Podstawy kryptografii i bezpieczeństwa systemów informatycznych

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2019/2020

inf.ug.edu.pl/~amb/

Integralność danych: MAC i funkcje skrótu

Integralność a prywatność

- Prywatność: Ewa nie potrafi odczytać wiadomości m z kryptogramu c
  – czy potrafi go zmienić złośliwie? (integralność)
- Szyfr strumieniowy
  – $c = m \oplus G(k)$, zmiana bitu w c zmienia ten sam bit w m
- Szyfr blokowy w trybach OFB oraz CTR
  – zasada szyfrowania jest podobna
  – zmiana bitu w c zmienia ten sam bit w m
- Szyfr blokowy w trybie ECB
  – zmiana bitu w c zmienia cały blok w wiadomości m
  – można bezkarnie zmienić kolejność bloków
- Szyfr blokowy w trybie CBC
  – zmiana bitu w IV zmienia bit w pierwszym bloku m
  – nagłówek pliku często ma ważne informacje

Integralność – definicja

- MAC (message authentication code)
  – algorytm generowania klucza k (np. losowy wybór)
  – algorytm obliczania $MAC(k, m)$ dla klucza i wiadomości m
  – algorytm weryfikacji dla klucza k, wiadomości m oraz kodu uwierzytelniającego t
- Odporność na atak
  – wybrany jest pewien klucz k
  – Ewa ma dostęp do wyroczni obliczającej $MAC(k, \_)$
  – Ewa wygrywa jeśli znajdzie prawidłowy skrót dla *jakiejkolwiek* nowej wiadomości
- Atak przez powtórzenie (Ewa kopiuje wiadomość ze skrótem)
  – nie jest objęty tą definicją
  – wymaga pojęcia stanu
  – np. uzgodniony zegar, albo licznik

Funkcja pseudolosowa jako MAC

- Dana funkcja pseudolosowa $F : [n] \times [n] \to [n]$
- $MAC(k, m) = F(k, m)$
  – weryfikacja: $t = MAC(k, m)$
- Twierdzenie: jest to bezpieczny algorytm uwierzytelniania dla ciągów ustalonej długości
  – dla funkcji losowej wartości $f(x)$ oraz $f(y)$ są niezależne,
  – funkcja pseudolosowa jest PPT nieodróżnialna od losowej
  – więc znajomość wielu wartości nie pomaga w znalezieniu nowej
- Problemem jest nadal funkcja $MAC : [n] \times [*] \to [*]$ dla ciągów dowolnej długości
  – a w praktyce będziemy żądać by $MAC : [n] \times [*] \to [\ell]$, stała długość

MAC dla ciągów dowolnej długości

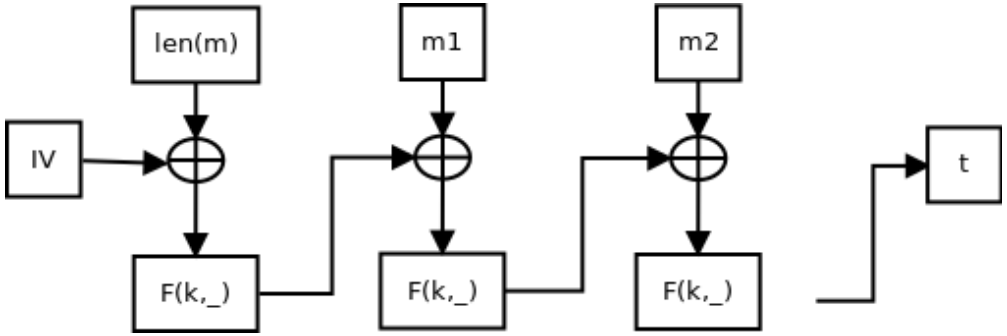
- Pomysły nieprawidłowe:
- Obliczyć $\oplus$ dla wszystkich bloków i wtedy MAC tej sumy
  – łatwo zmienić wiadomość zachowując sumę
- Obliczyć MAC dla każdego bloku osobno
  – można zmienić kolejność bloków
  – wiadomość można po prostu uciąć
  – można sklejać fragmenty różnych wiadomości
- Obliczyć MAC dla bloków numerowanych
  – nadal wiadomość można uciąć
  – nadal można sklejać fragmenty różnych wiadomości
- Obliczyć MAC dla bloków numerowanych i posiadających informację o łącznej długości
  – nadal można sklejać fragmenty różnych wiadomości zachowując długość

MAC dla ciągów dowolnej długości c.d.

- Rozwiązanie: każdy blok zawiera:
  – numer bloku (uniemożliwia przestawienie)
  – długość pliku (np. liczbę bloków, uniemożliwia ucięcie)
  – liczbę jednorazową (uniemożliwia sklejanie wiadomości)
  – oraz fragment wiadomości
  – MAC jest zestawem: liczba jednorazowa i ciąg MAC bloków

MAC dla ciągów dowolnej długości w trybie CBC

- Cel:
  – MAC powinien dawać wynik stałej długości
  – powinien stosować się do ciągów naprawdę długich
- MAC działa jak tryb blokowy CBC
  – ale tylko ostatni blok jest zwracany
  – wektor początkowy jest ustalony w definicji algorytmu
  – długość pliku też jest kodowana



Szyfrowanie w trybie CBC a MAC – różnice

- Wektor początkowy
  – dla szyfrowania jest niezbędny, zapewnia niedeterminizm
  – dla obliczania MAC jest szkodliwy, umożliwia zmianę pierwszego bloku wiadomości
- Zestaw wyników funkcji losowej
  – dla szyfrowania jest niezbędny, umożliwia obliczenie odwrotnego algorytmu (odszyfrowanie)
  – dla MAC jest niepotrzebny
- Kodowanie długości
  – dla szyfrowania jest niepotrzebne, cała wiadomość jest odtwarzana
  – dla MAC nie ma innego sposobu zaznaczenia długości pliku
  – gdyby zwracać wszystkie MAC'i też nie byłoby bezpiecznie
  – są powody, by długość kodować jako pierwszy blok

Funkcje skrótu

- Dowolna funkcja $h : [*] \to [n]$
  – dziedzina: ciągi bitów dowolnej (dużej) długości
  – przeciwdziedzina: ciągi bitów długości ustalonej i niedużej
- Przykład $h : [n+n] \to [n]$, $h(x, y) = x \oplus y$
  – funkcję tę można iterować i przetwarzać ciągi dowolnej długości: $H(x, \langle y, Z\rangle) = H(h(x, y), Z)$
  – dla dowolnego ciągu bitów trzeba jeszcze uzupełnić ostatni niepełny blok
- Własności funkcji H
  – łatwo się oblicza (w przykładzie jest to po prostu suma)
  – łatwo znaleźć dwa ciągi t.ż. $H(x) = H(y)$

Zastosowania funkcji skrótu

- Nie kryptograficzne:
  – znajdowanie indeksów tablic dla argumentów ze zbioru $[*]$ albo zbioru $[N]$ dla dużego N (tablice mieszające)
  – wykrywanie przypadkowych błędów transmisji: przesyłane są wiadomość m oraz skrót $h(m)$, odczytywane są m_1 oraz h_1, jeśli $h(m_1) \neq h_1$, to znaczy, że wystąpił błąd
- Kryptograficzne:
  – wykrywanie celowych i złośliwych zmian dokumentów
  – skrócenie wiadomości dla kryptografii asymetrycznej (podpis)
- Inne nazwy:
  – *hash*, odcisk palca (*fingerprint*), *message digest*
  – MAC (dla funkcji z hasłem)

Własności kryptograficznych funkcji skrótu

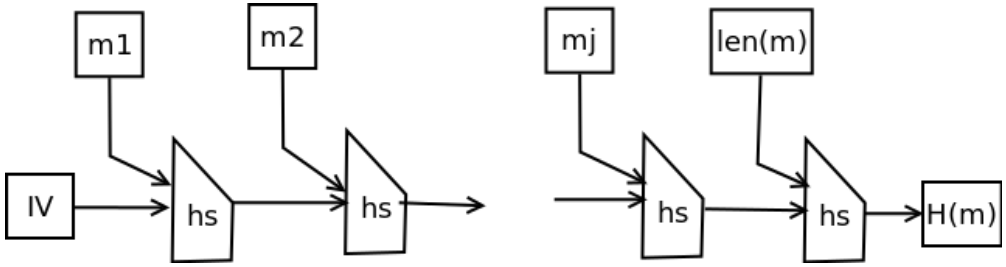
- Dana wiadomość m oraz skrót $h(m)$
  – zmieniona wiadomość m_1 oraz $h(m_1)$
  – jeśli $h(m) = h(m_1)$ to nie ma sposobu wykrycia zmiany
  – w.p.p. atakujący musi przekazać inny skrót
- Założenia:
  – łatwo obliczyć (ale nie MAC, które wymaga hasła)
  – trudno jest znaleźć jakiekolwiek m_1 t.ż. $h(m_1) = y$ (funkcja jednokierunkowa, problem 1. przeciwobrazu)
  – trudno jest znaleźć m_1 t.ż. $h(m_1) = h(m)$ dla danego m (słaba bezkolizyjność, problem 2. przeciwobrazu)
  – trudno jest znaleźć jakiekolwiek dwie wiadomości m oraz m_1 takie że $h(m) = h(m_1)$ (silna bezkolizyjność)

Atak urodzinowy

- Jakie jest prawdopodobieństwo, że dwie osoby spośród n mają urodziny tego samego dnia?
  – dla $n = 2$, prawd. $\approx \frac{1}{366}$
  – dla $n = 367$, prawd. $= 1$
  – dla $n = 23$, prawd. $> \frac{1}{2}$
- Prawd. $\approx exp(-n^{2/2N})$
  – jest znacznie większe niż żądanie, by zachodziła równość z zadaną wartością
  – dla dwóch zestawów liczb $< N$ o wielkości $\sqrt{N}$ elementów jest duża szansa na wspólny element
  – np. dla $N = 2^{56}$ wystarczy zgromadzić zestawy po 2^{28} elementów, czyli gigabajty

Metoda Merkle-Damgarda

- Dana funkcja $h_s : [n + n] \to [n]$ (funkcja kompresji)
  – (może zależna od dodatkowego parametru s)
  – iteracja dla ciągów dowolnej długości: $H(x, \langle y, Z\rangle) = H(h_s(x, y), Z)$
  – dla dowolnego ciągu bitów trzeba jeszcze uzupełnić ostatni, niepełny blok ciągu
  – na końcu blok kodujący długość pliku
  – wektor początkowy jest ustalony



Metoda Merkle-Damgarda, własności

- Jeśli funkcja kompresji h_s ma własność bezkolizyjności, to funkcja H też ma taką własność
  – dw. gdyby dwie wiadomości miały ten sam skrót H
  – to albo będą się różnić wielkością (ostatni blok da kontrprzykład dla h_s)
  – albo będą się różnić wcześniej, wcześniejszy blok da kontrprzykład dla h_s
- MAC w algorytmie Merkle-Damgarda
  – NMAC (*nested* MAC): dla klucza k dodatkowy blok na końcu $h_s(k, H(m))$

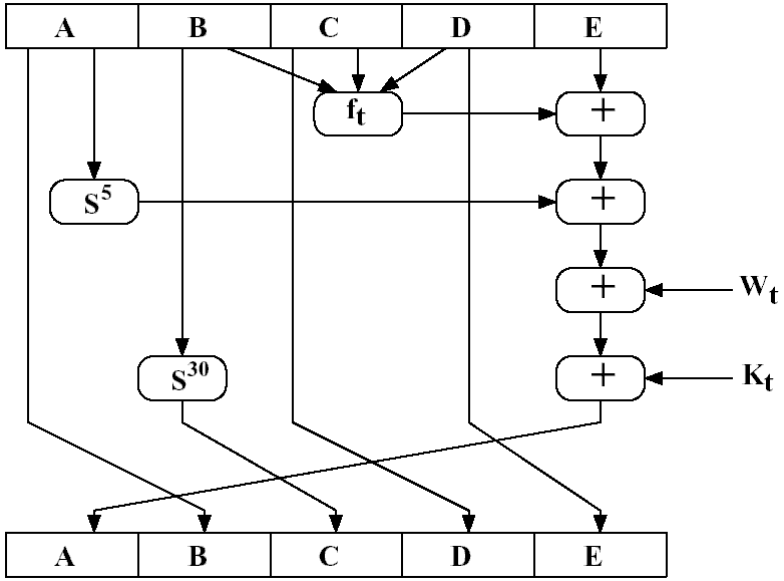
Standard SHA–1 (*secure hash algorithm*)

- Opracowany przez NIST (National Institute of Standards and Technology)
  – pierwsza wersja była niedoskonała, stąd SHA–1
  – produkuje skrót 160 bitowy
- Standard określa funkcję kompresji, jest ona iterowana
  – $m = \{m_0, m_1, m_2, \ldots\}$, X_0 początkowa wartość rejestru, $X_{j+1} = h(X_j, m_j)$, $h(m)$ jest równe ostatniej wartości rejestru
  – standard określa też początkową wartość rejestru oraz sposób wypełnienia ostatniego bloku: ostatnie 64 bity określają długość m, brakujące są uzupełnione zerami

SHA–1 dla jednego bloku

- Operacje nieliniowe: 'and', 'or' bitowo
  – operacje liniowe: 'not', $\oplus$ czyli dodawanie modulo 2, dodawanie modulo 2^{32}, przesunięcie bitów w lewo
  – definicja funkcji pomocniczych
  $f(B,C,D) = (B \wedge C) \vee ((\neg B) \wedge D)$, $f(B,C,D) = B \oplus C \oplus D$, ...
  – definicja 85 stałych 32 bitowych
- Rejestr ma 5 elementów: A, B, C, D, E po 32 bity, inicjalizowanych przez 5 stałych, blok ma 16 fragmentów 32–bitowych
  – dla 80 rund obliczamy $shift_5(A) + f(B,C,D) + E + W + K \mapsto A$, $A \mapsto B$, $shift_{30}(B) \mapsto C$, $C \mapsto D$, $D \mapsto E$, f są różne dla różnych rund, K kolejną stałą, w jest początkowo fragmentem bloku, później $W_j = shift(W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16})$

Jedna runda w SHA–1



`http://nsfsecurity.pr.erau.ededu/crypto/sha_1.html`
cytat za *Cryptography and Network Security: Principles and Practice* William

MD5

- Autor: Rivest
- Skrót 128 bitowy
  – 4 rejestry 32 bitowe
  – 64 rundy (4 cykle po 16)
  – w każdym cyklu inna funkcja nieliniowa z efektem lawinowym, przesunięcia bitów, dodawanie z bieżącymi danymi
- Znaleziono kolizje!
- MD2, inna funkcja
  – bardzo wolna
  – ale chyba bezpieczniejsza
  – jest używana w protokole PEM, bezpiecznej poczty elektronicznej

Inne funkcje skrótu

- Funkcje md4, md5, ripemd, sha-1
- Silna bezkolizyjność jest problematyczna, tylko sha-1 jest w miarę dobre
  – znaleziono dwa certyfikaty X.509 o tej samej wartości funkcji skrótu md5
  – gdyby jeden był podpisany przez wystawcę certyfikatów, to automatycznie drugi też
- Nie ma problemów ze słabą bezkolizyjnością i nieodwracalnością
  – gdyby te własności były naruszone, to duża część kryptografii byłaby w kłopocie

Funkcja skrótu za pomocą szyfrowania

- Dana funkcja szyfrująca $E : [\ell+n] \to [n]$, n długość szyfrowanych bloków, ℓ długość klucza
  – można przerobić ją na funkcję skrótu na wiele sposobów: $h(k, m) = E(k, m) \oplus m$, albo $E(k, m) \oplus m \oplus k$, albo $E(k, k \oplus m) \oplus m$ albo ...
  – i dalej iterować użycie dla większej liczby bloków
  – jest to równoważne zastosowaniu szyfru symetrycznego w wersji blokowej i przyjęciu ostatniego bloku jako skrótu
- Możliwość ataku urodzinowego wyklucza szyfry o kluczu mniejszym niż np. 128 bitów
  – np. klasyczny DES

Szyfrowanie za pomocą funkcji skrótu

- Funkcja skrótu h pozwala wygenerować ciąg pseudolosowy
  – x_0 musi być losowe i przesłane niezależnie jako IV
  – $x_j = 8$ bitów z $h(k, x_{j-1})$, k jest kluczem, jest użyte jako ciąg pseudolosowy
  – tzn. $c_j = m_j \oplus x_j$, ciąg x_j jest dodawany do wiadomości w celu zaszyfrowania/odszyfrowania

Szyfrowanie i uwierzytelnianie

- Dane dwa klucze, jak zapewnić poufność i integralność?
  1) przesłać $Enc(k_1, m)$ oraz $MAC(k_2, m)$
- ale MAC może ujawnić całą wiadomość
  – a praktycznie zawsze jest deterministyczny: ŹLE
  2) przesłać $Enc(k_1, m||MAC(k_2, m))$
- szyfr nie musi być odporny na atak z wybranym kryptogramem
  – być może nawet da się odtworzyć cały tekst jawny: ŹLE
  PRAWIDŁOWE ROZWIĄZANIE:
  3) przesłać $Enc(k_1, m)$ oraz $MAC(k_2, Enc(k_1, m))$
- uniemożliwia atak przez modyfikację kryptogramu
- bezpieczeństwo takie same jak dla Enc
- UWAGA: MAC i Enc mogą być funkcjami wzajemnie odwrotnymi (szyfry blokowe itp), klucze muszą być różne