C labb komplettering Lavineffekt

Gustav Jenner

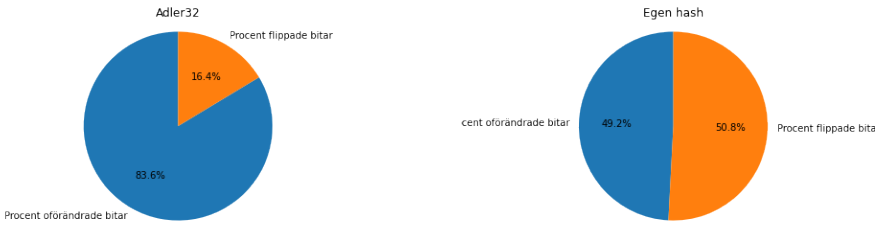
Maj 2022

1 Metod

Lavineffekts testet utfördes genom att jämföra hammingavståndet mellan två hashvärden där små förändringar i in nyckeln har förändrats. Exempelvis är hammingavståndet mellan bit numret ”0101” och ”1101” = 3. I denna test så skedde små förändringar i nyckel genom beräkna hashvärdet där en bit i nyckeln inverterades och sedan beräkna hamingavståndet mellan original hashvärdet mot det förändrade hashvärdet. Dvs in nyckel ”1000” jämfördes mot ”0000”, ”1100”,”1010”,”1001”. Alltså skedde N-gånger jämförelser för en nyckellängd med N-bitar.Sedan beräknades procenten av flippade gentemot oförändrade bitar. En hashfunktion som har en lavineffekt nära 50% alltså att bitarna har en 50% sannolikhet att flippas anses säker. En sådan hashfunktion klarar SAC eller ”Strict Avalanche Criterion”[1]. En hashfunktion där antingen majoriteten av bitar är de samma eller där majoriteten av bitarna flippas när småförändringar hos nyckeln sker anses däremot osäker.

2 Resultat

Datan gjordes på slumpmässig genererad nyckel på längd 10 och repeterades 10000 gånger för att få ett genomsnitt.



(a) Lavineffekt på Adler32

(b) Lavineffekt Egen hash

Figure 1: Lavineffekt för de två hash metoderna

3 Analys

Som tidigare nämnt i metoden är en 50% fördelning mellan flippade bitar och önskevärd. Adler32 har en dålig lavineffekt om man tittar på 1a och klarar inte SAC då mindre än 20% av bitarna förändrats detta innebär en svaghet hos adler32 och det kan innebära att nyckeln för ett hashvärde går att lista ut utifrån hashvärdet[1]. Lavineffekten för den egna hashfunktionen däremot ligger nära 50% och uppfyller SAC iallafall för längre nyckellängder. Om man tittar i appendix på figur 2 och 3 ser man att den egna funktion har en dålig lavineffekt för nyckellängder under 10 medans Adler32 alltid har en dålig lavineffekt. Intressant nog verkar Adler32 ha som bäst lavineffekt vid längden 40 vilket antagligen hänger ihop med att det även är längden där modulo gränsen uppnås (se original rapport).

4 Appendix

4.1 kod

```
#Logic gate: XOR
def XOR_Gate(A, B):
    C = int(A) ^ int(B)
    return C
#Logic gate: NOT
def NOT_Gate(A):
    return 1-int(A)
#Logic gate: XAND
def XAND_Gate(A, B):
    XOR = XOR_Gate(A,B)
    C = NOT_Gate(XOR)
    return C

#Räknar ut hamminavståndet mellan två tal och retunerar sedan procenten bitar de har gemens
def hamming_distance(num_1, num_2):
    count = 0
    x = format(num_1, "b")
    y = format(num_2, "b")

    if len(x) < len(y):# lägger på 0 talen har olika längder
        x = "0"*(len(y)-len(x)) + x

    elif len(y) < len(x):
        y = "0"*(len(x)-len(y)) + y

    for bit in range(len(x)):
        count += XAND_Gate(x[bit],y[bit])
```

```
    return count/len(x)
#tar in binärt tal, retunerar lista med binäratal med ett hammingavstånd på 1 från orginal t
#Dvs om invärdet är "000" blir ut listan "100","010","001"
def bit_change(num):
    key = list(num)
    change_list = []
    for bit in range(len(key)):
        tmp = key.copy()
        tmp[bit] = str(NOT_Gate(tmp[bit]))
        x = "".join(tmp)
        change_list.append(x)

    return change_list

#Räknar den genomsnittliga lavineffekten för en nyckel
def avalanche(key, hash):
    orig = hash(key)
    change_list = bit_change(ASCII_to_binary(key))
    bit_flip = []
    for num in range(len(change_list)):
        key_b = binary_to_ASCII(change_list[num])
        tmp = hamming_distance(hash(key_b),orig)
        bit_flip.append(tmp)
    return sum(bit_flip)/len(bit_flip)

#tar in binärt tal och retunerar ASCII värdet
def binary_to_ASCII(bin_num):
    tmp3 = ""
    u = int(len(bin_num)/7)
    for i in range(u):

        tmp = bin_num[i*7:(i+1)*7]
        tmp2 = int(tmp,2)
        tmp3 += chr(tmp2)
    return tmp3
#tar in ASCII värde och retunerar binärt tal
def ASCII_to_binary(string):
    bin_str = ""
    for chr in string:
        num = ord(chr)
        bin_str += format(num,"b")

    return str(bin_str)
```

4.2 figurer

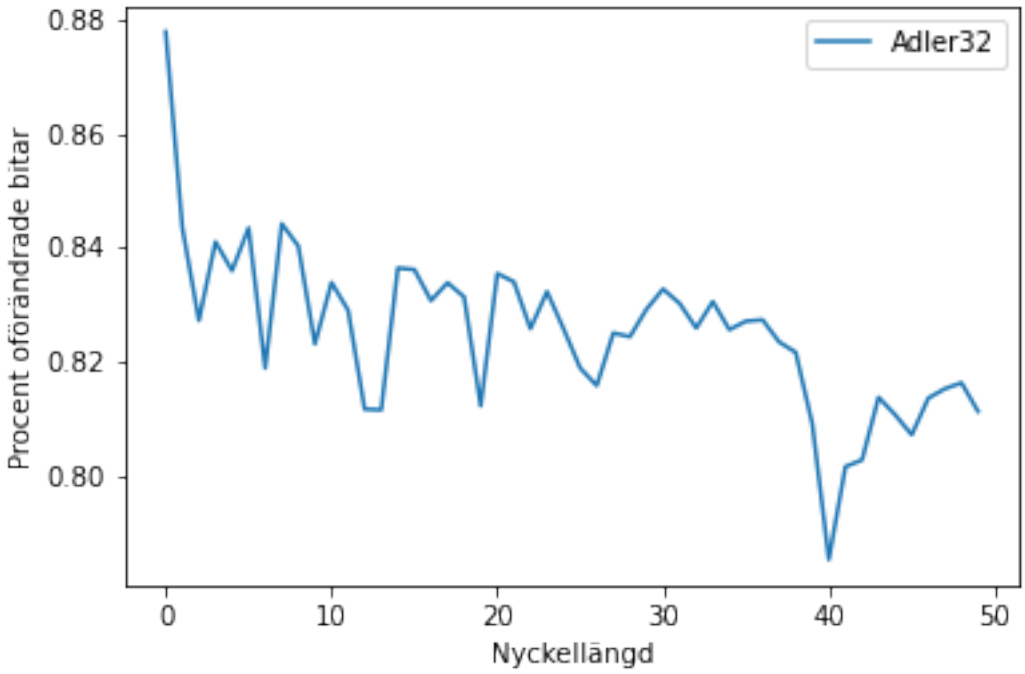


Figure 2: Nyckellängd gentemot lavineffekt för adler32

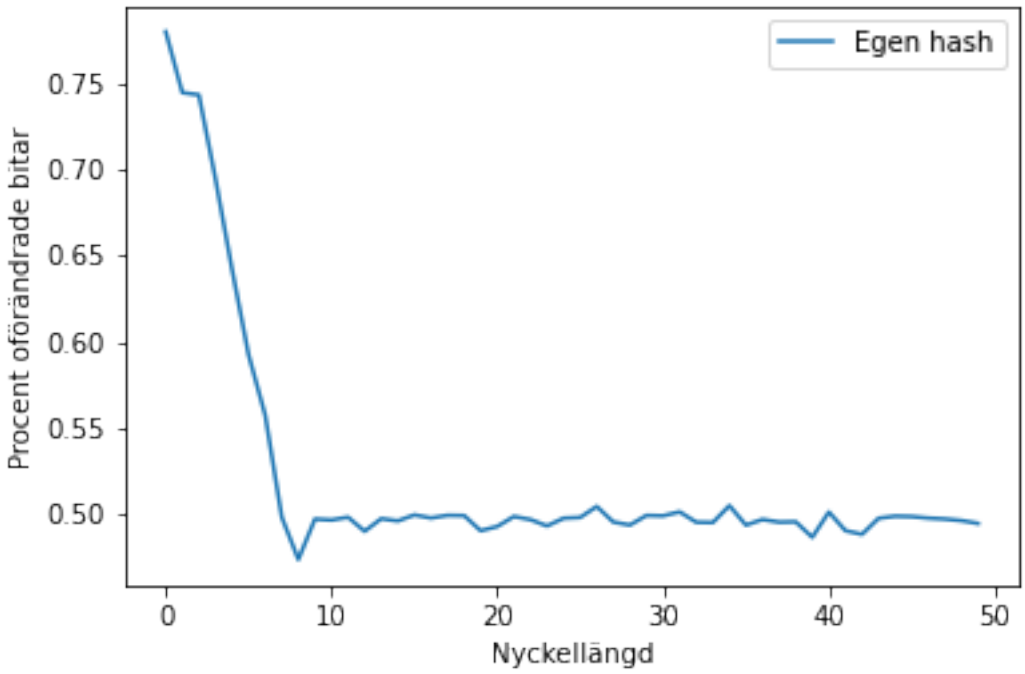


Figure 3: Nyckellängd gentemot lavineffekt för egna hashfunktion

5 Källor

[1]”Avalanche effect - Wikipedia”, En.wikipedia.org, 2022. [Online].Tillgänglig: https://en.wikipedia.org/wiki/Avalanche_effect. [Hämtad: 10- Maj- 2022].