Kryptografia i bezpieczeństwo danych
- szyfrowanie z uwierzytelnieniem

Sławomir Samolej
ssamolej.kia.prz.edu.pl
ssamolej@prz.edu.pl

Repetytorium: co umiemy do tej pory?

Poufność: bezpieczeństwo semantyczne na atak z wybranym tekstem jawnym (CPA attack)

- Takie szyfrowanie zabezpiecza tylko przed **podsłuchiwaniem**

Integralność:

- Odporność na fałszowanie z zastosowaniem ataku z wybraną wiadomością
- CBC-MAC, HMAC, PMAC, CW-MAC

Nowe widomości: poznamy schematy szyfrowania odporne na **fałszowanie**

- Pozwolą one na zapewnienie dla wiadomości zarówno poufności jak i integralności.

2

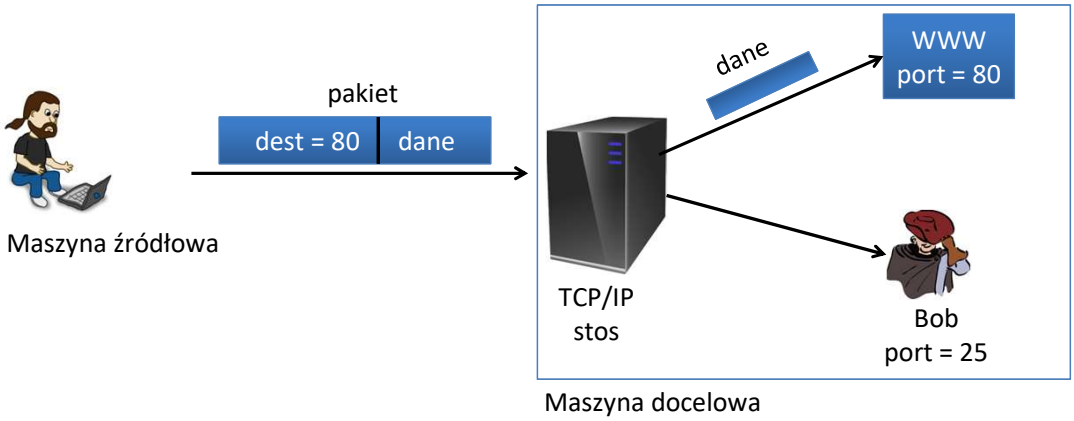
W pierwszej części wykładu omawialiśmy zagadnienia **poufności**. Zajmowaliśmy się szyfrowaniem wiadomości, tak aby były semantycznie bezpieczne na tzw. ataki z wybranym teksem jawnym (atakujący zna tekst jawny i jego szyfrogram, ale nie może wyliczyć z tego klucza <z bardzo dużym prawdopodobieństwem>). Takie szyfrowanie zabezpiecza jedynie przed **podsłuchiwaniem** widomości. Atakujący mają możliwość tylko odczytywania ruchu w sieci i próbują odgadnąć, jakie informacje są przesyłane. Nie próbują wprowadzać własnych pakietów lub zmieniać wybrane pakiety.

W drugiej części wykładu analizowaliśmy sposoby zachowania **integralności** wiadomości, przy czym sama wiadomość nie była utajniana. Chcieliśmy tylko mieć pewność, że wiadomość nie zostanie przekształcona w czasie jej przekazywania. Omawialiśmy zagadnienia kodów uwierzytelniających wiadomość (MAC – Message Authentication Codes). Poprawne rozwiązania, które omówiliśmy pozwalały na zapewnienie integralności wiadomości przy atakach z wybraną wiadomością. Potocznie rzecz ujmując, atakujący, nawet jeśli dysponował wskazanymi wiadomościami i ich MAC nie był w stanie wygenerować dla wiadomości innego poprawnego MAC oraz poprawnego MAC dla jakiejkolwiek innej wiadomości. W ramach tej części wykładu przedyskutowaliśmy szereg schematów kryptograficznych pozwalających na efektywne generowanie MAC.

Dalsza część wykładu będzie poświęcona projektowaniu schematów kryptograficznych odpornych na kombinację wyżej omówionych ataków. Atakujący może równocześnie próbować łamać szyfrogram i próbować fałszować część (lub całość) wiadomości. W konsekwencji odbiorca może odbierać poprawnie zaszyfrowaną wiadomość, o której treści decyduje atakujący.

Przykładowy atak z fałszowaniem (1)

TCP/IP: (na wysokim poziomie)

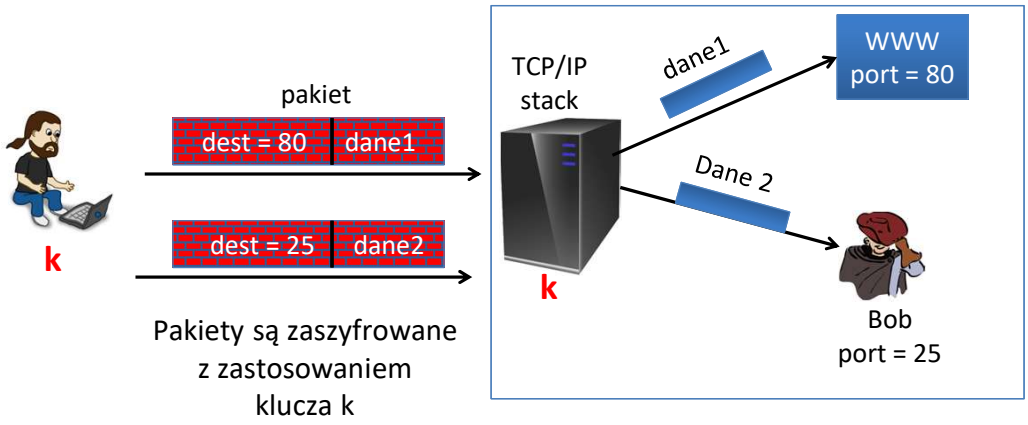


3

Przedyskutujemy kilka przykładów. Zacznijmy od atakującego, który może fałszować ruch sieciowy i w ten sposób złamać zaszyfrowaną wiadomość odporną na podsłuchiwanie (odporną na atak z wybranym tekstem jawnym). Pokażemy, że bez wprowadzenia mechanizmu zachowywania integralności samo szyfrowanie może być zniszczone. Rozważmy przykład z dziedziny zarządzania ruchem w sieci komputerowej. Będziemy rozważać uproszczone wysokopoziomowe własności protokołu TCP/IP nie w chodząc w szczegóły. Mamy dwie maszyny komunikujące się ze sobą. Użytkownik siedzi przy komputerze osobistym, drugą maszyną jest serwer. Serwer zawiera stos TCP/IP, który zajmuje się odbieraniem pakietów. Na podstawie informacji zawartych w pakiecie dane są dostarczane do właściwego miejsca. Możemy sobie wyobrazić dwa procesy nasłuchujące przychodzące pakiety. Jednym z nich jest serwer WWW, a drugim Bob. Serwer nasłuchuje na porcie 80, a użytkownik na 25. Stos TCP/IP analizuje przychodzące pakiety i wysyła dane skierowane na port 80 do serwera WWW, a te skierowane na port 25 – do Boba.

Przykładowy atak z fałszowaniem (2)

IPsec: (na wysokim poziomie)

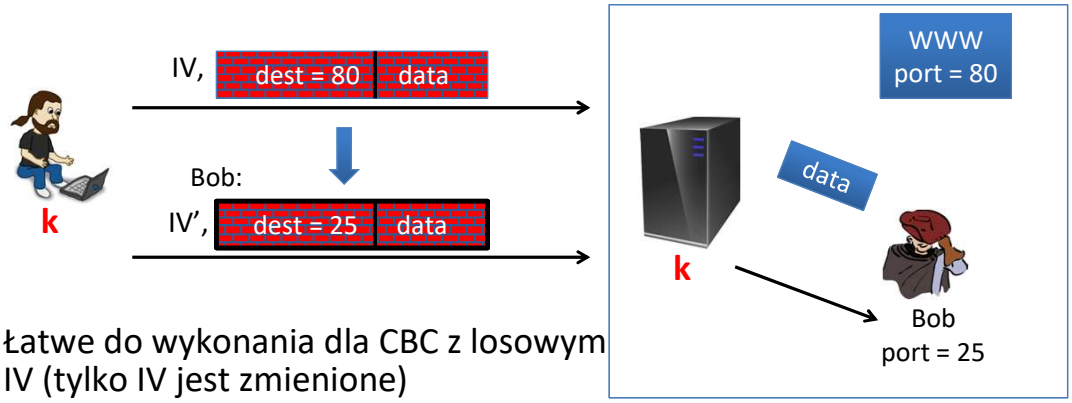


4

Dobrze znany protokół IPsec szyfruje pakiety przesyłane pomiędzy nadającym, a odbierającym. Nadawca i odbiorca mają wspólny tajny klucz. Pakiety wysłane przez nadawcę są szyfrowane z zastosowaniem tajnego klucza K. Stos TCP/IP u odbiorcy rozszyfrowuje pakiety, uzyskuje informację o porcie, na który ma przesłać dane i tam je wysyła. Można zauważyć, że w tym miejscu dane są już rozszyfrowane. Bez dodania to tego schematu mechanizmu zapewnienia integralności konstrukcja przestaje być bezpieczna.

Odczyt cudzych danych

Uwaga: atakujący otrzymuje odszyfrowaną zawartość szyfrogramu rozpoczynającą się od “dest=25”



Łatwe do wykonania dla CBC z losowym IV (tylko IV jest zmienione)

5

Załóżmy, że atakujący przechwytuje pakiet przeznaczony dla serwera WWW. Jest to zaszyfrowany pakiet z przeznaczeniem do portu 80. Atakujący może otrzymywać rozszyfrowane dane skierowane na port 25. Wystarczy, że serwer otrzyma informacje, że dane są skierowane właśnie do tego portu. Bob w naszym przypadku jest atakującym. Atak polega na przejęciu pakietu w zaszyfrowanej postaci i skierowanie go na port 25 (jak, zostanie to omówione za chwilę). Może on wtedy bez łamania klucza otrzymywać odszyfrowane dane skierowane pierwotnie do serwera.
Załóżmy, że dane są zaszyfrowane zgodnie ze schematem CBC z losowym IV (łańcuch zaszyfrowanych bloków). Wiemy że taki schemat jest semantycznie bezpieczny. Okazuje się, że konstrukcję można łatwo zaatakować. Jedynym elementem, który zostanie zmodyfikowany jest IV.

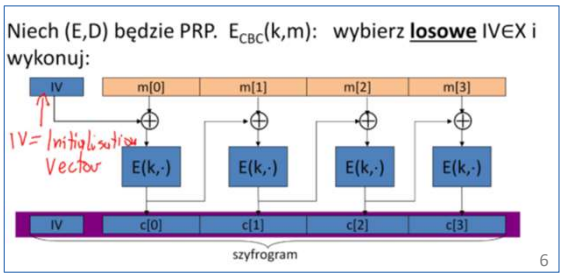
Jak sfałszować adres przeznaczenia pakietu?

Szyfrowania dokonano w schemacie CBC z losowym IV.

Jakie powinno być IV'? $m[0] = D(k, c[0]) \oplus IV$ = "dest=80…"

$$IV' = IV \oplus (\ldots 80 \ldots) \oplus (\ldots 25 \ldots)$$

CBC z losowym IV, przypomnienie:

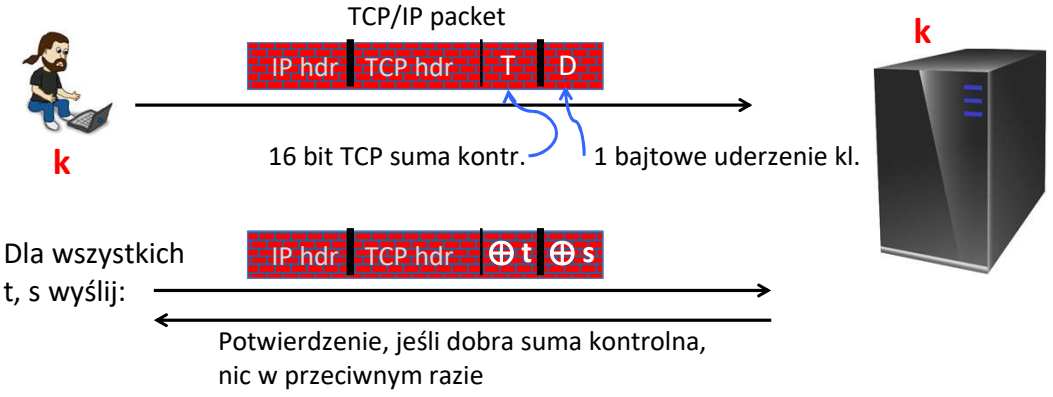


Atakujący przejął pakiet zaszyfrowany z zastosowaniem schematu CBC z losowym IV (por. prawy dolny róg slajdu). Wie, że docelowym portem jest 80, ale nie zna danych. Chce zmienić port docelowy z 80 na 25. Atak polega na odpowiednim zmienieniu IV. Przypomnijmy, że pierwszy blok wiadomości zaszyfrowanej według podanego schematu można odszyfrować obliczając wyrażenie: $m[0] = D(k, c[0]) \oplus IV$. I wiemy, że ten blok będzie miał postać "dest=80…". Okazuje się, że jeśli zmodyfikujemy odpowiednio IV, to możemy przekłamać pierwszy blok danych i skierować pakiety na inny port. Wystarczy, że $IV' = IV \oplus (\ldots 80 \ldots) \oplus (\ldots 25 \ldots)$ aby „usunąć" z pierwszego bloku w odpowiednim miejscu wartość 80 i podmienić ją na 25. Odbiorca (serwer) nieświadomy ataku na integralność zaszyfrowanej wiadomości skieruje odszyfrowane dane do innego portu. W szyfrogramie nie było wprowadzonego zabezpieczenia co do zachowania integralności wiadomości.

Szyfrowanie zapewniające tylko poufność nie jest odporna na ataki z możliwością modyfikacji szyfrogramu (fałszerstwo). Możliwość prostego wpłynięcia na zaszyfrowane dane jest w stanie złamać system je chroniący.

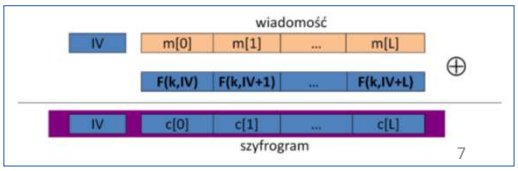
Atak stosujący tylko dostęp do sieci

Aplikacja zdalnego terminala:
każde uderzenie klawisza jest szyfrowane z zastosowaniem trybu CTR



{SumaKontr(hdr, D) = t ⊕ SumaKontr(hdr, D⊕s)}

⇒ można znaleźć D



Rozważmy kolejny atak z fałszerstwem, gdzie jedynym wymaganiem jest dostęp do sieci przez atakującego. Atakujący nie potrzebuje być składnikiem mechanizmu odszyfrowywania. W przykładzie mamy aplikację zdalnego terminala. Za każdym razem, kiedy użytkownik przyciska klawisz, zaszyfrowany numer klawisza jest przesyłany do serwera. Załóżmy, że szyfrowanie odbywa się w trybie licznikowym (por. prawy dolny róg slajdu). Użytkownik więc przesyła pakiet TCP/IP za każdym przyciśnięciem klawisza. D oznacza zaszyfrowany w trybie licznikowym kod klawisza. T jest sumą kontrolną sprawdzającą poprawność transmisji. Serwer, który otrzymuje pakiet ze złą sumą kontrolą odrzuca go. Nagłówek TCP wraz z sumą kontrolną i danymi są zaszyfrowane w trybie licznikowym. Atakujący chce się dowiedzieć, jaki był kod przyciśniętego klawisza.
Atakujący przejmuje pakiet, zachowuje i przesyła go dalej do serwera. Później dokonuje jego modyfikacji i wysyła do serwera. W zmodyfikowanych pakietach (wysyłanych później) dokonywany jest xor z sumą kontrolną i wartością t oraz xor z zaszyfrowanym kodem klawisza i wartością s. Atakujący dokonuje wysyłania takich zmodyfikowanych pakietów wiele razy. Należy pamiętać, że w trybie licznikowym wykonanie xor na szyfrogramie i jakiejś wartości da w rezultacie przy deszyfrowaniu wiadomość xor wartość. Serwer otrzymuje pakiet z zaszyfrowaną_sumą_kontrolną xor t i zaszyfrowanym_kodem_klawisza xor s. Serwer rozszyfrowuje wiadomość i otrzymuje sumę_kontrolną xor t i kod_klawisza xor s. Jeśli suma kontrolna się zgadza, odbiorca otrzymuje potwierdzenie. W przeciwnym wypadku pakiet jest odrzucany bez echa. Atakujący prowadzi statystykę dla jakich t i s otrzymuje potwierdzenie poprawności danych. Następuje analiza dla jakich modyfikacji s otrzymuje się określone t. Dysponując bazą wiedzy o wartościach s i t generujących poprawne pakiety można wydedukować wartość D. Warto wiedzieć, że dla algorytmów sum kontrolnych zastosowanych w protokole TCP taki atak raczej się nie powiedzie, ale gdy w rozwiązaniach stosuje się prostsze algorytmy generowania sum kontrolnych taki atak może zakończyć się sukcesem.
Być może lekko naciągany w stosunku do TCP, ale jest to dobry przykład nowego typu ataku – ataku z wybranym szyfrogramem. Atakujący modyfikuje szyfrogram i wysyła do odbiorcy w celu uzyskania informacji o wiadomości zaszyfrowanej.

Podsumowanie na tym etapie

Bezpieczeństwo na atak z wybranym tekstem jawnym (CPA security) nie gwarantuje bezpieczeństwa na tzw. aktywne ataki.

W konsekwencji użyteczne są dwie konstrukcje:

- Jeśli wiadomość wymaga zapewnienia integralności ale nie poufności, to należy zastosować **MAC**
- Jeśli wiadomość musi być poufna i zachować integralność należy zastosować tryby **szyfrowania z uwierzytelnieniem** (omówione dalej w tym wykładzie)

8

W tej część wykładu będziemy się spotykali z licznymi typami tak zwanych **aktywnych ataków**. Pokazane już ataki wykazały, że jeśli dysponujemy systemem kryptograficznym zapewniającym bezpieczeństwo tylko przeciw atakom z wybranym tekstem jawnym, to nie jest on odporny na ataki aktywne. Takie systemy kryptograficzne w zasadzie niczego nie gwarantują. Nie tylko nie zachowujemy integralność, to znaczy, że atakujący może sfałszować wiadomość. One też w zasadzie nie zapewniają poufności. W drugim przykładzie pokazaliśmy, że atakujący może wykorzystać odbiorcę do wydedukowania zaszyfrowanych danych.
W konsekwencji możemy zalecić dwa rodzaje postępowania: Jeśli mamy zapewnić tylko integralność a nie poufność, to możemy stosować **MAC**, Jeśli mamy zachować i poufność i integralność, to musimy się posłużyć **szyfrowaniem z uwierzytelnieniem**. Pół semestru nauki o bezużytecznych metodach bezpieczeństwa? Niekoniecznie, do tej pory poznaliśmy „klocki”, z których będziemy mogli zbudować naprawdę bezpieczny system przesyłania wiadomości.

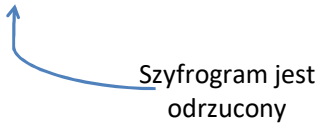
Cele

System szyfrowania z uwierzytelnieniem (E,D) jest szyfrem, gdzie

Jak zwykle: $E: K \times M \times N \longrightarrow C$

ale $D: K \times C \times N \longrightarrow M \cup \{\perp\}$

$\perp \notin M$



Bezpieczeństwo: system musi dostarczać

- Semantyczne bezpieczeństwo na atak z wybranym tekstem jawnym (CPA attack), i
- **Integralność szyfrogramu:**
  atakujący nie może stworzyć nowego szyfrogramu, który można prawidłowo odszyfrować

9

Zajmujemy się więc zdefiniowaniem szyfrowania z uwierzytelnieniem. Ma ono zapewnić bezpieczeństwo przesyłania informacji jeśli w systemie znajduje się aktywny atakujący (mogący przejąć i zmodyfikować wiadomość). W procesie szyfrowania tradycyjnie będą brały udział klucz, wiadomość i opcjonalnie „nonce". Proces deszyfrowania będzie się różnił. Oprócz otrzymania wiadomości M będzie zwracał nową wartość „sonda" (ang. bottom), która mówi, że szyfrogram jest niepoprawny i trzeba go zignorować. Jedynym wymaganiem jest, że sonda nie może należeć do przestrzeni wiadomości, czyli jest unikatowym symbolem wskazującym, że szyfrogram ma być odrzucony.
Jeśli chodzi o bezpieczeństwo takiego systemu, to musi on być bezpieczny semantycznie na atak z wybranym tekstem jawnym i zapewniać integralność. Druga właściwość oznacza, że jeśli atakujący wejdzie w posiadanie pewnej liczby szyfrogramów, to nie będzie on w stanie wygenerować dający się poprawnie odszyfrować inny szyfrogram. Sfałszowany szyfrogram odszyfruje się do innej wartości niż „sonda".

Szyfrowanie z uwierzytelnieniem

Definicja:

szyfr (E,D) zapewnia **szyfrowanie z uwierzytelnieniem** (ang. authenticated encryption (AE)) jeśli jest:

(1) bezpieczny semantycznie na atak z wybranym tekstem jawnym (CPA) i

(2) zapewnia integralność szyfrogramu

Zły przykład: CBC z losowym IV nie zapewnia szyfrowania z uwierzytelnieniem

- $D(k,\cdot)$ nigdy nie zwraca $\perp$, więc integralność nie jest zachowana

10

Można zdefiniować mechanizm szyfrowania z uwierzytelnieniem. Szyfr zapewnia takie szyfrowanie, jeśli zapewnia bezpieczeństwo semantyczna na atak z wybranym tekstem jawnym oraz zapewnia integralność szyfrogramu.
Rozważmy zły przykład. Konstrukcja CBC z losowym IV nie zapewnia szyfrowania z uwierzytelnieniem. Nie zwraca ona informacji o integralności wiadomości, stąd nie jest prawidłowa.

Konsekwencja 1: autentyczność

Atakujący nie może oszukać Boba w taki sposób, że Bob myśli, że dostał wiadomość od Alice (a wiadomość została spreparowana przez atakującego).

Nie można stworzyć weryfikowalny $c \notin \{ c_1, \ldots, c_q \}$

⇒ jeśli $D(k,c) \neq \perp$ Bob wie, że wiadomość jest od kogoś, kto zna klucz k
(ale wiadomość może być powtórką)

11

Zastosowanie szyfrowania z uwierzytelnieniem niesie za sobą kilka konsekwencji. Po pierwsze, atakujący, przy zastosowaniu tego szyfrowania, nie jest w stanie podszyć się pod nadawcę i przesłać odbiorcy widomość, którą nigdy nadawca nie wysłał. Atak rozpoczyna się o dialogu z nadawcą (Alice), poprzez wysłanie wiadomości i prośbę o ich zaszyfrowanie (atak CPA). Potem atakujący próbuje spreparować własny szyfrogram udający wiadomość od Alice. Ponieważ wiadomość została zaszyfrowana z zastosowaniem szyfrowania z uwierzytelnieniem, nie jest możliwe takie podszycie się pod nadawcę. Odbiorca wie, że otrzymał wiadomość tylko od tej osoby, która była w posiadaniu tajnego klucza k. Taka konstrukcja nie zabezpiecza przed powielaniem wiadomości. Atakujący może przechwycić szyfrogram i wielokrotnie go wysłać do odbiorcy. Jeśli wiadomość oznacza np. prośbę o przelanie 100 PLN na konto, to wysłanie kilku takich wiadomości bez wykrycia ataku może spowodować kilkukrotne przelanie pieniędzy. Okazuje się więc, że prawidłowe konstrukcje kryptograficzne powinny jeszcze zapewnić bezpieczeństwo przed atakami powtórzeniowymi. Samo stosowanie szyfrowania z uwierzytelnieniem nie zapewnia bezpieczeństwa na taki rodzaj ataku. Wrócimy do omawiania tego ataku na następnych slajdach.

Konsekwencja 2:

Szyfrowanie z uwierzytelnieniem $\Rightarrow$

Bezpieczeństwo na atak z wybranym szyfrogramem

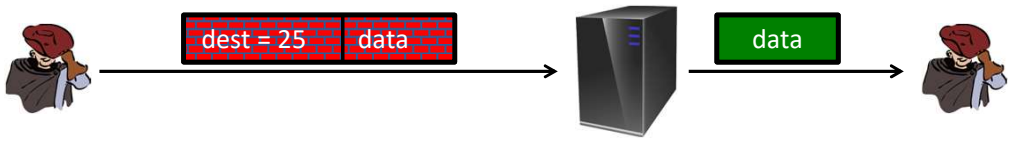
12

Drugą konsekwencją ze stosowania szyfrowania z uwierzytelnieniem jest uzyskanie bezpieczeństwa na atak z wybranym szyfrogramem (zagadnienie będzie omówione w dalszej części wykładu).

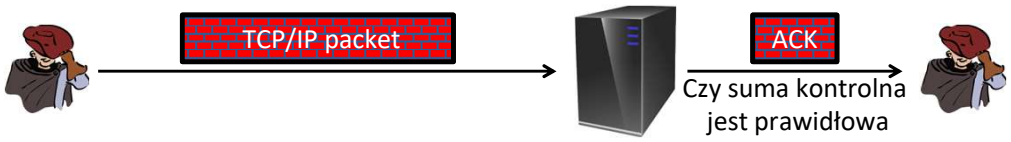
Przykłady ataku z wybranym szyfrogramem

Atakujący posiada szyfrogram c, który chce odszyfrować

- Często atakujący może oszukać serwer w calu odszyfrowania **pewnych** szyfrogramów (nie c)



- Często atakujący może nauczyć się częściowej informacji o danych szyfrowanych



13

Naszym celem jest odszyfrowanie szyfrogramu c. Często jesteśmy w stanie oszukać serwer w celu odszyfrowania pewnych wiadomości, ale nie tej, o którą nam chodzi (c). Na początku tego dokumentu dyskutowaliśmy sposób ataku polegający na podmianie pierwszego bloku szyfrogramu, w celu przekierowania danych przychodzących do serwera na inny port. To jest przykład możliwości odszyfrowania pewnych szyfrogramów, ale nie wszystkich. W drugim przypadku próbowaliśmy uzyskać informację przez wysłanie wielu zmodyfikowanych szyfrogramów do algorytmu deszyfrującego odsyłającego potwierdzenia. W naszym przypadku otrzymanie potwierdzenia oznaczało, że odszyfrowana wiadomość ma określoną wartość sumy kontrolnej, w przypadku braku potwierdzenia też mieliśmy informację – że nie zgadliśmy sumy kontrolnej dla określonej zaszyfrowanej wartości. To był kolejny scenariusz ataku z wybranym szyfrogramem. Atakujący manipulował szyfrogramem i uzyskiwał pewne informacje o danych zaszyfrowanych.

Bezpieczeństwo na atak z wybranym szyfrogramem

Możliwość przeciwnika: zarówno CPA (chosen plaintex attack) jak CCA (chosen ciphertext attack)

- Może otrzymać szyfrogram wskazanej wiadomości
- Może odszyfrować każdy szyfrogram inny niż ten, który chce odszyfrować
  (ostrożny model rzeczywistości)

Cel przeciwnika: Złamać semantyczne bezpieczeństwo

14

Kiedy rozważamy kryterium bezpieczeństwa na atak z wybranym szyfrogramem zakładamy dosyć duże możliwości atakującego. Po pierwsze może zaatakować atakiem z wybranym tekstem jawnym oraz atakiem z wybranym szyfrogramem. W praktyce dysponuje parami wiadomość-szyfrogram oraz może odszyfrować wybrane szyfrogramy, ale nie ten który chce zaatakować. W praktyce atakujący może oszukać system deszyfrujący, aby odesłał mu odszyfrowane wiadomości, ale nie tę konkretną, o którą mu chodzi.
Rozważana jest sytuacja, że atakujący dysponuje szyfrogramem i wchodzi w interakcję z systemem deszyfrującym, aby wydobyć informacje o danych zaszyfrowanych. Wysyła do niego wiele szyfrogramów do odszyfrowania, ale nie ten, na którym mu zależy.

Szyfrowanie z uwierzytelnieniem daje bezpieczeństwo na atak z wybranym szyfrogramem

Tw: Niech (E,D) będzie szyfrem zapewaniającym szyfrowanie z uwierzytelnieniem.

Wtedy (E,D) jest bezpieczne na atak z wybranym szyfrogramem!

Co z tego wynika?

Szyfrowanie z uwierzytelnieniem:

- Zapewnia poufność nawet jeśli **aktywny** atakujący może odszyfrować niektóre szyfrogramy

Ograniczenia:

- Nie zapobiega atakom powtórzeniowym
- Jeśli wyjawia więcej niż „sonda" jest podatne na ataki „bocznymi kanałami" (np. czasowe)

15

Istnieje udowodnione twierdzenie, które mówi, że jeśli dysponujemy szyfrem z uwierzytelnieniem, to jest on bezpieczny na atak z wybranym szyfrogramem. Jakie są z tego konsekwencje?
Nawet jeśli atakujący może odszyfrować pewną grupę szyfrogramów, to nie może złamać semantycznego bezpieczeństwa systemu. Pokazana konstrukcja nie gwarantuje bezpieczeństwa na ataki powtórzeniowe. W konstruowaniu systemów szyfrowania tego typu trzeba również bardzo uważać, aby zwracały one tylko podstawową wiedzę, czy dany szyfrogram jest, czy nie jest prawidłowy. Okazuje się, że jeśli system „zdradza" nieco więcej informacji (np. jest podatny na ataki czasowe), to może okazać się, że taki „wadliwy" system nie jest bezpieczny.

Szyfrowanie z uwierzytelnieniem – parę słów o historii

Szyfrowanie z uwierzytelnieniem: wprowadzone w 2000 r. [KY'00, BN'00]

Kryptograficzne API działające wcześniej: (np. MS-CAPI)

- Dostarczało API dla szyfrowania bezpiecznego na atak z wybranym tekstem jawnym (e.g. CBC with rand. IV)
- Dostarczało API dla MAC (e.g. HMAC)

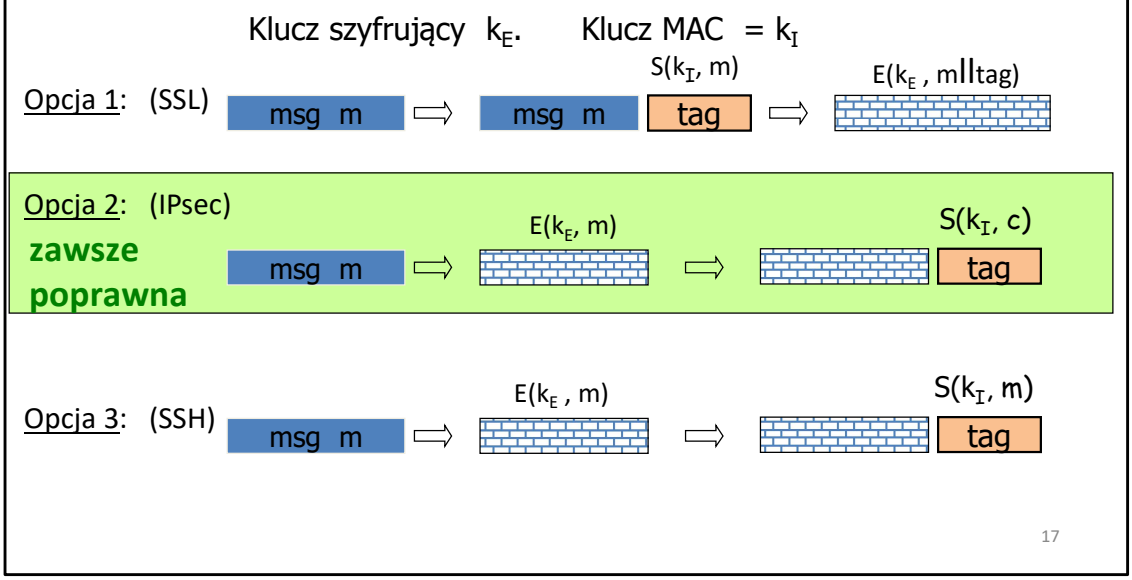
Każdy projekt musiał łączyć te dwa mechanizmy tak, jak to umieli programiści, bez jednoznacznie zdefiniowanych celów

- Niestety nie wszystkie „połączenia" miały cechy szyfrowania z uwierzytelnieniem …

16

Znamy już wymagania, co do szyfrowania z uwierzytelnieniem. Zastanowimy się, czy można zbudować taką konstrukcję w oparciu o poznane wcześniej mechanizmy. Zanim rozpoczniemy, opowiemy parę słów o historii. Szyfrowanie z uwierzytelnieniem zostało zaproponowane w roku 2000 w dwóch niezależnych publikacjach, do których będzie można znaleźć odniesienie na ostatnim slajdzie dotyczącym tego materiału. Zanim sformułowano szyfrowanie z uwierzytelnieniem, API służące do tworzenia oprogramowania kryptograficznego dostarczało rozdzielonych API do szyfrowania i do generowania MAC. Czyli inna funkcja służyła do szyfrowania a inna do generowania MAC. W konsekwencji twórcy oprogramowania samodzielnie konstruowali systemy, które miały zarówno utajniać komunikację jak i zapewnić uwierzytelnienie, z lepszym lub gorszym skutkiem. Część nie spełniała standardu szyfrowania z uwierzytelnieniem ponieważ nie było jeszcze wtedy wypracowanych sprawdzonych/udowodnionych schematów tworzenia takich konstrukcji kryptograficznych. Częstym błędem było niewłaściwe łączenie mechanizmów szyfrowania i uwierzytelniania.

Łączenie MAC i szyfrowania (CCA)



Na slajdzie pokazano trzy przykłady kombinacji szyfrowania i MAC. W każdym z nich stosuje się osobny klucz do szyfrowania i osobny klucz do generowania MAC. Klucze są niezależne od siebie i są generowane podczas inicjalizacji sesji szyfrowania/uwierzytelniania. Sposób generowania kluczy będzie pokazany później. Pierwszy przykład to protokół SSL. W protokole w pierwszym kroku oblicza się MAC dla wiadomości z zastosowaniem klucza k_I, a następnie wiadomość i dołączony do niej TAG szyfruje się z zastosowaniem algorytmu szyfrującego. W drugim rozwiązaniu (IPsec) najpierw szyfruje się wiadomość, a następnie z szyfrogramu oblicza się MAC i dołącza do szyfrogramu. Trzecie rozwiązanie stosowane w protokole SSH polega na zaszyfrowaniu wiadomości, a następnie dołączenie do niej MAC obliczonego z niezaszyfrowanej wiadomości. Pokazaliśmy trzy różne rozwiązania łączące szyfrowanie i MAC. Powstaje pytanie, które z nich jest bezpieczne.
Rozważmy rozwiązanie zastosowane w SSH (3 opcja). Tutaj MAC jest obliczany dla niezaszyfrowanej wiadomości i dołączany do szyfrogramu. Tu pojawia się problem, ponieważ MAC nie są konstruowane do zachowania poufności (do szyfrowania wiadomości). Dołączamy wtedy do szyfrogramu jakąś informację na temat niezaszyfrowanego tekstu. Chociaż więc konstrukcja SSH jest uznana za poprawną nie zaleca się w nowych rozwiązaniach stosowania takiej kombinacji szyfrowania i MAC. Rozważmy pozostałe dwie opcje. Warto wiedzieć, że rekomendowanym rozwiązaniem jest to zastosowane w IPsec. Zaletą rozwiązania jest to, że niezależnie jakie szyfrowania i jaką metodę generowania MAC zastosujemy, to zawsze otrzymamy rozwiązanie bezpieczne (szyfrowanie z uwierzytelnieniem). Dlaczego? Najpierw szyfrujemy wiadomość. Potem generujemy MAC dla szyfrogramu. W taki sposób zamykamy możliwość wygenerowania innego szyfrogramu, który byłby prawidłowy. Każda modyfikacja szyfrogramu będzie wykryta. Dla konstrukcji zastosowanej w SSL jest kilka patologicznych przykładów kiedy kombinacja szyfrowania odpornego na atak z wybranym tekstem (CPA) i MAC jest podatna na atak z wybranym szyfrogramem. Mogą się pojawić pewne złe interakcje pomiędzy schematem szyfrowania i algorytmem generującym MAC, które można zaatakować atakiem z wybranym szyfrogramem (CCA).
W nowych rozwiązaniach należy więc stosować najbezpieczniejszą kombinację: szyfrowanie wiadomości, a następnie obliczanie MAC dla szyfrogramu.

Stąd nie zawsze jest uznawana za poprawne rozwiązanie szyfrowania z uwierzytelnieniem.

Twierdzenia o szyfrowaniu z uwierzytelnieniem (A.E. – Authenticated Encryption)

Niech (E,D) będzie szyfrem bezpiecznym na atak z wybranym tekstem jawnym i (S,V) bezpiecznym MAC. Wtedy:

1. **Schemat Encrypt-then-MAC**: zawsze zapewnia szyfrowanie z uwierzytelnieniem
2. **Schemat MAC-then-encrypt**: może nie być bezpieczny, bo będzie podatny na atak z wybranym szyfrem

jednak: kiedy (E,D) jest losowym trybem licznikowym (rand-CTR) lub losowym trybem z łańcuchem bloków (rand-CBC), wtedy schemat M-then-E zapewnia szyfrowanie z uwierzytelnieniem

dla schematu rand-CTR, jednorazowy MAC jest wystarczający

przyspieszenie obliczeń

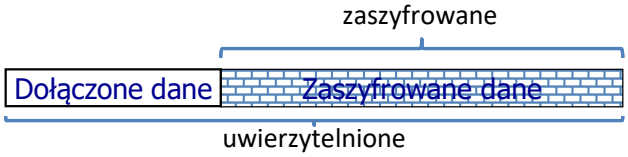
18

Są dowody matematyczne, które mają następujące konsekwencje:
1. Jeśli zastosujemy schemat szyfruj-potem-licz_MAC, to taka konstrukcja zawsze będzie bezpieczna.
2. Stosowanie schematu licz_MAC-potem-szyfruj może nie być w niektórych przypadkach bezpieczna. Jednak, kiedy zastosujemy dokładnie schematy rand-CRT lub rand-CBC jako mechanizmy szyfrujące, to wtedy konstrukcja jest uznawana za szyfrowanie z uwierzytelnieniem i jest bezpieczna.
Ponadto, w przypadku stosowania schematu rand-CRT, wystarczy dla zachowania bezpieczeństwa zastosowanie jednorazowego MAC (ten algorytm jest szybszy). Zalecaną obecnie konstrukcją jest ta pierwsza.

Standardy (na wysokim poziomie)

- **GCM**: tryb CTR potem zastosowanie CW-MAC
  (przyspieszone przez instrukcję procesorów Intela PCLMULQDQ)
- **CCM**: CBC-MAC potem szyfrowanie w trybie CTR (802.11i)
- **EAX**: szyfrowanie w trybie CTR potem CMAC

Wszystkie wspierają AEAD: (auth. enc. with associated data –
szyfrowanie z uwierzytelnieniem z dołączonymi danymi).
Opierają się na stosowaniu nonce



19

Kiedy pomysł na szyfrowanie z uwierzytelnieniem się upowszechnił, utworzono kilka standardów zawierających zalecenia, jak taką konstrukcję łączącą MAC i szyfrowanie realizować. Zostały one nawet ustandaryzowane. Wymienione zostaną trzy wybrane. Dwa pierwsze zostały zatwierdzone przez NIST (National Institute of Standards). Nazywane są „Galois Counter Mode" oraz „CBC Counter Mode". GCM stosuje szyfrowanie w trybie licznikowym, a potem obliczanie MAC z zastosowaniem algorytmu Cater'a-Wegman'a. Dla wiadomości obliczana jest najpierw funkcja hash, a potem jest liczony MAC. W architekturze procesorów Intela zaszyto nawet specjalną funkcję PCLMULQDQ do przyspieszenia obliczeń funkcji hash. Standard CCM stosuje najpierw obliczanie MAC w trybie CBC , a potem szyfrowanie w trybie licznikowym. Rozwiązanie obecnie nie jest rekomendowane, ale ponieważ stosuje tryb licznikowy do szyfrowania, to spełnia wymogi szyfru z uwierzytelnieniem. Warto wspomnieć, że konstrukcja CCM stosuje algorytm szyfrowania AES dla obliczania CBC MAC. A potem ten sam algorytm szyfrowania jest stosowany w trybie licznikowym. AES jest więc jednym wspólnym mechanizmem szyfrowania zastosowanym do generowania szyfrogramu i MAC. W konsekwencji implementacja konstrukcji może być relatywnie krótka (potrzebujemy realizację szyfrowania AES i jej umiejętne zastosowanie). Najprawdopodobniej codziennie wykorzystujemy to rozwiązania łącząc się do sieci bezprzewodowych.
Ostatni standard EAX znowu stosuje szyfrowanie w trybie licznikowym, a potem generowanie MAC z szyfrogramu (tym razem CMAC).
Wszystkie pokazane standardy stosują wartość nonce do szyfrowania. Ne ma tu więc wprowadzenia losowości. Jedynie, co musi być zapewnione, to, że para (klucz, nonce) ma być unikatowa w przesyłaniu zaszyfrowanych danych. Nonce może być licznikiem.
Wszystkie omówione schematy wspierają tzw. „szyfrowanie z uwierzytelnieniem z dołączonymi danymi". To jest rozszerzenie szyfrowania z uwierzytelnieniem stosowane w protokołach sieciowych. W takich rozwiązaniach nie chcemy, aby część wiadomości była zaszyfrowana. Część z danych ma pozostać jawna, ale cała wiadomość musi być uwierzytelniona (zapewniamy, że nie została zmodyfikowana). Dobrym przykładem takiej konstrukcji są pakiety internetowe. W pakiecie IP nie chcemy, żeby nagłówek był zaszyfrowany, bo muszą mieć do niego dostęp routery. Z drugiej strony chcemy mieć utajniony „ładunek" pakietu (to go szyfrujemy). Ostatecznie też chcemy, aby konstrukcja dawała gwarancję, że pakiet nie został zaatakowany, stąd stosujemy uwierzytelnienie (MAC) dla obu porcji danych. Jest to konstrukcja stosowana w bibliotece OpenSSL.

Przykład API (OpenSSL)

int **AES_GCM_Init**(AES_GCM_CTX *ain,
unsigned char ***nonce**, unsigned long noncelen,
unsigned char ***key**, unsigned int klen)

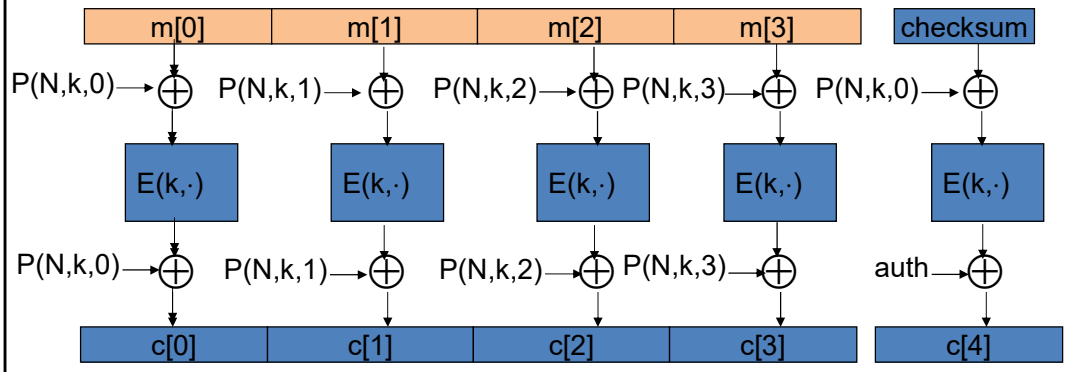
int **AES_GCM_EncryptUpdate**(AES_GCM_CTX *a,
unsigned char ***aad**, unsigned long aadlen,
unsigned char ***data**, unsigned long datalen,
unsigned char ***out**, unsigned long *outlen)

20

W bibliotece OpenSSL szyfrowanie z uwierzytelnieniem z zastosowaniem schematu GCM możliwe jest do wykonania przez wywołanie dwóch funkcji. Pierwsza z nich **AES_GCM_Init**() oczekuje na podanie wartości nonce i klucz. Funkcja realizująca szyfrowanie z uwierzytelnieniem z dołączonymi danymi przekazuje na wejście algorytmu dane, które nie mają być zaszyfrowane (aad), oraz dane, które mają być zaszyfrowane (data). W odpowiedzi zwracany jest przygotowany pakiet zawierający zaszyfrowane dane.

OCB: bezpośrednia konstrukcja z PRP

Bardziej wydajne szyfr. z uwierzyt.: one E() op. per block.



21

Po opublikowaniu prac definiujących szyfrowanie z uwierzytelnieniem, większość prac była skierowana na efektywne połączenie szyfrowania blokowego z generowaniem MAC w celu uzyskania schematów spełniających wymagania nowozdefiniowanego sposobu zabezpieczania danych. Po sformalizowaniu (ustaleniu ścisłych wymagań) szyfrowania z uwierzytelnieniem niektórzy badacze zaczęli się zastanawiać, czy można stworzyć schemat takiego szyfrowania w sposób alternatywny. Po zastanowieniu się jak wykonuje swoją pracę szyfrowanie w trybie licznikowym wraz z CMAC, okazuje się, że ten sam blok algorytmu szyfrowania najpierw służy do szyfrowania a potem do obliczania MAC. Powstaje więc pytanie, czy można zbudować konstrukcję spełniającą wymagania co do szyfrowania z uwierzytelnieniem wykonującą przekształcenia pojedynczego bloku danych raz (np. z zastosowaniem PRF). Okazuje się, że tak. Taka konstrukcja nazywa się OCB (ang. Offset Codeblock Mode) i jest szybsza od tych wcześniejszych bazujących na kombinacji szyfrowania i obliczania MAC. Pokazany schemat będzie omówiony w sposób ogólny. W górnej części slajdu widać dane do zaszyfrowania. Zaproponowanych schemat, jak łatwo zauważyć jest w naturalny sposób możliwy do wykonania współbieżnie. Każdy blok danych może być przetworzony osobno. Przetworzenie przez algorytm szyfrujący każdego z bloków danych odbywa się tylko raz. Na koniec wykonywany jest dodatkowy blok do wygenerowania tagu uwierzytelniającego. Zmniejszono więc w konstrukcji obciążenie obliczeniowe ze względu na osobne szyfrowanie i obliczanie MAC i wprowadzono możliwość zrównoleglenia obliczeń. Oprócz szyfrowania w schemacie musi zostać wykonana prosta funkcja P. Jej parametrami wejściowymi są N- nonce, k – klucz i licznik. Funkcja jest wykonywana dwukrotnie przed i po szyfrowaniu i z jej rezultatem wykonywany jest xor (na początku z blokiem widomości, a potem blokiem wychodzącym z algorytmu szyfrowania). Okazuje się, że taka konstrukcja oferuje możliwość uzyskania szyfrowania z uwierzytelnieniem w oparciu o szyfry blokowe. Udowodnione zostało dla niej bezpieczeństwo ze względu na atak z wybranym szyfrem i atak z wybranym tekstem jawnym (odnośnik do literatury znajdzie się na slajdzie końcowym). Niestety nie jest standardem ani je jest powszechnie stosowana, bo… objęta jest szeregiem patentów.

Wydajność: Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

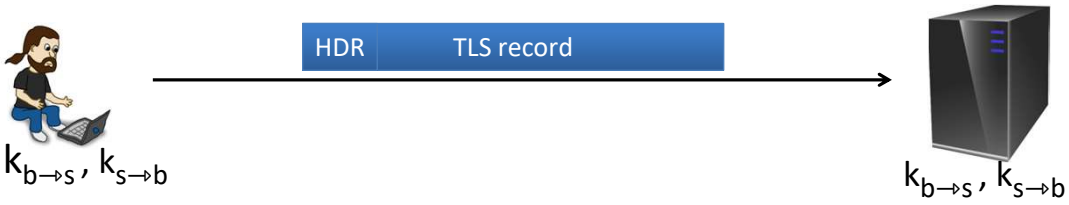
Szyfr	rozm. kodu	Szybkość (MB/sec)		
AES/GCM	duży **	108	AES/CTR	139
AES/CCM	mniejszy	61	AES/CBC	109
AES/EAX	mniejszy	61		
			AES/CMAC	109
AES/OCB		129*	HMAC/SHA1	147

* Wywnioskowane z rezulatów Ted Kravitz ** dla maszyn innych niż Intel

22

Na slajdzie zawarto kilka uwag dotyczących wydajności wybranych konstrukcji realizujących szyfrowanie z uwierzytelnieniem. Po prawej stronie slajdu zawarto dane wydajnościowe konstrukcji, których **nie powinno** się stosować. Można zauważyć, że konstrukcje AES/CBC i AES/CMAC mają podobne wydajności. Dla tych konstrukcji, które **powinno się stosować** (ustandaryzowanych). Najszybsza jest AES/GCM w której stosuje się szybkie obliczanie hash, a potem przetwarzanie w trybie licznikowym. Okazuje się też, że zastosowanie GCM w trybie licznikowym nie wprowadza wielkiego obciążenia. Konstrukcje CCM i EAX stosują blokowe szyfrowanie i blokowe obliczenia MAC. Stąd są prawie dwukrotnie wolniejsze od konstrukcji GCM. Konstrukcja OCB jest najszybsza, stosuje tylko jednokrotne użycie bloku szyfrującego dla jednego bloku wiadomości. Wydaje się więc, że rozwiązanie GCM jest najlepsze. Jego wadą jest duży rozmiar kodu, co może być ograniczeniem dla urządzeń dysponujących ograniczonymi zasobami. Z drugiej strony, jeśli stosuje się procesory Intela, to można posłużyć się implementacją zajmującą mniej pamięci (bo część algorytmu jest wbudowana w procesor). Podsumowując, jeśli rozmiar rozwiązania nie jest dla nas ograniczeniem, to warto stosować konstrukcję AES/GCM.
Uwagi, co do szyfrowania wiadomości na tym etapie naszej wiedzy można sformułować następująco. Jeśli chcemy mieć poprawnie zaszyfrowaną wiadomość, to musi ona być zaszyfrowana zgodnie ze schematem realizującym szyfrowanie z potwierdzeniem. Najlepiej jest to robić zgodnie z opublikowanymi standardami – jednym z trzech rozwiązań, które były przez nas omówione. Proszę nie implementować własnych schematów szyfrowania. Wiele współczesnych bibliotek kryptograficznych zawiera te schematy jako gotowe wywołania funkcji/metod. W dalszej części wykładu zastanowimy się, co się może stać, jeśli implementacją szyfrowania z uwierzytelnieniem zajmują się mniej doświadczeni programiści.

Protokół: TSL Record Protocol(TSL 1.2)



Jednokierunkowe klucze: $k_{b\to s}$ i $k_{s\to b}$

Szyfrowanie utrzymujące stan:

- Każda strona utrzymuje dwa 64-bitowe liczniki: $ctr_{b\to s}$, $ctr_{s\to b}$
- Liczniki są zerowane na początek sesji. Zwiększane o 1 dla każdego rekordu.
- Cel: zapobieganie atakom powtórzeniowym

23

Omówimy sobie rozwiązania szyfrowania z uwierzytelnieniem stosowane w rzeczywistości. Rozważmy więc TLS. Szyfrowanie danych w tym rozwiązaniu odbywa się z zastosowaniem „TLS Record Protocol” (rekord jest tu rozumiany jako porcja danych). Rekord danych jest poprzedzany nagłówkiem (HDR). Jego struktura zaraz będzie omówiona. Dane z nagłówkami służą do obustronnej komunikacji. W modelu komunikacji TLS największą porcję danych, jaką można przenieść w jednym rekordzie jest 16 kilobajtów. Jeśli ilość danych jest większa, to jest fragmentowana na mniejsze, max 16 kilobajtowe rekordy. TLS używa, tak zwanych jednokierunkowych kluczy. Osobny klucz służy do szyfrowania w kierunku od serwera do przeglądarki (browser), a osobny od przeglądarki do serwera. Jeden klucz służy więc wysyłaniu wiadomości, a drugi – odbieraniu. Zarówno serwer jak i przeglądarka znają oba klucze. Klucze są generowane z zastosowaniem tzw. „TLS Exchnge Key Protocol”, o którym będziemy mówili na następnych wykładach. Na chwilę obecną musimy założyć, że klucze zostały wygenerowane i (bezpiecznie) wymienione pomiędzy klientem a serwerem. Wymiana informacji pomiędzy serwerem a przeglądarką odbywa się z tzw. zachowaniem stanu. To znaczy, że na czas wymiany informacji jest utrzymywana sesja (stan) i informacja o przesyłaniu kolejnych pakietów w ramach sesji modyfikuje stan nadawcy i odbiorcy. Z punktu widzenia protokołu najważniejsze są dwa 65-bitowe liczniki utrzymywane po każdej ze stron wymieniających informacje. Jeden z liczników liczy rekordy wysłane do nadawcy, a drugi rekordy, które zostały odebrane. Liczniki są zerowane na początku nawiązania sesji, a potem zwiększane po każdym wysłaniu/odebraniu rekordu. Liczniki są zastosowane, żeby zapobiec atakom powtórzeniowym.

Rekord TLS: szyfrowanie (CBC AES-128, HMAC-SHA1)

$k_{b\to s} = (k_{mac}, k_{enc})$

Strona przeglądarki **enc($k_{b\to s}$, data, $ctr_{b\to s}$) :**

krok 1: tag ⟵ S(k_{mac} , [++$ctr_{b\to s}$ || header || data])

krok 2: pad [header || data || tag] do rozmiaru bloku AES

krok 3: szyfrowanie CBC z k_{enc} i nowym losowym IV

krok 4: dołączenie nagłówka

24

Record Protocol działa w następujący sposób. Konstrukcje jakie zastosowano, to AES-128 i HMAC-SHA-1. TLS stosuje schemat MAC-then-ENCRYPT, więc najpierw następuje obliczenie MAC, a potem zaszyfrowanie. Rozważmy sposób przesyłania danych od przeglądarki do serwera. Klucz przeglądarka-do-serwera ($k_{b\to s}$) składa się z 2 kluczy. Klucza do oblicznia MAC i klucza do szyfrowania. Są one ustalane w procesie inicjalizacji połączenia, który zostanie omówiony później. Ponieważ są osobne klucze: $k_{b\to s}$ (przeglądarka-do-serwera) i $k_{s\to b}$ (serwer-do-przeglądarki), w rezultacie w procesie przesyłania informacji biorą udział 4 klucze. Na rysunku pokazano, jak wygląda pakiet TLS. Nagłówek pakietu zwiera informację o typie pakietu, wersji protokołu oraz o długości pakietu on nie jest szyfrowany. Do procesu szyfrowania danych brany jest klucz $\mathbf{k_{b\to s}}$, dane i licznik ($\mathbf{ctr_{b\to s}}$). Najpierw obliczany jest MAC z danych połączonych z nagłówkiem oraz zwiększonym o jeden licznikiem. Wartość licznika (za wyjątkiem obliczonego z niego MAC) nigdy nie jest przesyłana. Serwer ma wiedzieć, jaka powinna być następna wartość licznika i zachowywać ją w wewnętrznym stanie. Znając ją i mając przesłany MAC może zweryfikować, czy nadesłano następny pakiet. Z punktu widzenia kryptograficznego liczniki są wartościami nonce i ponieważ obie strony wiedzą, jakiej kolejnej wartości należy się spodziewać, to nie musi ona być przesyłana. Do szyfrowania przeznaczone są dane oraz tag. Blok danych jest rozszerzany do rozmiaru akceptowalnego dla algorytmu AES. W tym wypadku stosujemy „prosty" pad. Jeśli w bloku brakuje 5 bajtów, to dołączamy do niego pięć bajtów, każdy z zapisaną w nim wartością 5 (…55555). Sposób uzupełniania ostatniego bloku wiadomości w szyfrowaniu blokowym (nie w generowaniu MAC!) był omówiony na wcześniejszych wykładach. Schemat szyfrowania to CBC z losowym IV. Na koniec do szyfrogramu dołączany jest jawny nagłówek (typ, wersja, długość). Daje to nam cały rekord w protokole TLS, który jest przesyłany do serwera. Dane zaznaczone ciemnym kolorem stanowią część zaszyfrowaną wiadomości, a zaznaczone na biało, to nagłówek, wcześniej zaszyfrowany i „otagowany", żeby go nie można podrobić, ale ostatecznie przesyłany w formie jawnej. Porównując to to wcześniej omówionych schematów posługujemy się tutaj rozwiązaniem MAC-then-ENCRYPT, przy czym włączamy w system licznik, zabezpieczający przed wysyłaniem powielonych wiadomości.

Rekord TLS: rozszyfrowywanie

(CBC AES-128, HMAC-SHA1)

Strona Serwera: **dec($k_{b\to s}$, record, $ctr_{b\to s}$)** :

krok 1: rozszyfrowanie schematu CBC z zastoswaniem k_{enc}

krok 2: sprawdzenie formatu padu:
wysłanie bad_record_mac jeśli się nie zgadza

krok 3: sprawdzenie tagu bloku [++$ctr_{b\to s}$ ll header ll data]
wysłanie bad_record_mac jeśli się nie zgadza

Zapewnia szyfrowanie z uwierzytelnieniem

(pod warunkiem, że nie dostarcza żadnych dodatkowych informacji podczas odszyfrowywania)

25

Odszyfrowywanie bloku danych wygląda w następujący sposób. Serwer otrzymuje blok danych (record) i posługuje się własną kopią klucza ($\mathbf{k_{b\to s}}$) i własną kopią licznika ($\mathbf{ctr_{b\to s}}$) do przeprowadzenia odszyfrowywania danych. W pierwszym kroku następuje uruchomienie algorytmu odszyfrowującego z kluczem k_{enc} . Potem sprawdzany jest format padu. Jeśli się nie zgadza, to odsyłany jest komunikat bad_record_mac i następuje zerwanie komunikacji. Do rozpoczęcia nowego połączenia muszą zostać wynegocjowane nowe klucze sesji. Po sprawdzeniu padu jest on odrzucany i sprawdzany jest MAC wiadomości. Znowu, jeśli sprawdzenie MAC zakończy się porażką protokół odsyła informację bad_record_mac i następuje zerwanie komunikacji. Jeśli wszystko jest OK od wiadomości odrzucany jest nagłówek i tag i przesyłany jako odszyfrowane dane. Proszę zwrócić uwagę, że jeśli ktoś przejmie jakiś blok danych i spróbuje po jakimś czasie przesłać go ponownie do serwera, to zostanie on odrzucony, ponieważ wewnętrzny stan licznika ulegnie zmianie i nie będzie się on zgadał z wartością licznika zapisaną w zaszyfrowanych danych. Liczniki okazują się eleganckim rozwiązaniem zapobiegającym atakom powtórzeniowym. Dodatkowo, ponieważ nadawca i odbiorca utrzymują sesję (w tym stan liczników) nie ma potrzeby włączania wartości liczników w przesyłany komunikat, a same liczniki nie wydłużają długości przesyłanej wiadomości. Zastosowany schemat gwarantuje zachowanie szyfrowania z uwierzytelnieniem, ponadto żadne dodatkowe informacje, poza stwierdzeniem, że odszyfrowanie się nie powiodło nie są wysyłane na zewnątrz. Istnieją ataki na TLS, jeśli system zwraca więcej informacji… W pokazanej wersji TSL znacznik bad_record_mac jest odpowiednikiem znacznika niepowodzenia w odszyfrowywaniu w szyfrowaniu z uwierzytelnieniem. Bardzo ważne jest, że odbiorca/atakujący dowiaduje się tylko , że odszyfrowanie się nie udało, ale nie jest mu podawana przyczyna niepowodzenia. Jeśli tylko dodalibyśmy, że odrzucenie odbyło się z jednej czy drugiej przyczyny, system mógłby być mocno zaatakowany (taki atak zostanie pokazany w dalszej części wykładu).

Literatura uzupełniająca

- The Order of Encryption and Authentication for Protecting Communications, H. Krawczyk, Crypto 2001.
- Authenticated-Encryption with Associated-Data, P. Rogaway, Proc. of CCS 2002.
- Password Interception in a SSL/TLS Channel, B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux, Crypto 2003.
- Plaintext Recovery Attacks Against SSH, M. Albrecht, K. Paterson and G. Watson, IEEE S&P 2009
- Problem areas for the IP security protocols, S. Bellovin, Usenix Security 1996.

Dan Boneh