_1) \leq \\
P(\mathcal{S}_1 \not\rightsquigarrow P_1, \mathcal{S}_1 \to error) P(\mathcal{S}_2 \not\rightsquigarrow P_2)^k P(\mathcal{S}_1 \not\rightsquigarrow P_1, \mathcal{S}_1 \to error)^k \leq \\
\big(P(\mathcal{S}_1 \not\rightsquigarrow P_1, \mathcal{S}_1 \to error) P(\mathcal{S}_2 \not\rightsquigarrow P_2)\big)^k = \\
\Big(\big(P(\mathcal{A}_1 \rightsquigarrow P_2) + \epsilon_1\big)\big(P(\mathcal{A}_2 \not\rightsquigarrow P_2) + \epsilon_2\big)\Big)^k = \\
\Big(\big(P(\mathcal{A} \rightsquigarrow P_2) + \epsilon_1\big)\big(P(\mathcal{A} \not\rightsquigarrow P_2) + \epsilon_2\big)\Big)^k = \\
\Big(\big(P(\mathcal{A} \rightsquigarrow P_2) + \epsilon_1\big)\big(1 - P(\mathcal{A} \rightsquigarrow P_2) + \epsilon_2\big)\Big)^k = (*).
\end{array}
$$

Ponieważ $\mathcal{S}_i$ emulują obliczeniowo odpowiednich $\mathcal{A}_i$, to dla k wystarczająco dużych $|\epsilon_1|, |\epsilon_2| \leq \frac{1}{5}$ i dostajemy dalej:

$$|(*)| \sim |((p \pm \frac{1}{5})(1 - p \pm \frac{1}{5}))^k| \leq (p(1-p) + \frac{1}{5} + (\frac{1}{5})^2)^k < (p(1-p) + \frac{1}{4})^k \leq (\frac{1}{4} + \frac{1}{4})^k = \frac{1}{2^k}.$$

Rozpatrzmy teraz przypadek, gdy symulator zwraca $loop_2$:

$$
\begin{array}{l}
P(\mathcal{S} \to loop_2) \leq \\
P(\mathcal{S} \text{ dochodzi do kroku 3.}) P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)^k \\
\Big((P(\mathcal{S}_1 \rightsquigarrow P_1, \mathcal{S}_1 \to continue) + P(\mathcal{S}_2 \rightsquigarrow P_2, \mathcal{S}_2 \to continue))\Big)^k \leq \\
\Big(P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)(P(\mathcal{S}_1 \rightsquigarrow P_1, \mathcal{S}_1 \to continue) + P(\mathcal{S}_2 \rightsquigarrow P_2, \mathcal{S}_2 \to continue))\Big)^k = \\
\Big(P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)(P(\mathcal{A}_1 \to continue) + \epsilon_1 + P(\mathcal{A}_2 \to continue) + \epsilon_2)\Big)^k = \\
\Big(P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)(P(\mathcal{A}_1 \to continue) + P(\mathcal{A}_2 \to continue) + \epsilon)\Big)^k = \\
\Big(P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)(P(\mathcal{A} \rightsquigarrow P_1 \operatorname{potem} \mathcal{A} \rightsquigarrow P_2) + P(\mathcal{A} \rightsquigarrow P_2 \operatorname{potem} \mathcal{A} \rightsquigarrow P_1) + \epsilon)\Big)^k = \\
\Big(P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2)(1 - P(\mathcal{A} \not\rightsquigarrow P_1 \operatorname{lub} \mathcal{A} \not\rightsquigarrow P_2) + \epsilon)\Big)^k = (**)
\end{array}
$$

dla wystarczających k, $|\epsilon| < \frac{1}{4}$:

$$|(**)| \sim |(p(1 - p \pm \frac{1}{4}))^k| \leq (p(1 - p + \frac{1}{4}))^k \leq (p(1-p) + \frac{1}{4})^k \leq \frac{1}{2^k}.$$

Oznaczmy powyższe prawdopodobieństwa zaniedbywalne w k:

$$
\begin{array}{l}
\mathcal{E}_i(k, \overrightarrow{in}, aux) = \left\{ \begin{array}{ll} 1 & \mathcal{S} \to error_i \\ 0 & \text{w przeciwnym przypadku} \end{array} \right. , \\
\mathcal{L}_i(k, \overrightarrow{in}, aux) = \left\{ \begin{array}{ll} 1 & \mathcal{S} \to loop_i \\ 0 & \text{w przeciwnym przypadku} \end{array} \right. .
\end{array}
$$

Weźmy dowolny probablistyczny algortym wielomianowy D i ustalmy $c > 0$. Dla uzasadnienia poprawności obliczenia pozostaje pokazanie, że:

$$
\begin{array}{l}
|P(D(1^k, (\overrightarrow{in}, aux), \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k, \overrightarrow{in}, aux, \overrightarrow{rand})) = 1) - \\
P(D(1^k, (\overrightarrow{in}, aux), \mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(k, \overrightarrow{in}, aux) = 1)| < k^{-c}.
\end{array}
$$

Grupując razem odpowiednie prawdopodobieństwa warunkowe (stosuję dalej zapis '$\to$' i '$\rightsquigarrow$', tym razem na oznaczenie odpowiednich zmiennych losowych oraz *out* na oznaczenie słów,

które nie są znakami specjalnymi *error* i *continue*):

$$
\begin{aligned}
&|P(D(1^k,(\overrightarrow{in},aux),\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,\overrightarrow{in},aux,\overrightarrow{rand}))=1)-\\
&P(D(1^k,(\overrightarrow{in},aux),\mathrm{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(k,\overrightarrow{in},aux)=1)|\leq\\
&\textstyle\sum_{j=1}^{2}|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{E}_j(k,\overrightarrow{in},aux)=1\Big)|+\\
&\textstyle\sum_{j=1}^{2}|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{L}_j(k,\overrightarrow{in},aux)=1\Big)|+\\
&|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{A}\not\rightsquigarrow P_1,\mathcal{A}\not\rightsquigarrow P_2\Big)-\\
&P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{S}_1\not\rightsquigarrow P_1,\mathcal{S}_1\to out\Big)|+\\
&|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{A}\rightsquigarrow P_1,\mathcal{A}\not\rightsquigarrow P_2\Big)-\\
&P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{S}_1\rightsquigarrow P_1,\mathcal{S}_1\to out\Big)|+\\
&|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{A}\not\rightsquigarrow P_1,\mathcal{A}\rightsquigarrow P_2\Big)-\\
&P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{S}_1\not\rightsquigarrow P_1,\mathcal{S}_1\to error,\mathcal{S}_2\rightsquigarrow P_2,\mathcal{S}_2\to out,\mathcal{L}_1=0\Big)|+\\
&|P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{A}\rightsquigarrow P_1,\mathcal{A}\rightsquigarrow P_2\Big)-\\
&P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{S}_1\rightsquigarrow P_1,\mathcal{S}_1\to continue,\mathcal{L}_2=0\Big)-\\
&P\Big(D(1^k,(\overrightarrow{in},aux),I(k,\overrightarrow{in},aux))=1|\mathcal{S}_1\not\rightsquigarrow P_1,\mathcal{S}_1\to error,\mathcal{S}_2\rightsquigarrow P_2,\mathcal{S}_2\to continue,\mathcal{L}_2=0\Big))|.
\end{aligned}
$$

Ponieważ $\mathcal{S}_i$ dla $i\in\{1,2\}$ są symulatorami oraz $\mathcal{E}_i$ i $\mathcal{L}_i$ są zaniedbywalne, to dla odpowiednio dużych k, każdy składnik powyższej sumy jest mniejszy od $\frac{1}{8k^c}$.

Udowodnię na koniec, że czas trwania symulacji jest wielomianowy w stosunku do przeciwnika rzeczywistego. W tym celu trzeba pokazać, że liczba symulacji w punkcie 2.(d) oraz restartów zmodyfikowanej procedury *sim_both* jest odpowiednia.

Oczekiwana liczba symulacji w punkcie 2.(d)vi. potrzebnych, by r zwiększyło się o jeden:

$$1+\sum_{j=0}^{\infty} jP(\mathcal{S}_1\not\rightsquigarrow P_1,\mathcal{S}_1\to error)P(\mathcal{S}_1\rightsquigarrow P_1\,\mathrm{lub}\,\mathcal{S}_1\not\rightsquigarrow error)^j=1+\sum_{j=0}^{\infty} jp(1-p)^j=1+\frac{1-p}{p}=\frac{1}{p}$$

Założyliśmy, że $p>0$, bo przeciwnym przypadku $\mathcal{S}$ nie wchodzi do 2.(d). Oczekiwana liczba obrotów pętli 2.(d) jest więc mniejsza od $\frac{k}{p}$, a ponieważ prawdopodobieństwo wejścia do 2.(d) jest równe p, więc ostatecznie oczekiwana liczba obrotów pętli 2.(d) jest liniowa w k.

Podobne proste obliczenie prowadzi do ograniczenia oczekiwanej liczby restartów zmodyfikowanej procedury *sim_both*:

$$
\begin{aligned}
&P(\mathcal{S}\,\text{dochodzi do kroku 3.})\\
&k(1+\textstyle\sum_{j=0}^{\infty} j(P(\mathcal{S}_1\rightsquigarrow P_1,\mathcal{S}_1\to continue\,\mathrm{lub}\,\mathcal{S}_2\rightsquigarrow P_2,\mathcal{S}_2\to continue))\\
&(P(\mathcal{S}_1\not\rightsquigarrow P_1\,\mathrm{lub}\,\mathcal{S}_1\not\rightsquigarrow continue,\mathcal{S}_2\not\rightsquigarrow P_2\,\mathrm{lub}\,\mathcal{S}_2\not\rightsquigarrow continue))^k)=\\
&=pk(1+\textstyle\sum_{j=0}^{\infty} jp(1-p)^k).
\end{aligned}
$$

Pokazaliśmy więc, że symulacja ta jest poprawna i jej złożoność jest wielomianowa względem przeciwnika rzeczywistego i parametru k, co kończy dowód. □

Udowodnione twierdzenie pozwala na uproszczenie dowodów bezpieczeństwa protokoów dwustronnych względem przeciwników deterministycznych. Można zawsze ograniczyć się do udowodnienia, że dany protokół jest bezpieczny względem przeciwników, którzy mogą skorumpować tylko ustalonego gracza. Niestety wydaje się, że nie pomaga to rozstrzygnąć w żadną stronę problemu pozostawionego w [2].

Kolejną dość ważną rzeczą jest fakt, że od przeciwnika w otwartych kanałach można żądać, by jako część wyjścia dawał usłyszaną komunikację. Jest to naturalne, ale pociąga za sobą dość ważne konsekwencje. Wynika stąd mianowicie, że jeśli istniałby symulator dla przeciwnika (niezależnie od tego czy jest on adaptywny, czy nie), to musiałby umieć zwrócić komunikację między graczami. Rozważmy sytuację, gdy przeciwnik dostaje na wejściu, jako zbiór graczy, których może skorumpować, zbiór pusty. Wtedy fakt ten mówi, że aby protokół był bezpieczny przy oryginalnej definicji, komunikacja nie może być w żaden sposób - także co do liczby komunikatów i ich kolejności - zależna od wejść lub wyjść graczy. Ewentualnie komunikacja może być stała. Dzieje się tak dlatego, że dla symulatora takiego przeciwnika komunikacja jest całkowicie ukryta w mechanizmie gracza zaufanego. Stąd:

Stwierdzenie *Bezpieczeństwo protokołu dwustronnego przy otwartych kanałach wobec przeciwników deterministycznych jest równoważne bezpieczeństwu tego samego protokołu przy tajnych kanałach wobec przeciwników niedeterministycznych, gdzie losowość przeciwnika, uzyskana kosztem losowości graczy, jest uzależniona od komunikacji protokołu przy otwartych kanałach.*

Można bowiem, skoro komunikacja jest niezależna od wejść i wyjść, założyć że tylko część komunikacji jest widziana przez przeciwnika. Można ją podzielić na losowość płynącą z oryginalnej komunikacji - widzianą przez przeciwnika oraz rzeczywisty mechanizm wymiany informacji między graczami - tajny. Ciągle jednak ta losowość jest związana z wejściami losowymi graczy. Trzeba jednak zauważyć, że wejścia te można podzielić na część, z której może losowość czerpać przeciwnik oraz drugą, od pierwszej niezależną, która bieże udział w obliczeniu. Ostatnim krokiem jest spostrzeżenie, że taki podział wejść losowych graczy, jest równoważny, całkiem naturalnie, niedeterminizmowi przeciwnika przy jednoczesnym „zmniejszeniu" wejściowej losowości graczy.

W oparciu o powyższe rozważania można więc zawęzić sprawę porównania bezpieczeństwa adaptywnego i nieadaptywnego do przeciwników niedeterministycznych ograniczonych strukturą złożoną tylko ze zbioru pustego i jednego gracza przy kanałach zamkniętych. Wydaje się, że w takim przypadku bezpieczeństwo adaptywne i nieadaptywne powinno być równoważne, bo każdego przeciwnika adaptywnego $\mathcal{A}_a$ można symulować w następujący sposób:

1. Symulator $\mathcal{S}_a$ dostaje parametr bezpieczeństwa k, wejście losowe $rand_0$ i strukturę $\mathcal{B} = \{\emptyset, \{P_1\}\}$, oraz dodatkowe wejście aux_0 od środowiska $\mathcal{Z}$.
2. $\mathcal{S}_a$ przekazuje przeciwnikowi $\mathcal{A}_a$ k, $rand_0$ i $\mathcal{B}$, oraz aux_0 (zachowując się jak środowisko).
3. Jeśli $\mathcal{A}_a$ skończy, nie korumpując gracza P_1, to $\mathcal{S}$ kończy z wynikiem takim jak $\mathcal{A}_a$.
4. Jeśli $\mathcal{A}_a$ zażąda skorumpowania P_1, to $\mathcal{S}_a$ korumpuje P_1 i uruchamia symulator następującego przeciwnika nieadaptywnego, oddając wyjście tego symulatora:
   (a) Przeciwnik $\mathcal{A}_{na}$, dostając na wejściu k, $rand_0$ oraz aux_0, aux_1 i in_1, korumpuje gracza P_1.
   (b) Korzystając z $rand_0$ i generatora pseudolosowego (odpowiedniego dla dziedziny wejść losowych przeciwnika), uruchamia przeciwnika $\mathcal{A}_a$ z odpowiednimi parametrami i losowością tak otrzymaną, dopóki $\mathcal{A}_a$ nie zdecyduje się na skorumpowanie P_1.
   Założenie możliwości wykorzystania przez przeciwnika generatora liczb pseudolosowych zdaje się konieczne w przypadku niedeterministycznym, jednak założenie, że przeciwdziedziną tego generatora jest dziedzina wejść losowych przeciwnika jest nieco obciążające.

(c) $\mathcal{A}_{na}$ powiela wszystkie kroki przeciwnika $\mathcal{A}_a$ i razem z nim kończy, zwracając jego wynik.

Oczywiście nie jest to w żadnym wypadku ścisły dowód, a jedynie odzwierciedlenie pewnej intuicji. Na podstawie tych rozważań skłaniam się jednak do tezy, że dla protokołów dwustronnych w modelu otwartych kanałów komunikacyjnych, bezpieczeństwo adaptywne jest równoważne nieadaptywnemu.

Rozdział 5

Podsumowanie

Pierwotnym celem tej pracy było obok przedstawienia definicji bezpieczeństwa według pracy Canettiego [1], rozwiązanie pozostawionego bez odpowiedzi w publikacji [2] pytania, czy istnieje protokól dla dwóch graczy, który przy otwartych kanałach komunikacyjnych jest bezpieczny nieadaptywnie i niebezpieczny adaptywnie. W kolejnych rozdziałach pokazuję definicję bezpieczeństwa i jej zastosowanie, pomagając sobie przykładami.

Pytanie okazało się jednak trudne i nie umiem podać na nie odpowiedzi. Przypadek ten bowiem zdaje się mocno wiązać z niedeterminizmem przeciwnika, który w pracy [2] także został opuszczony. Przedstawiłem jednak obok zaczerpniętych dowodów i przykładów, własne wnioski i pewne ograniczenia dla rozważań nad powyższym pytaniem. Postawiłem i udowodniłem twierdzenie, że - niezależnie od tajności kanałów - jeśli protokół dwustronny oblicza funkcję bezpiecznie względem przeciwników, którzy mogą skorumpować tylko jednego, ustalonego gracza, to protokól ten jest $\mathcal{B}$-bezpieczny dla dowolnej struktury $\mathcal{B}$. Dowiodłem w ten sposób, że pytanie można zawęzić do przeciwników korumpujących tylko jednego, ustalonego gracza i opisałem intuicję, która skłania mnie do opowiedzenia się po stronie równoważności bezpieczeństwa nieadaptywnego i adaptywnego w tym przypadku.

W podrozdziale 4.1.1 zamieściłem też, poboczne dla celu tej pracy, rozważania na temat bezpieczeństwa protokołu wobec przeciwników początkowo adaptywnych. Przedstawiam tam protokół tajnie obliczający pewną funkcję nieadaptywnie, który nie jest bezpieczny wobec pasywnych przeciwników początkowo adaptywnych.

Dodatek A

Zespoły rozkładów prawdopodobieństwa

Podaje tutaj podstawowe definicje i twierdzenia niezbędne do zrozumienia zagadnienia bezpieczeństwa w świetle definicji Canettiego. Część z nich, przez zaadaptowanie do podejścia przedstawionego w tej pracy, traci nieco na ogólności. Można je znaleźć w większości pozycji podanych w bibliografii. Zakłada się przy tym znajomość podstaw rachunku prawdopodobieństwa.

Definicja 3 *Zespołem (dyskretnych i skończonych[1]) rozkładów prawdopodobieństwa nazywamy dowolny indeksowany zbiór rozkładów prawdopodobieństwa* $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *niezerowych jedynie na zbiorze skończonym, gdzie* S *jest pewną dziedziną (zwykle* $S = \{0,1\}^*$*).*

Definicja 4 *Mówimy, że zespoły rozkładów prawdopodobieństwa* $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *i* $\{Y_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *są równo rozłożone i piszemy* $X =_d Y$*, jeśli:*

$$\forall_{k\in\mathbb{N},\alpha\in S} X_{k,\alpha} = Y_{k,\alpha}.$$

Definicja 5 *Funkcję* $\delta : \mathbb{N} \to \mathbb{R}$ *nazywamy zaniedbywalną, jeśli dla każdego wielomianu* p *o wartościach w* $\mathbb{R}_+$*:*

$$\exists_{N\in\mathbb{N}}\forall_{\mathbb{N}\ni n>N} |\delta(n)| < \frac{1}{p(n)}.$$

Definicja 6 *Funkcję* $\delta : \mathbb{N} \to \mathbb{R}$ *nazywamy ograniczeniem dystansu statystycznego zespołów rozkładów prawdopodobieństw* $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *i* $\{Y_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$*, jeśli:*

$$\exists_{K\in\mathbb{N}}\forall_{\mathbb{N}\ni k>K}\forall_{\alpha\in S} SD(X_{k,\alpha}, Y_{k,\alpha}) < \delta(k),$$

gdzie dystans statystyczny dwóch rozkładów określa się wzorem:

$$SD(X,Y) = \frac{1}{2}\sum_{\phi\in\Phi} |P(X=\phi) - P(Y=\phi)|$$

(dla Φ *będącego dziedziną rozkładów* X*,* Y*).*

Definicja 7 *Mówimy, że zespoły rozkładów prawdopodobieństwa* $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *i* $\{Y_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ *są statystycznie nierozróżnialne i piszemy* $X =_s Y$*, jeśli istnieje dla nich ograniczenie dystansu statystycznego będące funkcją zaniedbywalną* k*.*

[1] Cała praca opiera się na definicji z takimi założeniami.

Definicja 8 *Funkcję $\delta : \mathbb{N} \to \mathbb{R}$ nazywamy ograniczeniem dystansu obliczeniowego zespołów rozkładów prawdopodobieństw $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ i $\{Y_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$, jeśli dla dowolnego probablistycznego algorytmu D o wielomianowej złożoności ze względu na pierwszy parametr:*

$$\exists_{K\in\mathbb{N}}\forall_{\mathbb{N}\ni k>K}\forall_{\alpha\in S}|P(D(1^k,\alpha,x)=1)-P(D(1^k,\alpha,y)=1)|<\delta(k),$$

gdzie x i y są wzięte zgodnie z rozkładami $X_{k,\alpha}$ i $Y_{k,\alpha}$, a prawdopodobieństwo jest liczone po wyborach x i y oraz losowych wyborach algorytmu D.

Definicja 9 *Mówimy, że zespoły rozkładów prawdopodobieństwa $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ i $\{Y_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ są obliczeniowo nierozróżnialne i piszemy $X =_c Y$, jeśli istnieje dla nich ograniczenie dystansu statystycznego będące funkcją zaniedbywalną k.*

Twierdzenie 4 *Dla dowolnych dwóch zespołów rozkładów prawdopodobieństwa X i Y zachodzą implikacje: $X =_d Y \Rightarrow X =_s Y$ oraz $X =_s Y \Rightarrow X =_c Y$, ale nie można ich zastąpić równoważnościami*[2]*.*

Twierdzenie 5 *Określone na zbiorze zespołów rozkładów prawdopodobieństwa $\{X_{k,\alpha}\}_{k\in\mathbb{N},\alpha\in S}$ dla ustalonego S takich, że nośniki każdych dwóch rozkładów $\{X_{k',\alpha'}\}$ i $\{Y_{k',\alpha'}\}$ są identyczne dla ustalonych $k \in \mathbb{N}$ i $\alpha \in S$, relacje $=_d$, $=_s$ i $=_c$ są relacjami równoważności.*

[2]Dowód $\neg(X =_s Y \Leftrightarrow X =_c Y)$ dostępny w [5].

Bibliografia

[1] Ran Canetti: *Security and Composition of Multi-party Cryptographic Protocols*, 2000, Journal of Cryptology, Vol. 13, No. 1

[2] Ran Canetti, Ivan Damgaard, Stefan Dziembowski, Yuval Ishai, Tal Malkin: *On adaptive vs. non-adaptive security of multiparty protocols*, 2001, Lecture Notes in Computer Science, Vol. 2045, str. 262+

[3] Ran Canetti, Uri Feige, Oded Goldreich, Moni Naor: *Adaptively Secure Computation* 1996, 28th Symposium on Theory of Computing (STOC), ACM

[4] Oded Goldreich: *A Uniform Complexity Treatment of Encryption and Zero-Knowledge*, 1993, Journal of Cryptology, Vol. 6, No. 1, Springer Verlag, str. 21-53

[5] Oded Goldreich: *Foundation of Cryptography*, fragmenty książki dostępne pod adresem http://www.wisdom.weizmann.ac.il/~oded/frag.html

[6] Silvio Micali, Philip Rogaway: *Secure Computation* 1992, praca nieopublikowana