O weryfikacji protokołów kryptograficznych

Tomasz BRENGOS, Anna CICHOCKA, Hubert GROCHOWSKI, Konstanty JUNOSZA-SZANIAWSKI, Adam KOMOROWSKI, Agata PILITOWSKA

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska

Wyobraźmy sobie, że na ważnej kolacji spotykają się dyplomaci. Rzecz jasna, podczas takiego spotkania obowiązują pewne reguły – każdy wie, gdzie ma usiąść, każdy wie, co i kiedy może powiedzieć, każdy wie, jak zareagować na działanie innych dyplomatów, tak aby było to „zgodne z protokołem". Podobnie możemy wyobrażać sobie *protokół komunikacyjny* jako zbiór zasad i instrukcji dla każdej ze stron, które chcą się ze sobą porozumieć – taki protokół dyplomatyczny dla interlokutorów. W protokole komunikacyjnym występują różne strony komunikacji, nazywane *agentami* (może to być na przykład klient banku, serwer albo brygada wojskowa), które przesyłają między sobą różne *wiadomości* według ustalonego porządku. Kiedy komunikacja przebiega zgodnie z protokołem, strony dokładnie wiedzą, co się dzieje, o co chodzi i co należy zrobić. Często od protokołu komunikacyjnego będziemy oczekiwać zapewnienia odpowiedniego poziomu bezpieczeństwa – na przykład oprócz skutecznej komunikacji będzie on musiał zagwarantować, że część wiadomości będzie dostępna wyłącznie dla uprawnionych stron (dla pozostałych pozostanie ukryta). Taki protokół nazywamy *protokołem kryptograficznym*.

Naturalnym sposobem przekształcenia zwykłego protokołu komunikacyjnego w protokół kryptograficzny wydaje się *zaszyfrowanie wiadomości*, czyli przekształcenie jej w taki sposób, aby adresat mógł ją *odszyfrować* (tzn. odwrócić przekształcenie wykonane przez nadawcę) za pomocą odpowiedniej, innej wiadomości (nazywanej *kluczem*). Chcielibyśmy to zrobić tak, aby osoba postronna, która nie posiada klucza, nie potrafiła odczytać wiadomości. Przy tym podejściu pojawiają się co najmniej dwa problemy. Pierwszy z nich został przedstawiony na rysunku.

Zaszyfrowana wiadomość, która jest łatwa do odczytu. Rysunek pochodzi z [4]

Powyższy obrazek został zaszyfrowany za pomocą algorytmu szyfrującego uważanego powszechnie za bardzo silny, który dodatkowo wykorzystuje długi klucz. Mimo to bez trudu możemy odczytać, co znajdowało się na obrazku przed szyfrowaniem. Ten przykład pokazuje, że sam dobry szyfr lub długi klucz nie wystarczą – równie istotne jest ich właściwe użycie. Konkretniej, zastosowany w tym przypadku algorytm jest deterministyczny, co oznacza, że dla tych samych danych zawsze daje ten sam wynik. (Przeciwieństwem algorytmów deterministycznych są algorytmy losowe). Co więcej, powyższy obrazek przed szyfrowaniem składał się głównie z białego tła. Algorytm szyfrujący podzielił go na mniejsze fragmenty i zaszyfrował każdy z nich osobno. Ponieważ wiele fragmentów zawierało wyłącznie białe tło, a algorytm jest deterministyczny, to każdy taki fragment został zaszyfrowany w ten sam sposób. Z drugiej strony fragmenty zawierające choćby niewielką część liter były szyfrowane inaczej. W rezultacie po zaszyfrowaniu można zauważyć różnice między obszarami tła a literami, co pozwala na częściowe odtworzenie pierwotnego obrazu.

Drugim problemem z szyfrowaniem jest konieczność zapewnienia, by adresat posiadał klucz odpowiedni do odszyfrowania wiadomości – musi on odpowiadać kluczowi użytemu do jej zaszyfrowania. To wyzwanie nazywane jest problemem wymiany klucza i przez wiele lat stanowiło poważne wyzwanie dla kryptologów. Dawniej wystarczającym rozwiązaniem było fizyczne dostarczenie wspólnego (symetrycznego) klucza obu stronom komunikacji. Nietrudno jednak zauważyć, że metoda ta jest mało wygodna i czasochłonna. Z tego powodu podjęto wiele wysiłków w celu opracowania bardziej efektywnego podejścia. Jednym z najważniejszych rozwiązań tego problemu jest protokół Diffiego–Helmana [3], opracowany w 1976 roku przez Whitfielda Diffiego i Martina Hellmana. W tym protokole uczestniczą dwie strony – zwykle nazywane Alicją i Bobem. W podstawowej wersji protokołu, na samym początku wszystkim są znane dwa parametry – dostatecznie duża liczba pierwsza p oraz liczba naturalna g od 2 do $p-1$, która ma następującą własność:

$$\{g^k \bmod p : k \in \mathbb{N}\} = \{1, 2, \ldots, p-1\}.$$

Innymi słowy, kolejne potęgi liczby naturalnej g dają wszystkie możliwe, niezerowe reszty z dzielenia przez p. W języku teorii grup oznacza to, że liczba g jest *generatorem* grupy $\mathbb{Z}_p^*$.

Liczby p oraz g są parametrami *publicznymi*, czyli mogą je znać nawet strony nieuczestniczące w tej komunikacji. Następnie protokół przebiega zgodnie z następującym schematem:

1. Alicja losuje liczbę naturalną a i wysyła do Boba liczbę $g^a \bmod p$ (resztę z dzielenia g^a przez p);
2. Bob losuje liczbę naturalną b i wysyła do Alicji liczbę $g^b \bmod p$ (resztę z dzielenia g^b przez p).

Zauważmy, że w ten sposób zarówno Alicja, jak i Bob są w stanie ustalić wspólną liczbę, którą mogą wykorzystać jako klucz. Popatrzmy na ten problem z perspektywy Alicji. Otrzymała ona od Boba liczbę $g^b \bmod p$, ale nie zna liczby b. Jednakże Alicja może podnieść otrzymaną liczbę do wylosowanej przez siebie potęgi a. W ten sposób, korzystając z własności potęgowania i arytmetyki modularnej, otrzymujemy:

$$(g^b \bmod p)^a = g^{ba} \bmod p.$$

Analogicznie Bob może podnieść otrzymaną od Alicji liczbę $g^a \bmod p$ (nie zna on a) do swojej potęgi b, otrzymując:

$$(g^a \bmod p)^b = g^{ab} \bmod p = g^{ba} \bmod p.$$

W ten sposób zarówno Alicja, jak i Bob znają razem pewną wspólną liczbę. Okazuje się, że jeśli odpowiednio dobierzemy parametry publiczne g i p, to problem znalezienia właściwego wykładnika dla osoby postronnej, mającej tylko wiedzę publiczną, wydaje się obecnie trudny do rozwiązania. Problem ten znany jest jako *problem logarytmu dyskretnego*. Warto podkreślić, że gdy jako ludzkość będziemy dysponować odpowiednio złożonym komputerem kwantowym, problem ten na pewno będziemy mogli rozwiązać „szybko". To jednak temat na zupełnie inną opowieść...

Zadanie 1. Ile wynosi $13^7 \bmod 19$? Znajdź wszystkie liczby naturalne $a \in \mathbb{N}_+$ takie, że $13^a \bmod 19 = 7$.

Zadanie 2. Ile istnieje różnych generatorów grupy $\mathbb{Z}_{13}^*$?

Innym podejściem do szyfrowania jest wykorzystanie szyfrów *asymetrycznych*. W przeciwieństwie do szyfrów symetrycznych, wykorzystują one parę kluczy – klucz *publiczny* i klucz *prywatny*. Jeden z kluczy służy do szyfrowania wiadomości, a drugi do odszyfrowania. Jeśli upublicznimy klucz używany do szyfrowania, to każdy będzie mógł zaszyfrować wiadomość (dlatego nazywany jest kluczem publicznym) i wysłać ją do właściciela drugiego klucza z pary. Tylko właściciel klucza prywatnego będzie w stanie odszyfrować taką wiadomość (stąd nazwa klucz prywatny).

Aby lepiej zobrazować działanie tej techniki, można wyobrazić sobie bank, który udostępnia jeden klucz publiczny wszystkim swoim klientom. Każdy klient może użyć tego klucza do zaszyfrowania wiadomości do banku, a ponieważ tylko bank posiada odpowiadający klucz prywatny, tylko on jest w stanie odszyfrować te wiadomości. Dzięki temu bank nie musi przechowywać osobnych kluczy dla każdego klienta, co upraszcza proces komunikacji. Ten przykład ilustruje, w jaki sposób asymetryczne szyfrowanie pozwala na bezpieczne przesyłanie danych do jednego, centralnego odbiorcy.

Pojawia się jednak kolejny problem – skąd możemy mieć pewność, że klucz publiczny rzeczywiście pochodzi od adresata naszej wiadomości, a nie od osoby podszywającej się pod niego? Podobnie, skąd wiadomo, że wiadomość zaszyfrowana przy użyciu klucza publicznego faktycznie pochodzi od zamierzonego nadawcy? Rozwiązanie tej kwestii jest dobrze znane i powszechnie stosowane we współczesnym świecie – mowa o podpisie elektronicznym. Podpis weryfikuje się jednak za pomocą klucza publicznego, i znowu wracamy do poprzedniego problemu. Oczywiście istnieją skuteczne sposoby radzenia sobie z takimi trudnościami. Na przykład w systemach komputerowych klucze publiczne producentów są wbudowane na stałe, co umożliwia weryfikację autentyczności kolejnych kluczy publicznych. Sposoby przekazywania i weryfikowania autentyczności kluczy, wiadomości oraz tożsamości uczestników komunikacji stanowią *kluczową* istotę protokołów kryptograficznych.

Kiedy zatem możemy uznać, że protokół kryptograficzny jest bezpieczny? A właściwie, co to w ogóle znaczy, że jest bezpieczny? Bezpieczeństwa protokołu nie da się ocenić w oderwaniu od modelu zagrożeń, w którym funkcjonuje. Na przykład, jeśli założymy, że atakujący nie ma możliwości podsłuchiwania wiadomości ani dostępu do żadnych dodatkowych źródeł informacji, to każdy protokół można by uznać za bezpieczny. Jednak w bardziej realistycznym scenariuszu, gdy atakujący może podsłuchiwać komunikację lub obserwować działania użytkowników, ukrycie czegokolwiek przed nim staje się znacznie trudniejsze. Właśnie dlatego określenie modelu zagrożeń jest kluczowe dla oceny bezpieczeństwa protokołu kryptograficznego.

Jednym z najprostszych, a zarazem szeroko wykorzystywanych modeli tego typu jest model symboliczny, znany jako model Doleva–Yao. Zakłada on, że atakujący ma pełną kontrolę nad kanałem komunikacyjnym. Może on podsłuchiwać, przechwytywać, zatrzymywać oraz podmieniać dowolne przesyłane wiadomości. Atakujący posiada pełną wiedzę publiczną, ale nie ma dostępu do informacji znanych wyłącznie poszczególnym agentom, takich jak klucze prywatne.

Kluczowym założeniem modelu Doleva–Yao jest również to, że wykorzystywane prymitywy kryptograficzne (czyli szyfry, podpisy i inne podstawowe funkcje kryptograficzne, o których tutaj nie mówimy) są idealne. Oznacza to, że atakujący nie jest w stanie odszyfrować wiadomości bez posiadania odpowiedniego klucza – nie potrafi nic wywnioskować o kluczach prywatnych, nawet jeśli ma dostęp do niezaszyfrowanej i zaszyfrowanej wersji kilku wiadomości (rodzaj ataku znany jako „atak ze znanym tekstem jawnym").

Niektórzy mogą twierdzić, że takie założenia są nierealistyczne, ponieważ trudno wyobrazić sobie atakującego, który ma tak szerokie możliwości kontroli nad kanałem komunikacyjnym. Dlatego analiza bezpieczeństwa w modelu Doleva–Yao jest uznawana za dość rygorystyczną – jeśli protokół okaże się bezpieczny w tak wymagającym środowisku, istnieje szansa, że będzie bezpieczny również w bardziej realistycznych warunkach. Z drugiej strony w rzeczywistości prymitywy kryptograficzne mogą być słabe lub źle zaimplementowane, a klucze zbyt krótkie.

W takich sytuacjach, dysponując wystarczającą mocą obliczeniową, atakujący może złamać szyfr i odczytać zaszyfrowaną wiadomość. Zatem protokół, którego bezpieczeństwo udowodniono w modelu Doleva–Yao, jest bezpieczny tylko wtedy, gdy prymitywy kryptograficzne są rzeczywiście bezpieczne, a hasła odpowiednio długie i trudne do odgadnięcia.

W innym modelu komunikacji, nazywanym *modelem obliczeniowym*, zakłada się, że atakujący, dysponując odpowiednio dużą mocą obliczeniową, może złamać szyfr, zwłaszcza jeśli hasło lub klucz szyfrujący są relatywnie krótkie. W przeciwieństwie do modelu symbolicznego, model obliczeniowy jest bardziej realistyczny, ponieważ uwzględnia ograniczenia obliczeniowe atakujących oraz złożoność kryptograficznych prymitywów. Istotnym wnioskiem z tej różnicy jest to, że jeśli protokół jest bezpieczny w modelu obliczeniowym, to tym bardziej będzie bezpieczny w modelu symbolicznym. Implikacja w drugą stronę nie zawsze jest prawdziwa. Jednakże zaletą modelu symbolicznego w porównaniu z obliczeniowym jest łatwość sprawdzenia i formalnego dowodzenia własności protokołu. Dlatego, mimo mniejszego realizmu, wciąż jest on wykorzystywany. Skuteczny atak w modelu symbolicznym pozwala natychmiast odrzucić badany protokół, natomiast dowód bezpieczeństwa gwarantuje bezpieczeństwo strukturalne i pozwala skupić się na ocenie elementów składowych (algorytmach szyfrujących, hasłach, prymitywach kryptograficznych). Dzięki temu można oceniać protokół w sposób bardziej przejrzysty i uporządkowany.

Skupmy się teraz na modelu symbolicznym. Rozważmy przykład, w którym występują dwaj agenci – ponownie nazwijmy ich Alicją i Bobem. W tym przykładzie zakładamy, że wiedzą publiczną jest klucz publiczny `pk(skB)`, który jest powiązany z kluczem prywatnym Boba `skB` (klucz `skB` jest znany wyłącznie Bobowi).

Alicja chce przesłać do Boba wiadomość `m` i szyfruje ją przy użyciu klucza publicznego `pk(skB)`. Alicja może to zrobić, ponieważ zakładamy, że ten klucz jest dostępny publicznie, co umożliwia każdemu (w tym Alicji) zaszyfrowanie wiadomości przeznaczonej wyłącznie dla Boba. Zaszyfrowana wiadomość jest następnie przesyłana do Boba, który może ją odszyfrować dzięki swojemu kluczowi prywatnemu `skB`.

Omówiony protokół kryptograficzny można zapisać w ustrukturyzowany sposób, który przedstawiamy poniżej.

```
Wiedza:
    Publiczna: Alicja, Bob, pk(skB);
    Alicja: Alicja, Bob, m, pk(skB);
    Bob: Alicja, Bob, skB;
Akcje:
    Alicja -> Bob : { m } pk(skB).
```

Notacja ta jest znana w literaturze jako *notacja AnB* (skrót od *Alice and Bob*). Wyróżniamy w niej informację *początkową* każdego z agentów oraz to, jakie parametry są znane publicznie. Następnie wymieniamy kolejno wykonywane akcje w opisywanym protokole, zgodnie ze strukturą:

$$nadawca \to adresat : wiadomość.$$

Użyta w powyższym opisie składnia `{ m } pk(skB)` oznacza, że korzystając z szyfru asymetrycznego, wysyłamy wynik szyfrowania wiadomości `m` za pomocą klucza publicznego `pk(skB)`. Okazuje się, że taki protokół jest bezpieczny w modelu symbolicznym. Kłopot z nim polega jednak na tym, że zakłada on publiczną znajomość klucza `pk(skB)`. Co, jeśli to założenie nie byłoby spełnione? Prostym rozwiązaniem wydaje się przesłanie klucza `pk(skB)` przez Boba do Alicji. Ale Alicja najpierw musi w jakiś sposób zasygnalizować potrzebę komunikacji z Bobem. Może to na przykład zrobić, przesyłając do Boba swój identyfikator. Tak zmodyfikowany protokół, zapisany w notacji AnB, wygląda następująco.

```
Wiedza:
    Publiczna: Alicja, Bob;
    Alicja: Alicja, Bob, m;
    Bob: Alicja, Bob, skB;
Akcje:
    Alicja -> Bob : Alicja;
    Bob -> Alicja : pk(skB);
    Alicja -> Bob : { m } pk(skB).
```

Okazuje się, że ta prosta zmiana doprowadziła do fatalnych skutków. Zmodyfikowany protokół przestał być bezpieczny! Atakujący (nazwijmy go Oskarem, w skrócie `O`, zaś Alicję i Boba oznaczmy skrótowo jako `A` i `B`) może przechwycić wiadomość `A -> B : A`. W konsekwencji nie dotrze ona do Boba, za to Oskar będzie w stanie podszywać się pod Boba i wysłać Alicji swój własny klucz publiczny `pk(skO)` (dla którego będzie znał swój własny klucz prywatny). Alicja wyśle do Boba wiadomość `m` zaszyfrowaną kluczem publicznym `pk(skO)` (gdyż taki otrzymała), ale do Boba ta wiadomość nigdy nie dojdzie. Będzie tak dlatego, że Oskar ją przechwyci i odszyfruje kluczem prywatnym `skO`. Zwróćmy uwagę, że oba protokoły są bardzo podobne, a główna różnica między nimi polega na zmianie założenia, że `pk(skB)` nie jest znany na początku komunikacji jako parametr publiczny. Jak już zauważyliśmy, ta zmiana doprowadziła do utraty poufności wiadomości `m`. Zatem słabość tego protokołu nie ma nic wspólnego z siłą szyfrowania, a jedynie ze sposobem, w jaki się posługujemy szyfrowaniem.

Wróćmy jednak do pierwszego protokołu, przed modyfikacją. Okazuje się, że jeśli bezpieczeństwo naszego protokołu zdefiniujemy w inny sposób, to nie będzie on już wcale bezpieczny. Powiedzmy, że atakujący Oskar jest w stanie podejrzewać, jaka wiadomość może być zaszyfrowana. Dla przykładu może on myśleć, że Alicja donosi na niego do Boba, że jest

złośliwy. W tym celu Oskar może we własnym zakresie zaszyfrować wiadomość `Oskar Złośliwy` za pomocą klucza publicznego `pk(skB)` (zna on ten klucz, gdyż jest dostępny publicznie). Następnie bez trudu porówna swoją wiadomość z wiadomością przesyłaną przez Alicję do Boba. Jeśli będą one takie same, to znaczy, że zaszyfrowana wiadomość `m` to właśnie `Oskar Złośliwy`. O takim protokole powiemy wtedy, że nie posiada własności *słabej poufności* względem wiadomości `m`. Formalnie mówimy, że protokół ma taką własność względem pewnej wiadomości `abc`, jeśli jej wartość jest znana tylko osobom uprawnionym, zaś atakujący nie może nawet sprawdzić, czy wiadomość przyjmuje taką wartość, jaką podejrzewa. Nazwa ta może być dla niektórych myląca – i słusznie! Słaba poufność jest własnością silniejszą od zwykłej poufności (tajności) – protokół mający własność słabej poufności ma na pewno także własność poufności.

Oczywiście takich własności bezpieczeństwa, które możemy sprawdzać, jest o wiele więcej. Możemy też definiować je sami – w zależności od tego, do jakich celów chcemy taki protokół wykorzystać i co właściwie chcemy osiągnąć. Dla przykładu, jedną z takich własności jest *nierozróżnialność*. Upraszczając – jeżeli protokół posiada własność nierozróżnialności i wiemy, że tajna wiadomość może przyjąć jedną z dwóch wartości, to atakujący nie ma lepszej możliwości stwierdzenia, która wiadomość została wybrana, niż rzut monetą, czyli wybór losowy.

Sprawdzanie własności bezpieczeństwa protokołów kryptograficznych wydaje się zadaniem skomplikowanym. W związku z tym pojawia się naturalne pytanie – czy istnieją narzędzia komputerowe, których moglibyśmy użyć do sprawdzenia bezpieczeństwa protokołów kryptograficznych? Okazuje się, że tak! Jednym z takich narzędzi jest **Proverif** [2]. Program Bruno Blancheta jest przeznaczony właśnie do weryfikacji protokołów kryptograficznych w modelu symbolicznym. Jego działanie bazuje na zaprezentowaniu wejściowego protokołu za pomocą specjalnych formuł logicznych nazywanych *formułami Horna*. Formułą Horna o zmiennych logicznych nazwiemy każdą formułę logiczną, którą możemy zapisać w postaci alternatywy zmiennych logicznych z co najwyżej jedną niezanegowaną zmienną. Przykładami formuł Horna opartych na 3 zmiennych logicznych: x, y, z, są:

$$(\neg x) \vee y \vee (\neg z), \tag{1}$$

$$(\neg x) \vee (\neg z), \tag{2}$$

$$y. \tag{3}$$

Zauważmy, że każda z tych formuł może być zapisana równoważnie za pomocą następujących implikacji:

- formuła (1) jako $(x \wedge z) \Rightarrow y$,
- formuła (2) jako $(x \wedge z) \Rightarrow 0$,
- formuła (3) jako $1 \Rightarrow y$.

Innymi słowy, poprzednikiem implikacji jest zawsze koniunkcja niezanegowanych zmiennych logicznych (lub logiczna prawda, oznaczana jako 1), zaś następnikiem implikacji jest pojedyncza niezanegowana zmienna (lub logiczny fałsz, oznaczany jako 0). Dzięki tej interpretacji łatwiej jest zrozumieć wykorzystywanie tego konkretnego rodzaju formuł logicznych do modelowania protokołów kryptograficznych i własności bezpieczeństwa. Dla przykładu załóżmy, że atakujący chce poznać pewną wiadomość m. Niech $att(x)$ oznacza zmienną logiczną wyrażającą, że atakujący może poznać wiadomość x. Rozpatrzmy następującą formułę logiczną jako koniunkcję odpowiednich formuł Horna:

- $att(e)$ dla każdej wiadomości publicznej e oraz każdej wiadomości przesyłanej e,
- $att(f) \wedge att(x) \Rightarrow att(f(x))$ dla każdej funkcji f (co oznacza, że jeśli atakujący zna wartość x oraz funkcję f, to może poznać także wartość $f(x)$),
- $att(x) \Rightarrow att(\mathrm{resp}(x))$, gdzie $\mathrm{resp}(x)$ jest odpowiedzią dowolnego uczestnika protokołu na otrzymaną wiadomość x,
- $\neg att(m)$.

Jeśli atakujący może poznać wartość m, to z powyższej formuły będzie wynikać zależność $att(m) \wedge (\neg att(m))$, co prowadzi do sprzeczności.

Oprogramowanie Proverif może być trudne w obsłudze dla początkujących użytkowników, głównie ze względu na skomplikowany format wymaganych plików wejściowych. W rezultacie użytkownik musi opanować pełną składnię języka obsługiwanego przez to narzędzie. Aby złagodzić tę niedogodność, podjęliśmy badania w ramach projektu *Eksperymentalna platforma do automatycznej weryfikacji i walidacji algorytmów i protokołów kryptograficznych* (**EPW**), realizowanego przez Instytut Łączności (lider projektu), NASK oraz Politechnikę Warszawską [1]. Celem projektu było ułatwienie korzystania z narzędzi do formalnej analizy protokołów kryptograficznych, w tym Proverif.

Głównym wynikiem projektu EPW po stronie Wydziału Matematyki i Nauk Informacyjnych Politechniki Warszawskiej jest automatyczny program tłumaczący protokół zapisany w prostej, wspomnianej wcześniej, strukturze AnB do języka używanego przez Proverif. Dzięki temu program Proverif może stać się dużo bardziej dostępny i użyteczny nawet dla mniej zaawansowanych użytkowników.

Wszyscy autorzy tego artykułu prowadzą zajęcia na specjalności Matematyka w Cyberbezpieczeństwie na Wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej.

Literatura

[1] www.gov.pl/web/instytut-lacznosci/epw.

[2] Bruno Blanchet, "Modeling and verifying security protocols with the applied pi calculus and proverif", *Foundations and Trends® in Privacy and Security*, 1(1-2):1–135, 2016.

[3] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[4] Christof Paar and Jan Pelzl. *Understanding Cryptography – A Textbook for Students and Practitioners*. Springer, 2010.