Dlaczego niepotrzebne nam hasło do skrzynki mailowej?

Michał ZAJĄC[*]

Zapewne zdecydowana większość Czytelników ma skrzynkę poczty elektronicznej. Dostęp do takiej skrzynki uzyskuje się przez podanie w specjalnym formularzu na stronie internetowej nazwy użytkownika i hasła. Te dane są następnie wysyłane w zaszyfrowanej formie do serwera pocztowego, który porównuje je z umieszczonymi na nim wzorcami. Tak, w dużym uproszczeniu, wygląda proces weryfikacji użytkownika na serwerze.

Ściśle rzecz biorąc, do serwera wysyłane jest nie hasło, ale wartość funkcji skrótu obliczona na jego podstawie.

Czy można wyobrazić sobie inną metodę uzyskiwania dostępu do skrzynki pocztowej? Taką, by nie trzeba było podawać hasła? W końcu stanowi to pewne zagrożenie dla naszego bezpieczeństwa: jeśli ktoś przechwyci, w ten czy inny sposób, nasze hasło i nazwę użytkownika, to uzyska nieograniczony dostęp do naszych zasobów – listów, dokumentów, historii przejrzanych stron – czego zapewne sobie nie życzymy.

Dowody z wiedzą zerową

Z pomocą przychodzą wtedy tzw. dowody z wiedzą zerową. Są to protokoły pozwalające na potwierdzenie faktu znajomości pewnej informacji bez ujawniania jej. Dla lepszego zilustrowania tego typu protokołów rozpatrzmy następujący przykład weryfikacji tożsamości w banku:

Przykład 1. W protokole tym będzie brało udział dwóch „graczy” – pierwszy będzie chciał potwierdzić swoją tożsamość i oznaczany będzie literą P (od angielskiego słowa *Prover*; gracz ten jest nazywany także klientem), drugi będzie weryfikował informacje przesyłane przez gracza P i oznaczany będzie literą V (od angielskiego słowa *Verifier*; inna nazwa tego gracza to weryfikator).

W protokole wykorzystywane są dwa grafy o n wierzchołkach, G_0 i G_1. Gracz P będzie starał się udowodnić, że są one izomorficzne. Należy tu nadmienić, że stwierdzenie, czy dane dwa grafy są izomorficzne, jest w ogólnym przypadku problemem trudnym obliczeniowo i nie jest znane jego rozwiązanie działające w czasie wielomianowym.

Grafy G_0 i G_1 są izomorficzne, jeśli istnieje bijekcja π przeprowadzająca wierzchołki grafu G_0 na wierzchołki grafu G_1 zachowująca strukturę krawędzi. Dokładniej, jeśli wierzchołki każdego z grafów ponumerujemy od 1 do n, to szukana bijekcja π jest permutacją zbioru $\{1, 2, \ldots, n\}$ o następującej własności: jeśli $u, v \in G_0$ są połączone krawędzią, to $\pi(u), \pi(v) \in G_1$ również są połączone krawędzią, i odwrotnie, jeśli $u', v' \in G_1$ są połączone krawędzią, to $\pi^{-1}(u'), \pi^{-1}(v')$ również. Piszemy wówczas, że $\pi(G_0) = G_1$.

Ale jaki związek ma dowodzenie izomorficzności grafów z uwierzytelnianiem? Możemy przyjąć, że w bazie danych banku znajduje się informacja z imieniem i nazwiskiem gracza P oraz grafami G_0 i G_1, znanymi publicznie, podanymi przez gracza P podczas zakładania konta. Gracz P, chcąc przekonać bank, że on to on, przedstawia się, a następnie wykazuje, że jest osobą, która potrafi udowodnić izomorficzność grafów G_0 i G_1. Mógłby to, oczywiście, zrobić, zdradzając permutację π, jednak po takim zabiegu nic nie stałoby na przeszkodzie, by nieuczciwy gracz V^* po poznaniu permutacji podszył się pod klienta P, wypłacając w jego imieniu dużą ilość gotówki. Zamiast tego gracze P i V przeprowadzą między sobą k razy następujący protokół (oznaczmy go jako $\mathcal{P}$):

1. Klient P wybiera losową permutację φ zbioru n-elementowego i przesyła do weryfikatora V graf $H = \varphi(G_0)$.
2. Weryfikator V przysyła graczowi P losowy bit $b \in \{0, 1\}$.
3. Jeśli otrzymany przez gracza P bit ma wartość 0, to wybiera on nową permutację $\psi = \varphi^{-1}$, a w przeciwnym przypadku $\psi = \pi\varphi^{-1}$. Gracz P wysyła następnie permutację ψ do weryfikatora V.
4. Weryfikator V sprawdza teraz, czy $\psi(H) = G_b$. Jeśli tak, to uznaje, że gracz P pomyślnie przeszedł proces weryfikacji. Jeśli nie, odrzuca jego próbę i przerywa protokół.

Od takiego protokołu będziemy wymagać trzech rzeczy. Po pierwsze, protokół przeprowadzony przez uczciwego gracza P musi zostać zaakceptowany. Po drugie, próba udowodnienia nieprawdziwego twierdzenia (w tym przypadku – próba wykazania, że dwa nieizomorficzne grafy są izomorficzne) powinna

[*]Instytut Informatyki, Uniwersytet Warszawski

powodować zatrzymanie i brak akceptacji protokołu (z prawdopodobieństwem bliskim 1). Podobnie będzie z próbą udowodnienia prawdziwego twierdzenia przy braku znajomości przez gracza permutacji wyznaczającej izomorfizm. Trzecią własnością jest wiedza zerowa, tj. weryfikator V po wykonaniu (nawet wielokrotnym) protokołu $\mathcal{P}$ nie może podszyć się pod gracza P.

Na początku wykażemy, że jeśli klient P jest uczciwy, to weryfikator V zaakceptuje jego próbę uwierzytelnienia się. Otóż jeżeli bit wysłany przez weryfikatora V w drugim kroku protokołu miał wartość 0, to uczciwy klient P odpowiedział na niego permutacją φ^{-1}, co daje $\psi(H) = \varphi^{-1}(H) = \varphi^{-1}\varphi(G_0) = G_0$. Załóżmy teraz, że przesłany bit miał wartość 1. Klient P wysyła wtedy permutację $\psi = \pi\varphi^{-1}$, więc $\psi(H) = \pi\varphi^{-1}(H) = \pi\varphi^{-1}\varphi(G_0) = \pi(G_0) = G_1$. Zatem w obu przypadkach $\psi(H) = G_b$ i weryfikator V akceptuje wykonanie protokołu.

Wykażemy teraz, dlaczego protokół $\mathcal{P}$ zostanie przerwany przez weryfikatora V, jeśli oszust P^* będzie próbował przekonać go do nieprawdziwego twierdzenia. Przyjmijmy więc, że klient P^* został pozytywnie zweryfikowany przez V, podczas gdy grafy G_0 i G_1 nie są izomorficzne. Załóżmy, że w każdym z k wykonań protokołu oszust umiał odpowiedzieć poprawnie, zarówno wtedy, gdy zadany w drugim kroku bit miał wartość 0, jak i 1. Niech ψ_0 i ψ_1 oznaczają funkcje ψ otrzymane w trzecim kroku algorytmu, gdy zadany przez weryfikatora V bit miał odpowiednio wartość 0 i 1. Mamy wtedy $\psi_0(H) = G_0$ i $\psi_1(H) = G_1$ oraz $G_1 = \psi_1\psi_0^{-1}(G_0)$. Czyli, znając ψ_0 i ψ_1, oszust P^* jest w stanie wskazać izomorfizm pomiędzy grafami G_0 i G_1, co nie jest możliwe. Mogło oczywiście być tak, że w każdym z k wykonań protokołu gracz P^* poprawnie zgadł wartość bitu, o jaki zostanie zapytany przez sprawdzającego V (łatwo wykazać, że stworzenie takiego grafu H, iż oszust P^* jest w stanie poprawnie odpowiedzieć na dokładnie jedną wartość bitu b, nie stanowi problemu, jednak wtedy szansa na powodzenie tej iteracji protokołu wynosi dokładnie $\frac{1}{2}$, gdyż takie jest prawdopodobieństwo, że weryfikator V wylosuje bit b o ustalonej wartości). Jednak to zdarzenie ma prawdopodobieństwo równe $\frac{1}{2^k}$, a więc dla odpowiednio dużego k jest zaniedbywalnie małe.

Powyższe rozumowanie wykazuje również, iż nieuczciwy gracz P^*, który próbuje przekonać weryfikatora V do tego, że zna permutację κ, taką że $\kappa(G_0) = G_1$, mimo braku tej wiedzy, też nie zakończy protokołu sukcesem (tj. zaakceptowaniem przez weryfikatora V).

Aby wykazać, że weryfikator V w podanym protokole nie dowiaduje się niczego istotnego o sekrecie gracza P, tj. mimo wielokrotnego przeprowadzenia protokołu nie jest w stanie podszyć się pod niego, wprowadza się pojęcie symulatora działającego w czasie wielomianowym (z prawdopodobieństwem równym $1 - \varepsilon$, gdzie $\varepsilon > 0$ jest zaniedbywalnie małe). Symulator M wykonuje protokół z weryfikatorem V, „podszywając się" pod P (M nie ma dostępu do prywatnych informacji gracza P, a jedynie zna jego odpowiedzi na pewną liczbę zapytań w procesie weryfikacji). Protokół $\mathcal{P}$ jest bezpieczny, gdy weryfikator V nie jest w stanie stwierdzić, czy operacje, które wykonuje, przeprowadza z prawdziwym graczem P, czy z symulatorem M (prawdopodobieństwo, że odpowie dobrze, jest równe $\frac{1}{2} + \delta$, gdzie $\delta > 0$ jest zaniedbywalnie małe). Formalny opis symulatora M i przeprowadzenie pełnego rozumowania wymaga wprowadzenia dużej ilości nowej terminologii i zostanie w tym artykule pominięte. Zainteresowani Czytelnicy są zachęcani do przeczytania literatury podanej na końcu artykułu.

Inny przykład protokołu z wiedzą zerową (opartego nie na izomorfizmie grafów, ale na znajdowaniu w nich cyklu Hamiltona) znajdą Czytelnicy w artykule Krzysztofa Kulewskiego z *Delty* 4/2005.

Σ-protokoły

Zaprezentowany powyżej protokół musiał być wykonywany wiele razy pod rząd, by można go było użyć do weryfikacji, ponieważ poprawne „przejście" jego pojedynczej iteracji przez nieuprawnionego gracza ma szansę powodzenia $\frac{1}{2}$.

Aby ograniczyć liczbę iteracji do minimum (czytaj: do jednej), wprowadzono protokoły, w których gracz weryfikujący V przesyła w drugim kroku zamiast pojedynczego bitu b ciąg $e \in \{0,1\}^k$ zwany *wyzwaniem*, który traktujemy jako liczbę naturalną zapisaną binarnie. Ilustracją tej metody może być protokół zaprezentowany poniżej:

Przykład 2. Niech p i q będą liczbami pierwszymi, takimi że q jest dzielnikiem $p-1$, i niech g będzie elementem rzędu q w grupie $\mathbb{Z}_p^*$. Ponadto niech $h = g^w \bmod p$. W zaprezentowanym protokole wartości p, q, g i h są publiczne, a gracz P będzie przekonywać gracza V, że zna wartość w. Zrobi to w następujący sposób:

1. Gracz P wybiera losowo liczbę r z ciała $\mathbb{Z}_q$, a następnie wysyła graczowi V liczbę $a = g^r \bmod p$.
2. Weryfikator V wybiera losowe wyzwanie $e \in \{0,1\}^k$ i wysyła je graczowi P.
3. Gracz P oblicza $z = ew + r$ i wysyła tę wartość weryfikatorowi.
4. Weryfikator sprawdza, czy $g^z \equiv ah^e \pmod p$. Jeśli tak, akceptuje protokół, a jeśli nie, przerywa go.

Teraz już widać, czemu Σ-protokoły zawdzięczają swoją nazwę. Jeśli wiadomości przesyłane pomiędzy graczami oznaczymy strzałkami, otrzymana łamana będzie przypominała grecką literę Σ.

Przede wszystkim zauważmy, że uczciwy gracz P zostanie zawsze zaakceptowany przez gracza V. Jak łatwo sprawdzić, $g^z = g^{ew+r} = g^r \cdot (g^w)^e \equiv ah^e \pmod p$, co jest zgodne z ostatnim punktem protokołu.

Σ-protokoły mają jeszcze jedną interesującą własność: jeśli gracz P potrafi przy danym a odpowiedzieć na dwa różne wyzwania e i e', to potrafi znaleźć świadka w (a zatem odpowiedzieć na dowolne wyzwanie). Przyjmując, że odpowiedzią gracza P na wyzwanie e było z, a odpowiedzią na wyzwanie e' było z', mamy:

$$g^z \equiv ah^e \pmod p, \qquad g^{z'} \equiv ah^{e'} \pmod p.$$

Dzieląc jedno równanie przez drugie i mając na uwadze fakt, że $e - e' \neq 0$, otrzymamy

$$g^{z-z'} \equiv h^{e-e'} \pmod p, \qquad g^{\frac{z-z'}{e-e'}} \equiv h \pmod p.$$

W ten sposób uzyskaliśmy $w = \frac{z-z'}{e-e'} \bmod q$ (dzielenie rozumiemy tutaj jako mnożenie przez odwrotność modulo q). Zatem jeśli gracz P nie zna sekretu w, to jest w stanie odpowiedzieć poprawnie na co najwyżej jedno wyzwanie weryfikatora V (na jakie wyzwanie konkretnie będzie chciał odpowiadać, wybiera, tworząc wiadomość a). Prawdopodobieństwo, że gracz V wyśle mu wybrane przez niego wyzwanie e, wynosi $\frac{1}{2^k}$, a więc jest zaniedbywalnie małe.

Warto w tym miejscu zaznaczyć, że Σ-protokoły nie mają wspomnianej wcześniej własności bycia protokołem z wiedzą zerową. Są nimi, jeśli przyjmiemy dodatkowe założenie – o uczciwości weryfikatora V. Wydaje się, że jest ono niezbyt praktyczne, ale własność ta jest wystarczająca do budowy wielu bardziej złożonych konstrukcji, które mają zastosowanie w realnym świecie. Na przykład, można dzięki nim zbudować efektywny system obsługi płatności elektronicznych, który zapewnia taką samą anonimowość jak płacenie gotówką. Ale to już całkiem inna historia...

Rozwiązanie zadania M 1370.
Zauważmy, że dla $x \geqslant -1$ zachodzi

$$x \leqslant \frac{1}{3} + \frac{4}{3}x^3,$$

gdyż

$$\frac{4}{3}x^3 + \frac{1}{3} - x = \frac{1}{3}(4x^3 - 3x + 1) = \frac{1}{3}(4x(x^2-1) + x + 1) = \frac{1}{3}(x+1)(2x-1)^2 \geqslant 0.$$

Zatem

$$x_1 + \ldots + x_n \leqslant \leqslant \frac{n}{3} + \frac{4}{3}(x_1^3 + \ldots + x_n^3) = \frac{n}{3}.$$

Literatura

[1] M. Bellare, O. Goldreich, *On defining proofs of knowledge*, Crypto 92.
[2] I. Damgård, *On Σ-protocols*, `https://services.brics.dk/java/courseadmin/CPT/documents/getDocument/Sigma.pdf?d=33739`.
[3] I. Damgård, J. B. Nielsen, *Commitment schemes and zero-knowledge protocols*, `https://services.brics.dk/java/courseadmin/CPT/documents/getDocument/ComZK08.pdf?d=30199`.
[4] A. Fiat and A. Shamir, *How to prove yourself: practical solutions to the identification and signature problem*, Crypto 86.
[5] O. Goldreich, S. Micali, A. Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, FOCS 86, 174–187.
[6] C. Schnorr, *Efficient signature generation by smart cards*, J. Cryptology 4(3), 1991.