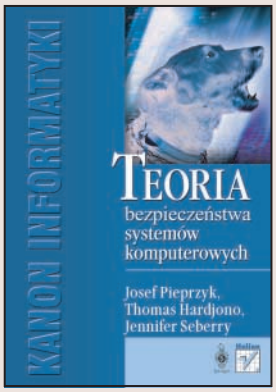
Teoria bezpieczeństwa systemów komputerowych



Autorzy: Josef Pieprzyk, Thomas Hardjono, Jennifer Seberry
Tłumaczenie: Andrzej Grażyński, Tomasz Żmijewski
ISBN: 83-7361-678-0
Tytuł oryginału: Fundamentals of Computer Security
Format: B5, stron: 600

Poznaj zasady tworzenia mechanizmów zabezpieczeń

- Matematyka i kryptografia
- Szyfrowanie informacji
- Bezpieczeństwo transakcji i danych

Tworzenie zabezpieczeń systemów komputerowych to ciągły wyścig. Rozwój technik hakerskich wymusza opracowywanie coraz doskonalszych mechanizmów zabezpieczających komputery, sieci, dane i transakcje dokonywane w sieci. Proste zabezpieczenia oparte na hasłach to już przeżytek. Współczesne podejście do zabezpieczania danych opiera się na zaawansowanych mechanizmach matematycznych i kryptograficznych. Ich znajomość jest niezbędna do zrozumienia tego, jak działają systemy zabezpieczające, poprawnego konfigurowania istniejących systemów i tworzenia własnych.

„Teoria zabezpieczeń systemów komputerowych" to przegląd podstawowych pojęć związanych z bezpieczeństwem komputera i sieci komputerowej. Książka została napisana w oparciu o wykłady dotyczące zabezpieczeń systemów komputerowych, prowadzone na Uniwersytecie w Wollongong w Australii. Przedstawia najnowsze osiągnięcia kryptografii oraz matematyczne podstawy zabezpieczania danych. Opisuje również metody wykrywania intruzów w sieciach oraz mechanizmy kontroli dostępu.

- Elementy teorii liczb i teorii informacji
- Szyfrowanie kluczem prywatnym
- Stosowanie klucza prywatnego
- Algorytmy wykorzystujące liczby pseudolosowe i mieszanie
- Podpis cyfrowy i autoryzacja dostępu
- Identyfikacja
- Wykrywanie intruzów
- Transakcje elektroniczne
- Zabezpieczanie baz danych

Dowiedz się, jak działają współczesne mechanizmy zabezpieczania danych

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

4.

Kryptosystemy z kluczem publicznym 159

5.

Pseudolosowość 201

6.

Mieszanie 223

12.

13.

14.

17.

18.

13
Identyfikacja

Identyfikacja jest zazwyczaj jednym z pierwszych kroków w kierunku zapewnienia ochrony zasobów systemu komputerowego przed nieautoryzowanym dostępem. Istotą identyfikacji jest ścisłe i niezawodne kontrolowanie, kto próbuje uzyskać dostęp do wspomnianych zasobów i w jaki sposób chce je wykorzystywać.

13.1. Podstawowe techniki identyfikacyjne

Identyfikacja osoby, hosta, terminala inteligentnego, programu, systemu itp. może być postrzegana jako protokół komunikacji między dwoma uczestnikami: udowadniającym (ang. *prover*), oznaczanym najczęściej przez *P* i zwykle kojarzonym z imieniem Peggy, oraz weryfikatorem, oznaczanym przez *V* i zwykle kojarzonym z imieniem Wiktor. Zadaniem Peggy jest przedstawienie się Wiktorowi w taki sposób, by przekonać go iż ma do czynienia właśnie z nią, nie z nikim innym.

Działanie każdego protokołu identyfikacyjnego może zakończyć się niepowodzeniem na dwa sposoby: po pierwsze, podszywającemu się pod Peggy intruzowi — nazwijmy go Oskarem — może udać się oszukać Wiktora, który uwierzy, iż komunikuje się z Peggy; sytuację taką nazywamy *fałszywą akceptacją* (ang. *false acceptance*). I odwrotnie: Wiktor, mimo jak najszczerszych intencji Peggy logującej się po drugiej stronie łącza, może nie uwierzyć w jej autentyczność — tę sytuację nazywamy *fałszywym odrzuceniem* (ang. *false rejection*). Wartości prawdopodobieństw wystąpienia tych sytuacji, oznaczane (odpowiednio) przez P_{f_a} i P_{f_r}, stanowią cechę charakterystyczną protokołu identyfikacyjnego.

Rozpatrzmy dwie skrajne, wyidealizowane postaci protokołu identyfikacji. W pierwszej z nich Wiktor zapytuje swego partnera o imię i po uzyskaniu odpowiedzi „Peggy" bezkrytyczne zakłada jej prawdziwość; prawdopodobieństwo fałszywej akceptacji jest tu stuprocentowe ($P_{f_a} = 1$), zaś fałszywe odrzucenie jest niemożliwe ($P_{f_r} = 0$). W drugim, całkowicie odwrotnym przypadku, podejrzliwy Wiktor zakłada stuprocentowe oszustwo interlokutora, nie dając prawdziwej Peggy żadnych szans na konwersację; zabezpiecza się w ten sposób przed podstępem ze strony intruza ($P_{f_a} = 0$), odrzucając jednak wszelkie próby nawiązania konwersacji ($P_{f_r} = 1$). Jest zrozumiałe, że w przypadku „porządnego" protokołu identyfikacyjnego obydwie wartości P_{f_a} i P_{f_r} — jako prawdopodobieństwa zdarzeń niepożądanych — powinny być jak najmniejsze.

Identyfikacja *podmiotu* (ang. *entity*) — którym może być osoba, host, terminal inteligentny, program itp. — może być rozpatrywana jako weryfikacja tegoż podmiotu w kategoriach:

- bycia,
- posiadania,
- wiedzy.

Pierwsza z wymienionych kategorii nazywana jest tradycyjnie *identyfikacją użytkownika* (ang. *user identification*), bowiem w systemie komputerowym to właśnie użytkownik cechuje się pewnymi niepowtarzalnymi, osobistymi właściwościami — jak odcisk palca, charakter pisma (podpis ręczny), barwa głosu, struktura siatkówki i tęczówki oka itp. Wszelkiego rodzaju informacja cyfrowa może być natomiast z łatwością zwielokrotniana (kopiowana), wskutek czego nie sposób tu odróżnić kopii od oryginału.

Kategoria *posiadania* odnosi się do pewnych unikalnych przedmiotów, jak np. karta magnetyczna lub chipowa, chroniona tajnym kodem dostępu. Przedmiot taki jest świadectwem autentyczności posiadacza, ten ostatni z założenia nie powinien go więc utracić ani udostępnić nikomu innemu.

Trzecia wreszcie kategoria bazuje na pewnej tajnej informacji, która znana jest jedynie uprawnionemu podmiotowi. Najbardziej znanym przykładem takiej informacji są wszelkiego rodzaju hasła. Z jednej strony ich tajność jest warunkiem *sine qua non* wiarygodności identyfikacji — hasło, które przestało być tajne, może być wykorzystane przez każdego intruza — z drugiej natomiast uprawniony podmiot, który hasła zapomniał, nie może zaświadczyć o swej autentyczności.

Obydwie ostatnie kategorie — posiadania i wiedzy — posiadają cechę wspólną, którą jest wykorzystywanie sekretnej informacji. Różnią się natomiast sposobem przechowywania tejże informacji: w pierwszym przypadku informacja zapisana jest wewnątrz pewnego gadżetu, czyli niejako *na zewnątrz identyfikującego się podmiotu*; w drugim przypadku informacja (w postaci określonej wiedzy) tkwi organicznie *wewnątrz podmiotu* — na przykład w umyśle użytkownika czy w numerze seryjnym BIOS-u komputera.

13.2. Identyfikacja użytkowników

Odciski palców przyjęto uważać za niepowtarzalną cechę osobniczą — niepowtarzalną tak wysoce, że zaakceptowaną jako niepodważalny dowód w postępowaniu sądowym. W systemach identyfikacji odcisk palca traktowany jest jako wzorzec wypukłości i wklęsłości, a dokładniej — jako złożenie (superpozycja) pewnych elementarnych, skatalogowanych szablonów linii papilarnych, zwanych w języku angielskim *minutiae*.

Dostępne obecnie automaty do identyfikacji linii papilarnych (skrótowo oznaczane jako AFIMs, od ang. *automated fingerprint identification machines*) cechują się prawdopodobieństwem błędnej akceptacji lub błędnego odrzucenia rzędu 10^{-3} lub mniejszym. Zapamiętanie pojedynczego wzorca odcisku palca wymaga około 1 kilobajta pamięci i trwa mniej niż 10 sekund; proces identyfikacji przeprowadzany jest w ciągu kilku sekund. Maszyny takie są jednak jeszcze dość drogie (ponad 1000 USD) i wciąż zbyt zawodne (10^{-3}, czyli statystycznie jedna fałszywa akceptacja lub nieuzasadnione odrzucenie na każde 1000 prób to ciągle zbyt dużo), a ich zastosowanie do identyfikacji użytkowników systemów komputerowych jest jak na razie raczej ograniczone.

Na podobnej zasadzie wykorzystać można niepowtarzalność struktury *siatkówki* i *tęczówki* oka. Urządzenia skanujące przetwarzają strukturę siatkówki (tęczówki) na superpozycję cyfrowych szablonów. Zapamiętanie jednego szablonu wymaga przeciętnie 40 bajtów; skanowanie zajmuje około

30 sekund, a proces identyfikacji mniej niż 2 sekundy. Prawdopodobieństwo błędnej akceptacji (błędnego odrzucenia) plasuje się w granicach 10^{-6}, jednakże zbyt wysoka cena niezbędnego sprzętu nie predestynuje obecnie tej techniki do masowych zastosowań w systemach komputerowych.

Spośród innych niepowtarzalnych cech biometrycznych, możliwych do zastosowania na potrzeby identyfikacji, wymienić należy *topografię dłoni* (zapamiętanie jednego wzorca w postaci zakodowanej wymaga jedynie 10 bajtów) oraz *geometrię twarzy* — ta ostatnia cecha jest o tyle atrakcyjna, iż nie wiąże się z niezbyt przyjemnymi zabiegami towarzyszącymi chociażby skanowaniu gałki ocznej; może być ponadto stosowana bez przeszkód na skalę masową, na przykład w celu wyszukiwania (znanych) terrorystów na dworcach lotniczych. Prawdopodobieństwo błędnej identyfikacji kształtuje się tu w okolicach 10^{-4}.

Podpis odręczny stosowany jest od niepamiętnych czasów jako metoda uwierzytelniania dokumentów papierowych. Mimo iż dwa podpisy złożone przez tę samą osobę nigdy nie będą identyczne, to jednak posiadać będą pewne cechy wspólne, wynikające z nawykowego charakteru samego pisania: nacisk i kierunek prowadzenia pióra, styl pisma, pochylenie liter, szybkość i przyspieszenie itp. Zapamiętanie tych cech dla danego podpisu wymaga ok. 1 kilobajta pamięci, a w celu ustalenia wspomnianych właściwości dla danej osoby musi ona złożyć 5-8 podpisów testowych. Weryfikacja tych cech w podpisie identyfikującej się osoby zajmuje mniej niż sekundę. Niezwykle atrakcyjną cechą tej techniki identyfikacji jest łatwość jej implementacji w systemie komputerowym — wystarczy mysz spełniająca funkcję pióra, bez żadnego dodatkowego wyposażenia.

Weryfikacja *głosu* identyfikującej się osoby należy do najbardziej zawodnych pod względem prawdopodobieństwa błędu. Jest jednak najprostsza w implementacji, może być bowiem łatwo przeprowadzona w sposób zdalny — przez telefon.

Osobniczą cechą użytkownika piszącego na klawiaturze komputera jest też *rytm i tempo naciskania oraz zwalniania poszczególnych klawiszy*. Jest to najbardziej „przyjazna dla komputera" metoda identyfikacji użytkowników; niestety, jak wykazały liczne eksperymenty, prawdopodobieństwo mylnego zaakceptowania lub mylnego odrzucenia użytkownika jest zbyt duże, by można było metodę tę zastosować na potrzeby niezawodnej identyfikacji. W celu zmniejszenia prawdopodobieństwa błędu należałoby wyposażyć klawiaturę w dodatkowe czujniki, rejestrujące inne parametry „stukania w klawisze" — siłę nacisku, szybkość naciskania i zwalniania klawisza itp.

Dla kompletności należy jeszcze wspomnieć o metodach identyfikacji bazujących na *kodzie DNA* człowieka. Teoretycznie prawdopodobieństwo błędu tej metody równe jest zeru — z dokładnością jednakże do faktu, iż kod DNA może być identyczny u bliźniaków. W praktyce zastosowanie tej metody ograniczone jest do specjalistycznych laboratoriów, jest ona bowiem czasochłonna, a ponadto wymaga dysponowania przykładowymi próbkami kodu genetycznego identyfikowanej osoby; rozmaite — w tym również prawne — aspekty zagadnienia wykluczają kod DNA jako powszechny środek identyfikacji użytkownika w systemie komputerowym.

Każda *biometryczna metoda* identyfikacji użytkownika, niezależnie od jej cech specyficznych, podatna jest na atak przez powtórzenie (ang. *replay attack*). Przykładowo głos danej osoby może zostać nagrany, a następnie odtworzony przez intruza za pośrednictwem telefonu (chyba że proces identyfikacji wymagać będzie od użytkownika powtarzania losowo wybieranych kwestii, czemu z oczywistych przyczyn intruz z magnetofonem sprostać nie zdoła).

13.3. Hasła

Najbardziej popularną techniką weryfikacji bazującej na wiedzy użytkownika jest system *haseł lub numerów* identyfikacji osobistej (ang. PIN — *Personal Identification Number*). Konkretna postać hasła może być wymuszona przez specyfikę urządzenia, z którego korzysta użytkownik: przykładowo

klawiatura bankomatu, zawierająca jedynie klawisze cyfr, wymusza stricte numeryczną postać PIN. Ponadto, jako że każde hasło musi być przez użytkownika *zapamiętane*, nie może być zbyt długie; w praktyce długości haseł zawierają się najczęściej między czterema a dziewięcioma znakami.

W przypadku hasła *n*-znakowego, wykorzystującego 26 liter (małe i wielkie litery nie są rozróżniane) prawdopodobieństwo *odgadnięcia* hasła wynosi 26^{-n} (pod warunkiem wszakże, iż każde z 26^n słów może wystąpić z jednakowym prawdopodobieństwem). Gdy małe litery odróżniane są od swych „wielkich" odpowiedników, prawdopodobieństwo odgadnięcia hasła gwałtownie spada do wartości 52^{-n}. Stanie się ono jeszcze mniejsze, jeśli oprócz liter dopuścimy cyfry i (lub) inne znaki „drukowalne" w rodzaju $, %, <, {, ; , " itp.

Pytanie o hasło pojawia się (zazwyczaj) każdorazowo, gdy użytkownik (Peggy) zamierza zalogować się do hosta *V*. Peggy pamięta swoje hasło, a host *V* dysponuje plikiem, w którym zapamiętane są hasła wszystkich (znanych mu) użytkowników. Po wprowadzeniu przez Peggy pary (nazwa, hasło) komputer porównuje tę parę z odpowiednim zapisem we wspomnianym pliku i zależnie od wyniku tego porównania akceptuje albo odrzuca logowanie. Zwróćmy uwagę, iż plik haseł musi być chroniony przed dostępem nie tylko ze strony „zwykłych" użytkowników, lecz także ze strony *superużytkownika*. Dla większego bezpieczeństwa zawartość pliku haseł jest szyfrowana lub haszowana; wprowadzone hasło jest wówczas najpierw poddawane takiemu samemu szyfrowaniu lub haszowaniu, po czym wynik tej operacji porównywany jest z zaszyfrowanym (haszowanym) wzorcem. Haszowanie ma w tym przypadku tę przewagę nad szyfrowaniem, że nie jest uzależnione od żadnego klucza, który intruz mógłby teoretycznie odgadnąć.

Bezpieczeństwo określonego hasła maleje systematycznie wraz z jego kolejnym użyciem, w związku z czym często ogranicza się jego *ważność* do określonego przedziału czasowego, typowo wahającego się między 3 tygodniami a 3 miesiącami. W przypadku ekstremalnym ważność hasła ograniczona jest tylko do *jednokrotnego użycia* (ang. *one-time password*). Implementacja systemu haseł jednokrotnych polega najczęściej na wygenerowaniu — dla każdego użytkownika — listy haseł i określeniu kolejności, w jakiej poszczególne pozycje tej listy muszą być używane przy kolejnych próbach logowania. Rozwiązanie to jest jednak uciążliwe dla użytkownika, który zmuszony jest *dokładnie* zapamiętać wiele haseł jednocześnie. Można tę uciążliwość zmniejszyć, wybierając zestaw n+1 haseł powiązanych jednokierunkową funkcją f zgodnie z regułą $p_i = f(p_{i-1})$ dla $i = 1, \ldots, n$. Użytkownik wyposażony zostaje w specjalne urządzenie (ang. *token*), umożliwiające szybkie otrzymanie p_i dla zadanego i; tym samym identyfikacja oparta na *wiedzy* użytkownika zmienia się automatycznie w identyfikację opartą na *posiadaniu* przez użytkownika wspomnianego urządzenia. Kolejność używania poszczególnych haseł jest odwrotna w stosunku do ich numeracji[1] — pierwszym użytym hasłem jest p_n, ostatnim p_0.

13.3.1. Atak na system haseł

Z każdym hasłem związane jest organicznie niebezpieczeństwo jego *skompromitowania*, czyli po prostu poznania przez osobę nieuprawnioną. Niebezpieczeństwo to rozpoczyna się już w momencie wpisywania hasła — intruz może po prostu zajrzeć „przez ramię" Peggy, a być może wystarczy mu tylko obserwacja ruchów jej ramion! Kolejną okazją do skompromitowania hasła jest przesyłanie go *niezabezpieczonym* kanałem od terminala do hosta; niebezpieczeństwo kompromitacji wzrasta

[1] Wynika to bezpośrednio z jednokierunkowego charakteru funkcji f. Intruz, któremu uda się przechwycić hasło p_i i poznać funkcję f, może co najwyżej „obliczyć" hasła p_{i+1}, p_{i+2} …, do niczego już nieprzydatne. Ze względu na jednokierunkowy charakter funkcji f nie jest możliwe obliczenie następnych poprawnych haseł $p_{i-1} = f^{-1}(p_i)$, $p_{i-2} = f^{-1}(p_{i-1})$ itd. z oczywistego powodu — funkcja f^{-1} po prostu nie istnieje — *przyp. tłum.*

znacznie, jeśli kanałem tym jest internet. Oczywiście nie jest niebezpieczne skompromitowanie hasła jednokrotnego — co jest silnym argumentem przemawiającym za używaniem tego typu haseł.

Niezwykle istotną sprawą jest wybór postaci samego hasła. Idealne hasło powinno być tworzone na drodze wyboru losowego; sęk jednak w tym, że takie losowe hasła, pozbawione jakiegokolwiek odniesienia do świata rzeczywistego, stosunkowo trudno się zapamiętuje. Użytkownicy przejawiają więc skłonność do tworzenia haseł o pewnym stopniu mnemoniczności, a te wrażliwe są niezmiennie na ataki typu *brute force*, czyli intensywne próby dopasowywania znanych wzorców za pomocą przeszukiwania wyczerpującego. Potencjalny intruz Oskar, znając upodobania Peggy — tytuły jej ulubionych książek, filmów, postaci, aktorów, piosenek itp. — może obszar tego przeszukiwania znacznie ograniczyć, wypróbowując imiona jej znajomych, jej ulubionych zwierzątek, nazwiska (pseudonimy) znanych gwiazd rocka, sportowców itp. Gdy to zawiedzie, Oskar może wypróbować nazwę hosta Peggy, numer jej telefonu, numer ewidencyjny z kartoteki pracowników, numer paszportu, elementy adresu, datę urodzenia itp. Gdy nie powiedzie się próba trafienia na hasło w czystej postaci, można następnie wypróbowywać kombinację wymienionych rzeczowników z przypadkowymi ciągami cyfr lub liter, dopełnieniem zerami, z zamianą małych liter na duże (i vice versa) z przestawianiem liter, z pisaniem wstecz itp.; tego typu kombinacje określane są ogólną nazwą *ataków słownikowych* (ang. *dictionary attacks*) — współczesne komputery potrafią (często skutecznie) przypuszczać takie ataki na bazie słowników zawierających kilkaset tysięcy słów. Ataki takie można skutecznie paraliżować, ograniczając liczbę nieudanych (z rzędu) prób logowania oraz wymuszając przerwy (od kilkunastu sekund do minuty) między kolejnymi próbami logowania. Nie zda się to jednak na nic, jeśli Oskar wejdzie w posiadanie pliku z zaszyfrowanymi hasłami.

Peggy może się udać pogodzić dwa z natury sprzeczne wymagania — losowość hasła, utrudniająca jego odgadnięcie, kontra jego mnemoniczność, ułatwiająca zapamiętanie — jeśli będzie ona przestrzegać trzech następujących reguł:

- Jeżeli w haśle dopuszcza się maksymalnie *n* znaków, to limit ten należy w całości wykorzystać; dłuższe hasło to większy kłopot z jego odgadnięciem.
- Hasło powinno zawierać znaki specjalne, nie wchodzące w skład słów w języku potocznym — `$`, `%`, `&`, `@`, `{`, `[`, `(` itp. — a małe litery powinny być przemieszane z wielkimi.
- W skład hasła nie powinny wchodzić ciągi znaków będące słowami z języka potocznego bądź figurujące w jakimkolwiek słowniku; w zamian wykorzystać można bezsensowne, lecz łatwe do zapamiętania zbitki liter, przeplatane sekwencjami cyfr i znaków specjalnych.

Jeszcze raz należy podkreślić, że tylko hasła w pełni losowe okazują się wysoce odporne na atak słownikowy.

13.3.2. Słabości systemów haseł

Identyfikacja oparta na systemie haseł związana jest organicznie z wieloma niedogodnościami, między innymi:

- W procesie identyfikacji hasło Peggy staje się znane Wiktorowi, który później może się pod nią podszywać.
- Wiktor nigdy nie identyfikuje się wobec Peggy, więc Oskar może podszyć się pod Wiktora w celu poznania hasła Peggy;
- Hasło, które Peggy komunikuje Wiktorowi, nie jest zależne od czasu; Oskar może wykorzystać ten fakt do późniejszego ataku przez powtórzenie.

Konsekwencje pierwszej z wymienionych słabości można złagodzić przez szyfrowanie lub haszowanie hasła natychmiast po wprowadzeniu i dalsze przetwarzanie go w takiej zaszyfrowanej (haszowanej) formie. Nie zmniejsza to jednak niebezpieczeństwa „podejrzenia" sekwencji klawiszy naciskanych przy wprowadzaniu hasła. Przy okazji stajemy przed koniecznością odpowiedzi na intrygujące pytanie: czy jest możliwe zweryfikowanie sekretnej informacji bez poznawania samego sekretu? Odpowiedź na to pytanie jest twierdząca — przykłady takiej weryfikacji poznamy w następnych punktach.

Drugi ze wspomnianych faktów oznacza, że relacja Peggy-Wiktor (czyli relacja „udowadniający-weryfikator") jest wysoce asymetryczna: z jednej strony Peggy musi uwierzytelnić się wobec Wiktora, z drugiej jednak strony o tożsamości Wiktora nic nie wie. Asymetria ta stanowi główną przeszkodę w upowszechnieniu opartej na hasłach identyfikacji w warunkach współdziałania równorzędnych partnerów, na przykład współpracujących ze sobą procesów równoległych. Co więcej, asymetria ta w połączeniu z pierwszą z wymienionych przesłanek umożliwia oszukiwanie użytkownika w celu wydobycia od niego upragnionego hasła. Oszustwo takie realizuje się najczęściej za pomocą terminala odłączonego od hosta, a przyłączonego do komputera (wyłącznie) kolekcjonującego hasła. Użytkownik, po wprowadzeniu swej identyfikacji i hasła, zamiast spodziewanego ekranu powitalnego widzi najczęściej komunikat w rodzaju

```
Host jest tymczasowo niedostępny ze względów technicznych
Spróbuj ponownie za 30 minut
```

(cóż bowiem innego zaoferować może ów „zastępczy" komputer?). Intruz może nawet faktycznie przyłączyć z powrotem terminal po 30 minutach i nieświadomy użytkownik nawet nie zda sobie sprawy z faktu, że jego hasło zostało zwyczajnie wykradzione. Pokrewnymi narzędziami opisywanego oszustwa są specjalnie spreparowane programy udające proces logowania: użytkownik, widząc znajome okno logowania (które faktycznie jest jednak oknem wspomnianego programu, nie systemowym oknem logowania!) wpisuje swą identyfikację i hasło, otrzymując inny komunikat zastępczy:

```
Błędne hasło, spróbuj ponownie
```

Oszukany użytkownik może z dużym prawdopodobieństwem uwierzyć, iż faktycznie pomylił się przy wprowadzaniu hasła i nic nie wiedzieć o tym, że hasło to właśnie mu ukradziono.

Niezależność hasła od czasu uniemożliwia Wiktorowi rozróżnienie, czy podawane właśnie hasło jest autentyczne, czy też stanowi kopię hasła przechwyconego jakiś czas temu. Ta okoliczność także jest wykorzystywana do projektowania rozmaitych ataków na systemy haseł.

13.4. Identyfikacja „wyzwanie-odpowiedź"

Identyfikacja wymieniona w tytule nazywana jest także *silną identyfikacją podmiotów* (ang. *strong entity authentication*) lub *protokołem negocjowania* (dosłownie „protokołem uścisku dłoni", ang. *handshaking protocol*). Informacja taka ma formę dialogu między P a V, w którym to dialogu nie wymieniają oni między sobą żadnych haseł; zamiast tego *wspólne* hasło, znane P i V, wykorzystywane jest do generowania „poprawnych" odpowiedzi na losowe wyzwania. To wspólne hasło pełni w tym kontekście funkcję klucza kryptograficznego, służącego do szyfrowania wyzwań. Protokół „wyzwanie-odpowiedź" może być także wykorzystywany przez P i V do upewnienia się, że realizowany przez nich uprzednio protokół prowadzący do wynegocjowania wspólnego klucza zakończył się powodzeniem; innymi słowy, P i V mogą (i powinni) upewnić się, że dysponują poprawną kolekcją kluczy.

13.4.1. Uwierzytelnianie współdzielonych kluczy

Załóżmy, że dwa równorzędne podmioty A i B (nazwijmy je Alicją i Bobem) chciałyby zweryfikować przypuszczenie, iż dysponują tym samym kluczem kryptograficznym k. Typowy protokół, realizujący taką weryfikację, może mieć postać następującą.

Protokół „wyzwanie-odpowiedź” (weryfikacja współdzielonego klucza)

Cel: Wzajemne uwierzytelnienie A i B drogą sprawdzenia, czy współdzielą oni ten sam klucz k.
Założenia: A i B wybierają losowo dwie liczby losowe (*nonces*) r_A i r_B i wykorzystują ten sam algorytm szyfrujący.
Sekwencja komunikatów:

(1) $A \to B : r_A$.

(2) $B \to A : \{A, r_A, r_B\}_k$.

(3) $A \to B : \{B, r_B\}_k$,

gdzie $\{A, r_A, r_B\}_k$ oznacza wynika zaszyfrowania komunikatu $\{A, r_A, r_B\}$ kluczem k.

Przedstawiony protokół funkcjonuje następująco. Najpierw A ujawnia swe wyzwanie r_A przed B. W odpowiedzi B konkatenuje nazwę A i jego wyzwanie (r_A) ze swym własnym (r_B); tak otrzymaną trójkę szyfruje kluczem k i odsyła A wynik tego szyfrowania. A otrzymany kryptogram rozszyfrowuje, wydobywa z niego parę *nonces* i sprawdza, czy pierwsza z nich równa jest r_A — jeśli tak jest, to znaczy, że A i B używają tego samego klucza szyfrującego. Taką pewność musi jeszcze uzyskać B: w tym celu A wysyła mu kryptogram $\{B, r_B\}_k$; B po rozszyfrowaniu kryptogramu weryfikuje, czy pierwszy jego element jest nazwą B, a drugi — liczbą r_B: jeśli tak, to B ma prawo być przekonany, że A korzysta z tego samego klucza szyfrującego, co on.

Bezpieczeństwo przedstawionego protokołu zależy od trzech czynników: długości klucza k, jakości algorytmu szyfrującego oraz świeżości wyzwań r_A i r_B. Protokół ten łatwo można zaadaptować do warunków identyfikacji jednostronnej (A przeprowadza identyfikację B, lecz nie odwrotnie).

Algorytm szyfrowania można zastąpić funkcją jednokierunkową, bazującą na bezkolizyjnej funkcji haszującej. Jeśli A i B stosują tę samą funkcję haszującą h, to sekwencja wymienianych między nimi komunikatów ma postać następującą:

(1) $A \to B : r_A$.

(2) $B \to A : r_B, h(A, r_A, r_B, k)$.

(3) $A \to B : h(k, B, r_B, r_A)$.

A komunikuje B swe wyzwanie r_A; w odpowiedzi B haszuje zestaw (A, r_A, r_B, k) i odsyła A parę $(r_B, h(A, r_A, r_B, k))$. A weryfikuje wynik haszowania i odsyła B wartość $h(k, B, r_B, r_A)$. Zwróćmy uwagę na dokonywaną przez A zmianę kolejności r_A i r_B — ma ona uodpornić protokół na ewentualne późniejsze ataki przez powtórzenie.

13.4.2. Uwierzytelnianie za pomocą kluczy publicznych

Załóżmy, że A i B nawzajem znają swe klucze publiczne K_A i K_B; oczywiście każde z nich zna swój klucz prywatny (odpowiednio) k_A i k_B. Przypuśćmy teraz, że każde z nich chciałoby sprawdzić, czy klucz prywatny jego partnera istotnie jest komplementarny w stosunku do (znanego publicznie) klucza publicznego. Jak pamiętamy, system szyfrowania z kluczami publicznymi może być użyty zarówno do ukrycia informacji (tajności), jak i do weryfikacji jej autentyczności (podpisy cyfrowe). Protokół „wyzwanie-odpowiedź" dla jednostronnego uwierzytelnienia B przez A w sytuacji, gdy B używa swego klucza publicznego do zapewnienia tajności, funkcjonuje następująco.

Protokół „wyzwanie-odpowiedź" (szyfrowanie z kluczem publicznym)

Cel: A identyfikuje B za pomocą sprawdzenia, czy posiadany przez B klucz prywatny k_B jest komplementarny w stosunku do jego klucza publicznego K_B.
Założenia: A wybiera losowo wyzwanie r_A . B wykorzystuje własną parę kluczy w celu zachowania poufności.
Sekwencja komunikatów:

(1) $A \to B : [r_A, A]_{K_B}$.

(2) $B \to A : r_A$,

gdzie $[r_A, A]_{K_B}$ jest wynikiem zaszyfrowania kryptogramu (r_A, A) kluczem K_B.

A, znając klucz publiczny B (K_B), wysyła mu zaszyfrowany tym kluczem komunikat. Komunikat ten może zostać rozszyfrowany jedynie pod warunkiem znajomości komplementarnego klucza k_B, który może być znany jedynie B. Dowodem tej znajomości jest wydobyta z rozszyfrowanego komunikatu wartość r_A, którą B odsyła A. A zyskuje tym samym pewność, że jego partnerem jest rzeczywiście właściciel klucza K_B, czyli B.

Protokół wymaga pewnych zmian, by B mógł wykorzystać swój system szyfrowania z kluczem publicznym w celach uwierzytelnienia:

Protokół „wyzwanie-odpowiedź" (uwierzytelnianie)

Cel: A identyfikuje B za pomocą sprawdzenia, czy posiadany przez B klucz prywatny k_B jest komplementarny w stosunku do jego klucza publicznego K_B.
Założenia: A wybiera losowo wyzwanie r_A , B wybiera losowo wyzwanie r_B. B wykorzystuje system szyfrowania z kluczem publicznym w celach uwierzytelnienia.
Sekwencja komunikatów:

(1) $A \to B : r_B$.

(2) $B \to A : r_B, \langle r_A, r_B \rangle_{k_B}$.

A wysyła swe losowe wyzwanie r_A do B. B dołącza swe wyzwanie r_B i podpisuje parę (r_A, r_B) za pomocą klucza k_B. Podpisany komunikat $\langle r_A, r_B \rangle_{k_B}$ wysyłany jest do A wraz z wartością r_B w celu jej porównania z wartością wydobytą z komunikatu. Zwróćmy uwagę, że r_B nie musi być wysyłane w jawnej postaci w przypadku, gdyby sygnatura $\langle r_A, r_B \rangle_{k_B}$ pozwalała na wydobycie wartości r_B.

13.5. Protokoły identyfikacji

Powróćmy do systemów dowodzenia z wiedzą zerową, omawianych w rozdziale 12. Jak pamiętamy, systemy takie umożliwiają udowadniającemu P zademonstrowanie wobec weryfikatora V znajomości pewnego sekretu, bez ujawniania jakichkolwiek jego szczegółów. Własność ta jest idealna dla celów identyfikacji. Zauważmy, że bezpośrednie wykorzystanie systemu dowodzenia z wiedzą zerową umożliwia jednostronną weryfikację udowadniającego P (Peggy) przez weryfikatora V (Wiktora). Protokół identyfikacyjny realizowany będzie wówczas w formie wieloiteracyjnej interakcji między P i V. Pojęcia *zupełności* i *rzetelności* systemu muszą jednak ulec pewnemu przedefiniowaniu w nowych warunkach: protokół identyfikacyjny nazywać będziemy *zupełnym*, jeżeli legalny udowadniający (postępujący zgodnie z protokołem) zawsze zostanie właściwie rozpoznany przez weryfikatora, czyli jeżeli prawdopodobieństwo błędnego odrzucenia wynosi w tym protokole 0. Rzetelność protokołu oznacza natomiast wymóg, żeby prawdopodobieństwo fałszywej akceptacji, czyli pozytywnego zweryfikowania intruza, było jak najmniejsze — nie przekraczające 2^{-t} po wykonaniu t iteracji. Pojęcie „zerowej wiedzy" oznaczać będzie natomiast niemożność poznania sekretu skrywanego przez udowadniającego, przy założeniu pewnych rozsądnych ograniczeń na moc obliczeniową weryfikatora.

W niniejszym punkcie przedyskutujemy protokół identyfikacyjny Fiata-Shamira oraz jego bardziej efektywną odmianę stworzoną przez Feigego, Fiata i Shamira. Następnie omówimy protokół Guillou i Quisquatera, bazujący na tożsamościach podmiotów i szczególnie przydatny do celów identyfikacji za pomocą inteligentnych kart chipowych (ang. *smart cards*). Zajmiemy się także schematem Schnorra i jego odmianą stworzoną przez Okamoto. Inne protokoły identyfikacji, nie omawiane w niniejszym rozdziale, bazują na kodach korekcyjnych [490, 94]. Jednym z bardziej egzotycznych problemów trudnych obliczeniowo, wykorzystanych do projektowania protokołów identyfikacyjnych, jest tzw. problem perceptronowy, wywodzący się z obszaru sieci neuronowych [409]; jest on problemem NP-zupełnym.

13.5.1. Protokół identyfikacyjny Fiata-Shamira

Bezpieczeństwo protokołu identyfikacyjnego Fiata-Shamira [181] zasadza się na trudności obliczania pierwiastków kwadratowych modulo N pod warunkiem nieznajomości czynników pierwszych liczby N. Obliczanie modularnych pierwiastków kwadratowych nie jest bowiem łatwiejsze obliczeniowo od rozkładu na czynniki pierwsze (faktoryzacji).

Protokół identyfikacyjny Fiata-Shamira (FS)

Obliczenia wstępne wykonywane przez ***TA***: *TA* utrzymuje w sekrecie dwie liczby pierwsze p i q, publikując jedynie ich iloczyn $N = pq$.

Rejestracja: P wybiera w sekrecie losową wartość $s \in_R \mathbb{Z}_N^*$, po czym rejestruje w *TA* wartość $\sigma \equiv s^2 \pmod{N}$ jako swą publiczną informację identyfikacyjną.

Sekwencja komunikatów: P demonstruje wobec V znajomość wartości s, wykonując t-krotnie następującą sekwencję:

(1) $P \rightarrow V : u \equiv r^2 \bmod N$ gdzie $r \in_R \mathbb{Z}_N^*$.

(2) $V \to P : b \in \{0,1\}$.

(3) $P \to V : v \equiv r * s^b \ \mathbf{mod}\ N$.

Weryfikacja: V sprawdza, czy

$$v^2 \stackrel{?}{\equiv} u * \sigma^b \ \mathbf{mod}\ N .$$

Jeśli nie, to V zatrzymuje protokół w stanie odrzucenia; w przeciwnym razie wykonywana jest następna runda. Po wykonaniu t rund protokół zatrzymuje się w stanie akceptacji.

Zaufane centrum autoryzacji *TA* przechowuje informację identyfikacyjną wszystkich zarejestrowanych użytkowników; warunkiem uruchomienia opisanego protokołu jest uprzednie zarejestrowanie się udowadniającego *P*. Rejestracja ta przeprowadzona zostanie w postaci wzajemnego uwierzytelnienia *TA* i *P* poprzez okazanie stosownych dokumentów (paszportu, karty identyfikacyjnej ze zdjęciem itp.) — jest to czynność krytyczna z punktu widzenia bezpieczeństwa.

Na wstępie weryfikator *V* musi upewnić się, że *P* jest tym samym podmiotem (osobą), którego publiczna (opublikowana przez *TA*) informacja identyfikacyjna równa jest σ; zadaniem udowadniającego *P* jest przekonanie weryfikatora *V*, że jest tak istotnie. Proces przekonywania *V* przebiega w *t* iteracjach; każda runda jest niezależna od pozostałych w tym sensie, że rozpoczyna się od wybrania przez *P* losowej wartości *r*, którą następnie podnosi on do kwadratu modulo *N* i wysyła wynik do *V*. *V* komunikuje następnie swe binarne wyzwanie *b*, w odpowiedzi na co *P* odsyła wartość $v \equiv r * s^b \ \mathbf{mod}\ N$. Ostatecznie *V* podnosi otrzymaną wartość do kwadratu i porównuje z wartością $u * \sigma^b$; jeżeli obydwie wartości równe są modulo *N*, wykonywana jest następna runda protokołu, w przeciwnym razie *V* zatrzymuje protokół w stanie odrzucenia. Po wykonaniu *t* rund weryfikator *V* uznaje się za przekonanego — protokół zatrzymuje się w stanie akceptacji.

Oszust Oskar może oszukać Wiktora (czyli weryfikator *V*), jeśli będzie w stanie zgadnąć wybraną przez *V* wartość *b*. Niech $g \in_R \{0,1\}$ będzie domniemywaną przez Oskara wartością *b*; Oskar wybiera losowo wartość *r* i wysyła Wiktorowi swe zobowiązanie

$$u \equiv r^2 \sigma^{-g} \ \mathbf{mod}\ N ,$$

otrzymując w odpowiedzi (od Wiktora) losową wartość binarną *b*. I teraz Oskar staje przed problemem obliczenia wartości

$$v \equiv r * s^{b-g} \ \mathbf{mod}\ N ,$$

by pozytywnie przejść test porównania

$$v^2 \stackrel{?}{\equiv} u * \sigma^b \ \mathbf{mod}\ N .$$

Porównanie to równoważne jest porównaniu

$$v^2 = r^2 \sigma^{b-g} \stackrel{?}{\equiv} u * \sigma^{b-g} .$$

Jeśli $g \neq b$, to $b - g$ równe jest 1 albo –1 i w celu obliczenia poprawnej wartości $v \equiv r * s^{b-g}$ **mod** N Oskar musiałby znać wartość s albo s^{-1}; wartość s jest jednak utrzymywana w tajemnicy przez P. Ponieważ w każdej iteracji prawdopodobieństwo tego, że $g = b$, wynosi $\frac{1}{2}$, z takim właśnie prawdopodobieństwem oszustwo Oskara nie zostanie wykryte. Gdy wykonywanych jest t iteracji, prawdopodobieństwo niewykrycia oszustwa (czyli prawdopodobieństwo fałszywej akceptacji) wynosi 2^{-t}.

Zilustrujmy opisane obliczenia konkretnym przykładem. Niech p = 4787 i q = 9643, wtedy publikowana przez *TA* wartość $N = pq$ równa jest 46161041. P wybiera w tajemnicy wartość s = 21883917 i rejestruje swą publiczną identyfikację równą $\sigma = s^2 \equiv 25226214$ (**mod** 46161041). Niech w pierwszej rundzie protokołu losowo wybrana wartość r równa będzie 41435437. P wysyła swe zobowiązanie:

$$P \rightarrow V : u = r^2 \equiv 6360246 \,(\textbf{mod}\ 46161041).$$

V odpowiada losowym wyzwaniem binarnym b = 1, w odpowiedzi na co P wysyła wartość

$$P \rightarrow V : v = rs \equiv 39085596 \,(\textbf{mod}\ 46161041).$$

Ostatecznie V dokonuje porównania dwóch wartości:

$$v^2 \equiv 42178320 \,(\textbf{mod}\ 46161041)$$

oraz

$$u\sigma \equiv 42178320 \,(\textbf{mod}\ 46161041).$$

Ponieważ obydwie te wielkości mają tę samą wartość, protokół przechodzi do następnej rundy.

13.5.2. Protokół identyfikacyjny Feigego-Fiata-Shamira

Protokół Fiata-Shamira (FS) wymaga dużej liczby iteracji, w konsekwencji czego proces identyfikacji jest powolny i kosztowny obliczeniowo — zarówno dla udowadniającego, jak i dla weryfikatora. Feige, Fiat i Shamir zaproponowali w związku z tym bardziej efektywny protokół [167], którego bezpieczeństwo uzależnione jest od obliczeniowej trudności faktoryzacji (rozkładu liczby złożonej na czynniki pierwsze).

Protokół identyfikacyjny Feigego-Fiata-Shamira (FFS)

Obliczenia wstępne wykonywane przez ***TA***: *TA* wybiera dwie liczby pierwsze p i q takie, że $p \equiv 3$ **mod** 4 i $q \equiv 3$ **mod** 4; utrzymuje je w sekrecie, publikując ich iloczyn $N = pq$.

Rejestracja: P wykonuje kolejno następujące czynności:

(1) wybiera losowo ℓ liczb całkowitych $s_1, \ldots, s_\ell \in_R \mathbb{Z}_N^*$,

(2) wybiera losowo wektor bitowy $(e_1, \ldots, e_\ell)$,

(3) oblicza $w_i \equiv (-1)^{e_i} (s_i)^{-2} \pmod{N}$ dla $i = 1, \ldots, \ell$,

(4) rejestruje w *TA* wektor $(w_1, \ldots, w_\ell)$ jako swoją informację identyfikacyjną, zachowując jednocześnie w tajemnicy wektor $(s_1, \ldots, s_\ell)$.

Sekwencja komunikatów: *P* demonstruje wobec *V* znajomość wektora $(s_1, \cdots, s_\ell)$, wykonując *t*-krotnie następującą sekwencję:

(1) $P \to V : u \equiv r^2 \pmod{N}$, gdzie $r \in_R \mathbb{Z}_N^*$.

(2) $V \to P : (b_1, \ldots, b_\ell) \in_R \{0,1\}^\ell$.

(3) $P \to V : v \equiv r \prod_{i=1}^{\ell} (s_i)^{b_i} \pmod{N}$.

(4) Weryfikacja: *V* sprawdza, czy

$$u \overset{?}{\equiv} \pm v^2 \prod_{i=1}^{\ell} (w_i)^{b_i} \pmod{N}.$$

Jeśli nie, to *V* zatrzymuje protokół w stanie odrzucenia; w przeciwnym razie wykonywana jest następna runda. Po wykonaniu *t* rund protokół zatrzymuje się w stanie akceptacji.

Oskar, zamierzający wcielić się w *P*, może liczyć na sukces tylko wówczas, gdy uda mu się zgadnąć wybrany przez *V* wektor $(b_1, \ldots, b_\ell)$. Oznaczmy domniemywany przez Oskara wektor jako $(g_1, \ldots, g_\ell)$; Oskar wybiera losową wartość $r \in_R \mathbb{Z}_N^*$ i wysyła do *V* swe zobowiązanie, oczywiście bazując na wektorze $(g_1, \ldots, g_\ell)$:

$$u \equiv r^2 \prod_{i=1}^{\ell} (w_i)^{g_i} \pmod{N}.$$

Teraz kolej na *V*, który przesyła Oskarowi swe wyzwanie $(b_1, \ldots, b_\ell)$. Jeśli $(b_1, \ldots, b_\ell) = (g_1, \ldots, g_\ell)$, Oskar po prostu przesyła w odpowiedzi wartość *v* równą *r*. *V* sprawdza następnie, czy zachodzi równość

$$u \equiv v^2 \prod_{i=1}^{\ell} (w_i)^{b_i} \pmod{N}.$$

Załóżmy, że Oskar dokonał jakiegoś wyboru (przypuszczenia) $(g_1, \ldots, g_\ell)$ i wysłał swe zobowiązanie $u \equiv r^2 \prod_{i=1}^{\ell} (w_i)^{g_i} \pmod{N}$, a w odpowiedzi otrzymał takie wyzwanie $(b_1, \ldots, b_\ell)$, że $b_1 \neq g_1$, natomiast $b_i = g_i$ dla $i = 2, \ldots, \ell$ — pomylił się więc w swych przypuszczeniach co do pierwszego elementu wektora. Oskar zamierza odpowiedzieć, wysyłając albo wartość rs_1 (gdy $g_1 = 0$ i $b_1 = 1$),

albo wartość $r(s_1)^{-1}$ (gdy $g_1 = 1$ i $b_1 = 0$); w każdym z tych dwóch przypadków musiałby jednak znać wartość s_1, tymczasem wszystkie wartości s_i utrzymywane są przez P w tajemnicy i bez znajomości współczynników w_i nie sposób obliczyć ich w rozsądnym czasie. Prawdopodobieństwo, że po t rundach oszustwo Oskara pozostanie niewykryte, wynosi już nie 2^{-t}, ale $2^{-\ell t}$.

Przyjmijmy przykładowe wartości p = 1367 i q = 1103; łatwo sprawdzić, że $p \equiv 3 \bmod 4$, $q \equiv 3 \bmod 4$ i N = 1507801. Niech $\ell = 4$, zatem P wybiera cztery losowe liczby całkowite — niech będą to:

s_1 = 1281759,
s_2 = 63306,
s_3 = 100742,
s_4 = 647983.

Następnie V wybiera losowo wektor binarny — niech będzie to wektor (1, 1, 0, 1) — i oblicza:

$$w_1 = (-1)(s_1)^{-2} \equiv 559476 \bmod 1507801$$
$$w_2 = (-1)(s_2)^{-2} \equiv 1445404 \bmod 1507801$$
$$w_3 = (s_3)^{-2} \equiv 663524 \bmod 1507801$$
$$w_4 = (-1)(s_4)^{-2} \equiv 120740 \bmod 1507801$$

Wektor (w_1, w_2, w_3, w_4) stanowi publiczną identyfikację P i zarejestrowany zostaje w TA. Gdy P zamierza przekonać V co do znajomości wektora (w_1, w_2, w_3, w_4), nawiązuje z nim konwersację przebiegającą w t iteracjach (rundach). Pokażemy przykładowy przebieg jednej z rund. P rozpoczyna rundę, wybierając r = 736113 i wysyła do V swe zobowiązanie $u = r^2 \equiv 887797 \bmod 1507801$. V odpowiada czterobitowym wyzwaniem — niech będzie nim (1, 0, 1, 0); takiemu wektorowi odpowiada wartość $v = rs_1s_3 \equiv 1045302 \bmod 1507801$, którą P odsyła do V. V sprawdza teraz, czy

$$u \stackrel{?}{\equiv} \pm v^2 w_1 w_3 \pmod N.$$

Faktycznie 887797 ≡ ± 620004 (**mod** 1507801). Protokół przechodzi do następnej rundy.

13.5.3. Protokół identyfikacyjny Guillou-Quisquatera

Protokół wymienionych w tytule autorów, nazywany skrótowo protokołem GQ, jest modyfikacją protokołu FS opisaną w [223]. Jego bezpieczeństwo opiera się na (domniemanej, lecz wciąż nie udowodnionej) trudności obliczeniowej faktoryzacji, czyli rozkładu liczby złożonej na czynniki pierwsze. Atrakcyjną cechą protokołu GQ jest oparcie go na tożsamościach podmiotów, dzięki czemu weryfikator nie musi posługiwać się żadnymi certyfikowanymi elementami, wyjąwszy oczywiście publicznie dostępną tożsamość udowadniającego i publiczny klucz TA.

Protokół identyfikacyjny Guillou-Quisquatera (GQ)

Obliczenia wstępne wykonywane przez ***TA***: *TA* wybiera dwie liczby pierwsze *p* i *q*; utrzymuje je w sekrecie, publikując ich iloczyn $N = pq$. Następnie TA wybiera dwa wykładniki *e* i *d* takie, że $d * e \equiv 1 \bmod \varphi(N)$, gdzie $\varphi(N)$ jest funkcją Eulera. Wartość *d* podawana jest do publicznej wiadomości, wartość *e* utrzymywana jest w tajemnicy.

Rejestracja: (1) Udowadniającemu *P* przypisywana jest unikalna tożsamość ID_P. Jest ona konwertowana na unikalną liczbę całkowitą $J_P \ (1 \le J_P \le N-1)$, nazywaną *ukrytą tożsamością* (ang. *shadowed identity*). Konwersja ta jest publicznie znana.

(2) *TA* podpisuje ukrytą tożsamość J_P za pomocą tajnego klucza *e*, co daje w wyniku wartość $\sigma \equiv (J_P)^{-e} \ (\bmod N)$. Wartość ta komunikowana jest *P*; *P* weryfikuje podpis, sprawdzając, czy $\sigma^d \equiv (J_P)^{-1} \bmod N$. Wartość σ utrzymywana jest w tajemnicy zarówno przez *P*, jak i przez *TA*.

Sekwencja komunikatów: Udowadniający *P* przedstawia się weryfikatorowi *V* jako podmiot o tożsamości ID_P. *V* konwertuje tę tożsamość na wartość liczbową J_P. Proces identyfikacji przebiega w ramach *t* rund, z których każda ma następującą postać:

(1) $P \to V : u \equiv r^d \bmod N$ gdzie $r \in_R \{1, \ldots, N-1\}$.

(2) $V \to P : b \ \ gdzie\ b \in_R \{1, \ldots, d\}$.

(3) $P \to V : v \equiv r * \sigma^b \bmod N$.

Weryfikacja: *V* sprawdza, czy

$$(J_P)^b * v^d \equiv u \bmod N .$$

Jeśli nie, to *V* zatrzymuje protokół w stanie odrzucenia; w przeciwnym razie wykonywana jest następna runda. Po wykonaniu *t* rund protokół zatrzymuje się w stanie akceptacji.

Jak łatwo się zorientować, *TA* wykorzystuje tu kryptosystem RSA z kluczem publicznym *d* i modułem *N*, stosując klucz prywatny *e* do podpisywania J_P. Certyfikat σ utrzymywany jest w tajemnicy zarówno przez *TA*, jak i przez *P*; to właśnie jego znajomość demonstrowana jest przez *P* wobec weryfikatora *V*. Dostępną dla *V* publiczną informację na temat *P* stanowi para (ID_P, J_P).

Każda runda rozpoczyna się od losowego wyboru *r* przez *P*, który następnie wysyła do *V* zobowiązanie $u = r^d$. *V* oznajmia *P* swe losowe wyzwanie *b*, na co *P* odpowiada wartością $v = r * \sigma^b$. *V* sprawdza wówczas, czy $(J_P)^b * v^d$ równe jest r^d.

Zobaczmy teraz, co robi intruz Oskar, próbujący wcielić się w rolę *P*. Po pierwsze, przedstawia się oczywiście, podając tożsamość ID_P.; następnie wybiera losową wartość *r* i próbuje zgadnąć wartość wyzwania, które wybierze *V* — oznaczmy tę domniemywaną wartość przez *g*. Oskar wysyła do *V* swe zobowiązanie

$$O \to V : u \equiv r^d * (J_P)^g \bmod N ,$$

na co *V* odpowiada wyzwaniem *b*. Oskar wyśle wówczas wartość

$$O \to V : v \equiv r\sigma^{g-b} \ \mathbf{mod}\ N ,$$

po czym V sprawdzi, czy

$$(J_P)^b * \left(r\sigma^{g-b}\right)^d \equiv r^d * (J_P)^g \ \mathbf{mod}\ N .$$

Aby Oskar mógł ukryć swe oszustwo, musi wykonać przynajmniej jedną z dwóch rzeczy: przewidzieć wybór wyzwania przez V (tak, by $g = b$) bądź obliczyć σ. W pierwszym przypadku ma szansę jedną na d, w drugim staje wobec trudności obliczeniowej i o obliczaniu σ może raczej zapomnieć. Gdy proces identyfikacji obejmuje t rund, prawdopodobieństwo odgadnięcia przez Oskara *każdego* z t wyzwań V równa się d^{-t} — i jest to jednocześnie prawdopodobieństwo fałszywej akceptacji. Zauważmy, że jeżeli P i V stosują się do protokołu, to — zgodnie z przedstawionym wcześniej wymogiem — fałszywe odrzucenie jest wykluczone.

Protokół GQ zaprojektowany został z myślą o efektywności. Zaleca się wybór niewielkiej wartości d, w granicach 2^{30}; im mniejsza wartość d, tym szybsze obliczenia wykonywane zarówno przez P, jak i przez V. W większości praktycznych zastosowań protokoły GQ dla $d \approx 2^{30}$ mogą wymagać tylko jednej iteracji ($t = 1$). Z drugiej jednak strony zbyt mała wartość d może wymagać bardzo wielu iteracji, by utrzymać na założonym poziomie prawdopodobieństwo błędnego odrzucenia.

Ponownie rozpatrzmy konkretny przykład. Niech parametry RSA wybrane przez *TA* równe będą odpowiednio $p = 563$ i $q = 719$. Wówczas $N = 404797$ i $\varphi(N) = 403516$. Niech $d = 23$, wtedy $e = 298251$. Wartości N i d są publicznie znane; ukryta tożsamość obliczona dla P wynosi $J_P = 123456$. *TA* udostępnia P jego sekretną wartość $\sigma \equiv (J_P)^{-e} \equiv 79833 \ \mathbf{mod}\ 404797$; P weryfikuje tę wartość, sprawdzając równość

$$\mathrm{J_P} = 123456 \stackrel{?}{\equiv} \sigma^{-\mathrm{d}} = 123456 \ (\mathbf{mod}\ 404797) .$$

Obydwie strony tej równości wynoszą 123456, co upewnia P odnośnie prawdziwości σ.

Gdy V zapyta P o jego tożsamość, P zaprezentuje wartość ID_P, po czym P i V przystąpią do wykonywania kolejnych rund protokołu. Rozpatrzmy przykładową rundę: P wybiera losowo $r = 133504$ i oznajmia V swe zobowiązanie $u = r^d \equiv 172296 \ \mathbf{mod}\ 404797$. V przekazuje P losowe wyzwanie $b = 11$; zgodnie z oczekiwaniami odpowiedź P równa jest $v = r\sigma^b \equiv 41169 \ \mathbf{mod}\ 404797$. Ostatecznie V oblicza

$$(J_P)^b * v^d \equiv 172296 \ \mathbf{mod}\ 404797 .$$

Wartość ta jest równa u, V i P przechodzą więc do następnej rundy protokołu.

Jak więc widać, systemy dowodzenia z wiedzą zerową mogą służyć jako podstawa realizacji protokółów identyfikacyjnych. Protokół FS jest klasycznym przykładem bezpośredniego użycia takiego systemu. Efektywniejsze od niego protokoły FFS i GQ pozwalają weryfikatorowi na generowanie ℓ-bitowych wyzwań (zamiast jednobitowych, jak w protokole FS). Zwiększa to efektywność protokołu, lecz jednocześnie rodzi pewne problemy: najpoważniejszym z nich jest to, że własność wiedzy zerowej staje się trudniejsza do dowiedzenia. Przypomnijmy, iż punktem wyjścia takiego dowodu jest zaprojektowanie efektywnego symulatora, którego produkt jest nieodróżnialny od widoku generowanego przez rzeczywistą interakcję. Symulator taki wykonywać się będzie

w czasie wielomianowym jedynie wówczas, gdy wyzwanie generowane przez V będzie łańcuchem o długości określonej logarytmem; gdy będzie ono łańcuchem o większej długości, nie będzie można udowodnić, że system jest systemem dowodzenia z wiedza zerową. Staje się to oczywiste w sytuacji, gdy protokół identyfikacyjny obejmuje tylko jedną rundę składającą się z trzech kroków: zobowiązania (świadectwa) $P \to V$, wyzwania $V \to P$ oraz odpowiedzi $P \to V$. Owo pojedyncze wyzwanie musi być wystarczająco długie, by prawdopodobieństwo fałszywej akceptacji można było utrzymać na odpowiednio niskim poziomie. Wyklucza to w oczywisty sposób istnienie efektywnego symulatora interakcji.

13.6. Schematy identyfikacji

Rozpatrzmy „trzykrokowe" protokoły. W dalszym ciągu będziemy mianem *schematów* określać skrócone wersje protokołów identyfikacyjnych o dowolnej liczbie kroków. Niektórzy autorzy podają inne kryterium schematu — jako konstrukcji, z której nie wydostają się na zewnątrz informacje na temat sekretów skrywanych przez udowadniających. Ów brak przecieku informacji konkretyzowany jest jako „*brak przepływu użytecznej informacji*" (ang. *no useful information transfer*) [167] lub jako „*brak przenośnej informacji związanej z bezpieczeństwem*" (ang. *no transferable information with security level*) [388]. Podejście alternatywne opiera się na metodzie dowodu „nie wprost" i polega na udowodnieniu, że przełamanie schematu identyfikacji równoważne byłoby istnieniu algorytmu wielomianowego rozwiązującego problem trudny obliczeniowo (przykładem takiego problemu jest logarytm dyskretny DL).

13.6.1. Schemat identyfikacyjny Schnorra

Schnorr [452] zaprojektował schemat identyfikacyjny nadający się głównie dla inteligentnych kart chipowych, cechujących się bardzo ograniczoną mocą obliczeniową i znikomo małą pamięcią. Bezpieczeństwo schematu opiera się na (domniemanej) trudności obliczeniowej logarytmowania dyskretnego.

Schemat identyfikacyjny Schnorra

Obliczenia wstępne wykonywane przez ***TA***: *TA* inicjuje parametry protokołu:

(1) Wybiera moduł p będący liczbą pierwszą.
(2) Wybiera liczbę pierwszą q, będącą dzielnikiem $p - 1$.
(3) Wybiera liczbę całkowitą $\alpha \in \mathbb{Z}_p^*$, generującą grupę rzędu q — czyli taką, że $\alpha^q \equiv 1 \pmod{p}$.
(4) Określa kolekcję możliwych wyzwań $\{0, 1, \ldots, 2^t - 1\}$.
(5) Stosuje swój prywatny klucz do wydawania certyfikatów, podczas gdy klucz publiczny służy do ich weryfikacji.
(6) Udostępnia do publicznej wiadomości p, q, α, t i swój klucz publiczny.

Rejestracja: *P* podejmuje następujące kroki w celu uzyskania certyfikatu od *TA*:

(1) Wybiera losowo swój klucz prywatny $s \in_R \mathbb{Z}_q^*$ i na jego podstawie oblicza swój klucz publiczny $K \equiv \alpha^{-s} \bmod p$.

(2) Rejestruje klucz K w TA, w związku z czym TA publikuje certyfikat (podpis) S dla (ID_P, K).

Sekwencja komunikatów: P udowadnia wobec V swą tożsamość w trzech krokach (komunikatach):

(1) $P \to V : (ID_P, K, S, u)$, gdzie S jest certyfikatem wygenerowanym przez TA dla (ID_P, K), $u \equiv \alpha^r \bmod p$ dla losowej liczby całkowitej $r \in_R \mathbb{Z}_q$.

(2) V weryfikuje certyfikat S.

(3) $V \to P : b \in_R \{0, \ldots, 2^t - 1\}$.

(4) $P \to V : y \equiv r + sb \bmod q$.

(5) Weryfikacja:

$$u \stackrel{?}{\equiv} \alpha^y K^b \bmod p .$$

Jeżeli powyższa równość jest prawdziwa, to V akceptuje oświadczenie P, w przeciwnym razie je odrzuca.

TA dostarcza publicznych parametrów dla systemu. Klucz publiczny TA używany jest do weryfikowania certyfikatu S udowadniającego. Protokół schematu realizuje się w trzech krokach. P wybiera losowo liczbę całkowitą $r \in_R \mathbb{Z}_q$, wylicza swe zobowiązanie $u \equiv \alpha^r \bmod p$ i wysyła do V czwórkę (ID_P, K, S, u). V sprawdza, czy para (ID_P, K) oraz certyfikat S pasują do siebie; jeśli tak, to wysyła do P losowe wyzwanie b, na co P odpowiada zwróceniem wartości $y \equiv r + sb \bmod q$. Ostatecznie V porównuje wartość u z wartością $\alpha^y K^b \bmod p$.

Jest oczywiste, że jeżeli P trzyma się zasad protokołu, zawsze zostanie poprawnie zidentyfikowany przez V. Z drugiej strony intruz Oskar mógłby oszukiwać, gdyby tylko udało mu się zgadnąć wartość wyzwania, które V wygeneruje w następnym kroku — oznaczmy to przypuszczenie przez g. Zamiast obowiązkowej wartości $u = \alpha^r$ Oskar wysłałby swe zobowiązanie

$$u \equiv \alpha^r * K^g \bmod p .$$

Po otrzymaniu od V wyzwania b Oskar musiałby wysłać w odpowiedzi wartość

$$y \equiv (r + (b - g)s) \bmod q .$$

Jeśli $b \equiv g \pmod q$, to wartość ta jest równa $r \bmod q$ i oszustwo Oskara nie zostałoby rozpoznane; prawdopodobieństwo wystąpienia takiego zdarzenia wynosi 2^{-t}— jest to prawdopodobieństwo fałszywej akceptacji.

Zilustrujmy ten schemat konkretnymi wartościami liczbowymi (w praktyce wartości te powinny być znacznie większe niż tutaj, by można było w ogóle mówić o bezpieczeństwie protokołu):

niech parametry wybrane przez *TA* równe będą odpowiednio $p = 285457$, $q = 313$, $\alpha = 146159$. *P* wybiera swój klucz prywatny $s = 237$ i oblicza odpowiadający mu klucz publiczny $K = \alpha^{-s} \equiv 166428 \textbf{ mod } 285457$. *P* rejestruje w *TA* swą tożsamość i swój klucz publiczny, w wyniku czego *TA* publikuje klucz *K* i certyfikat *S* dla *P* (ID_P, K).

Załóżmy, że *V* żąda od *P* zidentyfikowania się. *P* wybiera losowo liczbę *r* — w naszym przypadku $r = 133$ — oblicza swe zobowiązanie $u = r^2 \equiv 36157 \textbf{ mod } 285457$ i wysyła do *V* czwórkę (ID_P, K, S, u). *V* sprawdza następnie, czy para (ID_P, K) i certyfikat *S* pasują do siebie (pominiemy szczegóły tej operacji) — jeżeli tak, *V* wysyła do *P* swe wyzwanie, w naszym przykładzie $b = 167$. *P* oblicza wartość $y \equiv (r + sb) \textbf{ mod } q = 274$ i przesyła ją do *V*. *V* oblicza

$$a^y K^b = (146159^{274} * 166428^{167}) \equiv 36157 \textbf{ mod } 285457 .$$

Ponieważ wartość ta równa jest *u*, *V* akceptuje identyfikację *P*.

Schemat Schnorra jest schematem bardzo efektywnym: udowadniający (procesor karty inteligentnej) musi wykonać tylko jedno potęgowanie modulo *p*, by wyliczyć swe zobowiązanie. Obliczenie odpowiedzi *y* wymaga pojedynczego mnożenia i jednego dodawania modulo *q*. Bezpieczeństwo protokołu zasadza się na (domniemanej) trudności obliczeniowej logarytmowania dyskretnego — zakładając tę trudność, można udowodnić odporność schematu na ataki bierne.

13.6.2. Schemat identyfikacyjny Okamoto

Zmodyfikowana wersja schematu Schnorra, także bazująca na obliczeniowej trudności logarytmowania dyskretnego, zaproponowana została przez Okamoto [388]. Bazując na wspomnianej trudności, można udowodnić odporność tego schematu na ataki czynne.

Schemat identyfikacyjny Okamoto

Obliczenia wstępne wykonywane przez ***TA***: *TA* inicjuje parametry protokołu:

(1) Wybiera moduł *p*, będący liczbą pierwszą.
(2) Wybiera liczbę pierwszą *q*, będącą dzielnikiem *p*–1.
(3) Wybiera dwie liczby całkowite α_1 i α_2, będące elementami rzędu *q* w grupie Z_p^*.
(4) Wybiera liczbę *t* rzędu $O(p)$, powiedzmy $t \geq 20$.
(5) Stosuje swój prywatny klucz do wydawania certyfikatów, podczas gdy klucz publiczny służy do weryfikacji.
(6) Publikuje *p*, *q*, α_1, α_2, *t* i swój klucz publiczny.

Rejestracja: *P* podejmuje następujące kroki w celu uzyskania certyfikatu od *TA*:

(1) Wybiera losowo swój klucz prywatny $(s_1, s_2) \in_R Z_q \times Z_q$ i na jego podstawie oblicza swój klucz publiczny $K \equiv \alpha_1^{-s_1} \alpha_2^{-s_2} \textbf{ mod } p$.
(2) Rejestruje klucz *K* w *TA*, w związku z czym *TA* publikuje certyfikat (podpis) *S* dla (ID_P, K).

Sekwencja komunikatów: P udowadnia swą tożsamość wobec V:

(1) $P \to V : (ID_P, K, S, u)$, gdzie S jest certyfikatem wygenerowanym przez TA dla (ID_P, K), $u \equiv \alpha_1^{r_1}\alpha_2^{r_2} \pmod p$ dla losowych liczb całkowitych $r_1, r_2 \in_R Z_q$.

(2) V weryfikuje certyfikat S.

(3) $V \to P : e \in_R \{0, \ldots, 2^t - 1\}$.

(4) $P \to V : \quad y_1 \equiv r_1 + es_1 \pmod q, \quad y_2 \equiv r_2 + es_2 \pmod q$.

(5) Weryfikacja:

$$u \stackrel{?}{\equiv} \alpha_1^{y_1}\alpha_2^{y_2}K^e \pmod p .$$

Jeżeli powyższa równość jest prawdziwa, to V akceptuje oświadczenie P, w przeciwnym razie je odrzuca.

Zilustrujmy opisany schemat konkretnymi wartościami liczbowymi. Niech p = 6491, q = 59, α_1 =1764, α_2 = 4269, t = 5, s_1 = 21, s_1 = 47. Klucz publiczny TA ma wartość

$$K = \alpha_1^{-s_1}\alpha_2^{-s_2} = 1764^{-21}4269^{-47} \equiv 5196 \pmod{6491}.$$

P wybiera losowo r_1 = 13, s_2 = 33 i oblicza swe zobowiązanie

$$u = \alpha_1^{r_1}\alpha_2^{r_2} = 1764^{13}4269^{33} \equiv 1131 \pmod{6491}.$$

po czym przesyła je do V. V odpowiada wyzwaniem e = 12, wobec czego P oblicza kongruencje

$$y_1 = r_1 + es_1 \equiv 29 \pmod{59},$$
$$y_2 = r_2 + es_2 \equiv 7 \pmod{59}.$$

Otrzymawszy y_1 i y_2, V oblicza wartość

$$\alpha_1^{y_1}\alpha_2^{y_2}K^e = 1764^{29}4269^{7}5196^{12} \equiv 1131 \pmod{6491}$$

i porównuje ją z wartością u. Wynik porównania jest pozytywny, więc V akceptuje identyfikację P.

13.6.3. Schematy identyfikacyjne a podpis elektroniczny

Schematy identyfikacji mogą być łatwo przekształcone w schematy podpisu elektronicznego. Należy w tym celu zastąpić weryfikator funkcją haszującą, posiadającą dwa argumenty: podpisywany komunikat i zobowiązanie; wartość tej funkcji (skrót) staje się odpowiednikiem wyzwania. Poniższy schemat podpisu elektronicznego wywodzi się wprost ze schematu identyfikacyjnego Schnorra.

Schemat Schnorra podpisu elektronicznego

Obliczenia wstępne wykonywane przez ***TA***: *TA* inicjuje parametry protokołu, a w szczególności:

(1) Wybiera moduł p będący liczbą pierwszą ($p>2^{512}$), liczbę pierwszą q będącą dzielnikiem $(p-1)$($q>2^{140}$) oraz liczbę całkowitą $\alpha \in \mathbb{Z}_p^*$ będącą generatorem grupy rzędu q.

(2) Wybiera funkcję haszującą $h : \mathbb{Z}_p \times \mathbb{Z} \to \{0, 1, \ldots, 2^t - 1\}$.

(3) Stosuje swój prywatny klucz do wydawania certyfikatów, podczas gdy klucz publiczny służy do weryfikacji.

(4) Publikuje p, q, α, h i swój klucz publiczny.

Rejestracja: Podmiot podpisujący (ang. *signer*) S podejmuje następujące kroki w celu uzyskania certyfikatu od *TA*:

(1) Wybiera losowo swój klucz prywatny $s \in_R \mathbb{Z}_q^*$ i na jego podstawie oblicza swój klucz publiczny $K \equiv \alpha^{-s} \bmod p$.

(2) Rejestruje klucz K w *TA*, w związku z czym *TA* publikuje certyfikat (podpis) S dla (ID_S, K).

Podpisywanie: Aby podpisać komunikat m, S wybiera losowo liczbę całkowitą $r \in \mathbb{Z}_q$ i oblicza $u = \alpha^r \bmod p$ oraz skrót $b = h(u, m)$ dla komunikatu $m \in \mathbb{Z}$. Podpisem elektronicznym komunikatu m jest para $SG_s = (b, y)$, gdzie

$$y \equiv (r + sb) \bmod q .$$

Weryfikacja: Weryfikator V otrzymuje komunikat $\overline{m}$, jego podpis $(\overline{b}, \overline{y})$, pobiera z *TA* klucz publiczny K (wraz z niezbędnymi parametrami), po czym rekonstruuje wartość

$$\overline{u} \equiv \alpha^{\overline{y}} K^{\overline{b}} \bmod p$$

i sprawdza, czy

$$\overline{b} \stackrel{?}{\equiv} h(\overline{u}, \overline{m}) \pmod p .$$

Jeśli wynik sprawdzenia jest pozytywny, podpis uważany jest za autentyczny.

Jak wykazali Pointcheval i Stern [410], jeżeli w stosunku do schematu podpisu Schnorra jest możliwe tzw. *fałszerstwo egzystencjonalne*[2] (ang. *existential forgery*), to jednocześnie problem logarytmowania dyskretnego jest łatwo rozwiązywalny w podgrupach. Stwierdzenie to pozostaje prawdziwe w warunkach tzw. losowej wyroczni (ang. *random oracle*).

[2] Jest to fałszerstwo polegające na generowaniu podpisów dla wiadomości, której postaci nie sposób przewidzieć a priori. Pokrewnymi fałszerstwami podpisów są: *fałszerstwo selektywne* (ang. *selective forgery*), polegające na generowaniu podpisów dla wiadomości o znanej postaci, oraz *fałszerstwo uniwersalne* (ang. *universal forgery*), polegające na generowaniu podpisów dla dowolnych wiadomości — *przyp. tłum.*

W podobny sposób można przekształcić schemat identyfikacyjny Okamoto na schemat podpisu elektronicznego.

Schemat Okamoto podpisu elektronicznego

*Obliczenia wstępne wykonywane przez **TA***: *TA* inicjuje parametry protokołu:

(1) Wybiera moduł obliczeń p będący liczbą pierwszą ($p>2^{512}$), liczbę pierwszą q będącą dzielnikiem $(p-1)$ ($q>2^{140}$)) i dwie liczby całkowite α_1 i α_2 będące elementami rzędu q w grupie $\mathbb{Z}_p^*$.

(2) Wybiera funkcję haszującą $h: \mathbb{Z}_P \times \mathbb{Z} \to \{0, 1, \ldots, 2^t - 1\}$.

(3) Stosuje swój prywatny klucz do wydawania certyfikatów, podczas gdy klucz publiczny służy do weryfikacji.

(4) Publikuje p, q, α_1, α_2, h i swój klucz publiczny.

Rejestracja: *P* podejmuje następujące kroki w celu uzyskania certyfikatu od *TA*:

(1) Wybiera losowo swój klucz prywatny $(s_1, s_2) \in_R \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ i na jego podstawie oblicza swój klucz publiczny $K \equiv \alpha_1^{-s_1} \alpha_2^{-s_2} \pmod{p}$.

(2) Rejestruje klucz K w *TA*, w związku z czym *TA* publikuje certyfikat (podpis) S dla (ID_P, K).

Podpisywanie: Aby podpisać komunikat $m \in \mathbb{Z}$, S wybiera losowo dwie liczby całkowite $r_1, r_2 \in_R \mathbb{Z}_q^*$, oblicza $u \equiv \alpha_1^{r_1} \alpha_2^{r_2} \pmod{p}$, znajduje skrót

$$e = h(u, m)$$

i rozwiązuje dwie kongruencje:

$$y_1 \equiv (r_1 + e s_1) \pmod{q},$$
$$y_2 \equiv (r_2 + e s_2) \pmod{q}.$$

Podpisem komunikatu m jest trójka (e, y_1, y_2).

Weryfikacja: Weryfikator V otrzymuje komunikat $\overline{m}$, podpis $(\overline{e}, \overline{y_1}, \overline{y_2})$, pobiera z *TA* klucz publiczny K, oblicza

$$\overline{u} \equiv \alpha_1^{\overline{y_1}} \alpha_2^{\overline{y_2}} K^{\overline{e}} \pmod{p}$$

i sprawdza, czy

$$\overline{e} \stackrel{?}{\equiv} h(\overline{u}, \overline{m}).$$

Jeśli wynik sprawdzenia jest pozytywny, podpis uważany jest za autentyczny.

Jak udowodnił Okamoto, z trudności obliczeniowej logarytmowania dyskretnego wynika odporność powyższego schematu na atak z wybranym komunikatem — pod warunkiem, że funkcja haszująca *h* jest funkcją nieskorelowaną. Istnienie funkcji nieskorelowanych jest wymaganiem silniejszym niż istnienie funkcji bezkolizyjnych (szczegóły w [388]).

13.7. Problemy i ćwiczenia

(1) Rozpatrzmy następujący protokół identyfikacyjny. Peggy podaje swą nazwę Wiktorowi. Wiktor rzuca monetą symetryczną: jeśli wypadnie orzełek, Wiktor akceptuje identyfikację, w przeciwnym razie odrzuca ją. Oblicz prawdopodobieństwo fałszywego odrzucenia i fałszywej akceptacji w tym protokole. Czy protokół ten nadaje się do praktycznego zastosowania?

(2) Wiktor zakupił dwa komputery wyposażone w dodatkowe urządzenia identyfikacyjne. Do pierwszego komputera przyłączony jest czytnik linii papilarnych, a prawdopodobieństwo fałszywego odrzucenia i fałszywej akceptacji wynoszą (odpowiednio) $P_{f_{r_1}}$ oraz $P_{f_{a_1}}$. Urządzenie przyłączone do drugiego komputera bazuje na rozpoznawaniu twarzy, a wspomniane prawdopodobieństwa wynoszą odpowiednio $P_{f_{r_2}}$ oraz $P_{f_{a_2}}$. Wiktor zastanawia się, jak połączyć ze sobą te dwie maszyny, by wymienione prawdopodobieństwa zmniejszyć; rozważa w związku z tym dwa następujące rozwiązania:
   (a) identyfikacja jest przyjęta tylko wówczas, gdy przyjęta zostanie przez każdy z obydwu komputerów,
   (b) identyfikacja jest przyjęta wówczas, gdy przyjęta zostanie przez co najmniej jeden z komputerów.

   Oblicz prawdopodobieństwo fałszywego odrzucenia i fałszywej akceptacji dla każdego z tych rozwiązań (w razie potrzeby możesz poczynić rozsądne założenia o charakterze probabilistycznym).

(3) Wyobraźmy sobie schemat identyfikacyjny oparty na czterocyfrowym numerze PIN. Jakie prawdopodobieństwo zgadnięcia numeru ma intruz (złodziej karty bankomatowej!), jeżeli dopuszczalne są trzy próby wprowadzenia tego numeru?

(4) Rozważmy 10-znakowe hasła, których kolejne znaki wybierane są losowo z pewnego zbioru; rozpatrzmy następujące typy zbiorów:
   (a) wszystkie małe litery alfabetu angielskiego,
   (b) wszystkie litery (małe i wielkie) alfabetu angielskiego,
   (c) wszystkie znaki alfanumeryczne (małe i wielkie litery oraz cyfry) alfabetu angielskiego,
   (d) wszystkie znaki dostępne na typowej klawiaturze (łącznie 96 znaków).

   Jakie jest prawdopodobieństwo zgadnięcia hasła w jednej próbie dla każdego z wymienionych zbiorów? Ile czasu wymagałoby wyczerpujące sprawdzenie wszystkich haseł dla każdego z wymienionych zbiorów, jeśli można by dokonać 1000 sprawdzeń w ciągu sekundy?

(5) Rozważmy 7-znakowe hasło składające się z małych liter alfabetu angielskiego — hasło to może przyjąć jedną z 26^7 postaci. Tak się niedobrze stało, że pewien intruz uzyskał dostęp do pliku zawierającego haszowane hasła wszystkich użytkowników; intruz ten dysponuje programem wykonującym 1000 prób „dopasowania" hasła w ciągu sekundy. Jak często użytkownik powinien zmieniać swe (losowo wybierane) hasło, by prawdopodobieństwo powodzenia intruza nie przekroczyło 10^{-3}? (Po każdej zmianie hasła intruz uruchamia swój program na nowo).

(6) Zmodyfikuj protokół „wyzwanie-odpowiedź" dla weryfikacji współdzielonego (tajnego) klucza tak, by obydwoje uczestników interakcji mogło wykorzystywać znaczniki czasowe.

(7) Dwaj użytkownicy *A* i *B* znają nawzajem swe klucze publiczne, otrzymane z *TA*. Zaprojektuj protokół typu „wyzwanie-odpowiedź", umożliwiający tym użytkownikom wzajemne uwierzytelnienie. Rozważ dwa przypadki:
   (a) gdy klucze publiczne używane są do szyfrowania,
   (b) gdy klucze publiczne używane są do uwierzytelniania.

(8) Udowodnij zupełność i rzetelność protokołu identyfikacyjnego Fiata-Shamira. Napisz symulator interakcji tego protokołu i oceń jego efektywność.

(9) Udowadniający *P* i weryfikator *V* używają od pewnego czasu protokołu Fiata-Shamira do celów identyfikacji. Intruz Oskar, podsłuchujący ich dialogi i kolekcjonujący zapisy interakcji, zauważył, że wartości (binarnych) wyzwań wybierane przez *V* nie mają rozkładu jednostajnego: wyzwania o wartości 0 generowane są z prawdopodobieństwem ε, wyzwania o wartości 1 — z prawdopodobieństwem $1 - \varepsilon$, gdzie $\varepsilon < 0{,}5$. Oskar zamierza wcielić się w rolę *P* (Peggy) i zdaje sobie sprawę z faktu, że w tym celu musiałby z wyprzedzeniem trafnie odgadnąć najbliższą wartość wyzwania generowanego przez *V*. W tym celu rozważa on dwie następujące strategie:
   (a) zakładanie wartości 0 i 1 z takim samym rozkładem prawdopodobieństwa, z jakim statystycznie są one generowane,
   (b) konsekwentne zakładanie, że wyzwanie będzie miało wartość 1.
   (c) Jakie prawdopodobieństwo powodzenia ma Oskar w każdym z obydwu przypadków? Która ze strategii jest lepsza? Czy jakaś inna strategia byłaby dla Oskara jeszcze lepsza?

(10) Oskar dysponuje zapisami interakcji prowadzonych między *P* i *V* w procesie identyfikowania za pomocą protokołu Fiata-Shamira. Analizując te zapisy, Oskar zauważył dwie pozycje (u_1, b_1, v_1) i (u_2, b_2, v_2) takie, że $u_1 = u_2$ i jednocześnie $b_1 \neq b_2$. Jakie jest prawdopodobieństwo wystąpienia takiej sytuacji? Czy ułatwia ona Oskarowi przełamanie protokołu?

(11) Udowodnij zupełność i rzetelność protokołu Feigego-Fiata-Shamira. Zaprojektuj odpowiadający mu symulator interakcji i przedyskutuj zależność jego efektywności od przyjętej długości (liczby bitów) wyzwania ℓ.

(12) Analizując przebieg protokołu Feigego-Fiata-Shamira, Oskar zauważył, że *V* pierwsze swe wyzwanie wybiera zgodnie z protokołem, czyli losowo i jednorodnie ze zbioru $\{0,1\}^{\ell}$, lecz już w następnych rundach wyzwania generowane są na zasadzie swoistego recyklingu: jeżeli mianowicie w rundzie o numerze $i - 1$ wyzwanie miało postać $b_{i-1} = \{\alpha_1, \ldots, \alpha_\ell\}$, to w następnej rundzie ma ono postać $b_i = \{\alpha_2, \ldots, \alpha_\ell, \alpha_{\ell+1}\}$, gdzie bit $\alpha_{\ell+1}$ generowany jest losowo z równym prawdopodobieństwem wystąpienia wartości 0 i 1. Jakie jest prawdopodobieństwo wystąpienia fałszywej akceptacji w tych warunkach?

(13) Dokonaj konwersji protokołów FFS i GQ na odpowiadające im schematy podpisu elektronicznego. Przedyskutuj bezpieczeństwo stworzonych schematów.

(14) Zmodyfikuj schemat identyfikacji Schnorra tak, by wszelkie obliczenia prowadzone były w ciele Galois $GF(2^{521})$, a q było liczbą pierwszą Mersenne'a równą $2^{521} - 1$. Przedyskutuj efektywność zmodyfikowanego schematu.

(15) Rozważ schemat identyfikacyjny Schnorra z binarnymi wartościami wyzwań $b \in_R \{0,1\}$. Udowodnij jego zupełność i rzetelność w tej postaci; zaprojektuj stymulator interakcji dla tego protokołu i przedyskutuj jego efektywność.

(16) Peggy, stosująca schemat Okamoto, tworzy swój prywatny klucz w ten sposób, iż losowo wybiera wartość s_1 i tę samą wartość przypisuje s_2. Przedyskutuj konsekwencje takiego postępowania; czy protokół będzie nadal bezpieczny, jeżeli intruz zorientuje się, że zawsze $s_1 = s_2$?

(17) Powróćmy do schematu Okamoto i załóżmy, że wybierane przez *TA* wartości α_1 i α_2 są konsekwentnie identyczne ($\alpha_1 = \alpha_2$). Czy schemat nadal pozostaje bezpieczny? Uzasadnij odpowiedź.