情報セキュリティ基礎 演習問題

光成滋生

last update: 2026/1/7

HMAC-SHA3が使われない理由

適切な語句を入れよ。

- HMAC-SHA3が使われない理由を述べる。
  SHA-2などの(A)は(B)を受けるが、SHA-3は(C)であり(B)を受けない。
  HMACはハッシュを(D)回することで(B)を防いでいるが、SHA-3では不要であるため。

答え

- (A) Merkle–Damgård構造
- (B) 伸長攻撃
- (C) スポンジ構造
- (D) 2

位数計算による素因数分解

適切な語句を入れよ。

- 素因数分解を位数計算に帰着する。
  $n = 2021027$ とする。
  $g = 2$ としたとき $g^r \equiv 1 \pmod{n}$ となる最小の r（位数）は(A)である。
- $a = g^{(r/2)} - 1 \pmod{n}, b = a + 2 \pmod{n}$
  とする。
- このとき $g^r - 1 \equiv ab \pmod{n}$ であり、a と n の最大公約数は(B)であり、b と n の最大公約数は(C)である。
  よって $n = pq\,(p < q)$ としたとき、$p =$(D), $q =$(E)である。

答え

- (A) 72072, (B) 2003, (C) 1009, (D) 1009, (E) 2003

前問の計算方法

Pythonによるプログラム例

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def find_order(g, n):
    v = 1
    for r in range(1, n):
        v = (v * g) % n
        if v == 1:
            return r
    raise Exception('not found')

n=2021027
g=2
r=find_order(g, n) # 72072
a=pow(g, r//2,n)-1 # 943413
b=a+2 # 943415
gcd(a,n) # 2003
gcd(b,n) # 1009
```

TLS 1.3

適切な語句を入れよ。

- TLS 1.3で公開鍵暗号化方式(Public Key Encryption)が使われなくなった理由を述べよ。
- (A)が漏洩したときに、盗聴されていた(B)が過去にさかのぼって(C)される危険性があるから。すなわち(D)を保つため。

答え

- (A) 秘密鍵
- (B) 暗号文
- (C) 復号
- (D) 前方秘匿性

ECDSA攻撃1

適切な語句を入れよ。

- ECDSAの署名は次のアルゴリズムで表される。
  Hをハッシュ関数、Pを楕円曲線の位数rの点、秘密鍵sを固定する。
  メッセージmに対してh=H(m)。kをランダムにとり、tをkPのx座標、σ=(t, u)=(t, (h+st)/k mod r)とする。
- 今、ある署名プログラムのkの選択が、ある初期時刻からの秒数の値を利用していたとする。ある時刻におけるm1の署名がσ1=(t1, u1)、a秒後におけるm2の署名がσ2=(t2, u2)であった。このとき、署名鍵sを求めよ。
- 解法: σ1、σ2の生成時に使ったkをk1, k2, h1=H(m1), h2=H(m2)とするとk2 = (A)である。
- $\mathbb{F}_r$ の中で(B)となる。よってk1とsに関する連立方程式Qができる。

$$\begin{cases} u_1 k_1 - t_1 s = h_1 \\ u_2 k_1 - t_2 s = h_2 - u_2 a \end{cases}$$

ECDSA攻撃1

続き

- 行列Aを

$$A = \begin{pmatrix} u_1 & -t_1 \\ u_2 & -t_2 \end{pmatrix}$$

とするとdetA=(C)である。Aの逆行列を求めてQを解くとs =(D)となる。

答え

- (A) k1 + a
- (B) u1=(h1 + s t1)/k1, u2=(h2 + s t2)/k2
- (C) u1 * (-t2) + t1 * u2
- (D) (1/detA)*(-u2 h1 + u1 * (h2 - u2 a))

ECDSA攻撃2

記号はそちらを参照すること。

- ある時刻におけるm1="Don't use AI."の署名が
  σ1=(t1, u1)=
  (0x6d5db36d6b5c5e18cedcbd7165dcdf5219d1b98646334c810103e714696185d4,
  0xda87a6f4db00a9a69e726b782aa1098d028bd2476d427987d290a35fb9d142e8)
- 60秒後に生成されたm2="Solve it yourself."の署名が
  σ2=(t2, u2)=(0x8fbc1bed0b1ffa5068b75144503fceb6047bca7fca10cc2629a6ea6e998c99db,
  0x1b31394d4b10d86770a01c92b9038fa6d2a916957ae134241f35aaab1a79cd6c)
  であった。このとき、署名鍵sを求めよ。
- ただし、楕円曲線とハッシュ関数はsecp256k1とSHA256を利用し、
  r=0xfffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
  h1=H(m1)=0x94cf850eb237e579a6336dcab8aeafccea7a7b3ef15cd283c691c3fb2dff4c4e
  h2=H(m2)=0x97c982f7320f4ff1402636575c354c452e729f4999381c9116b65c8c99e8f385
  とする。

ECDSA攻撃2

```
r=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
a=60
h1=0x94cf850eb237e579a6336dcab8aeafccea7a7b3ef15cd283c691c3fb2dff4c4e
h2=0x97c982f7320f4ff1402636575c354c452e729f4999381c9116b65c8c99e8f385
t1=0x6d5db36d6b5c5e18cedcbd7165dcdf5219d1b98646334c810103e714696185d4
u1=0xda87a6f4db00a9a69e726b782aa1098d028bd2476d427987d290a35fb9d142e8
t2=0x8fbc1bed0b1ffa5068b75144503fceb6047bca7fca10cc2629a6ea6e998c99db
u2=0x1b31394d4b10d86770a01c92b9038fa6d2a916957ae134241f35aaab1a79cd6c
```

- Pythonで上記を入力し、detA = u1 * (-t2) + t1 * u2を計算すると、detA=(A)である。
  1/detA=invA=pow(detA, -1, r)はinvA=(B)、
  s = (invA * (-u2 * h1 + u1 * (h2 - u2 * a)))%rはs=(C)となる。回答は全て10進数で入力せよ。

答え

```
(A) 288465872406719137860896422350897879918948234796204654026321553605768148869l6
(B) 309348750245149562026789234287895669891547696274952505710623255424824020606 72
(C) 111601559888206902938973665514922793002413295977247509890789347730332982800612
```

Regev暗号

適切な語句を入れよ。

- 小さい（安全ではない）パラメータでRegev暗号の動作を確認する。
  q=257, n=m=2とする。x^Tをxの転置とする。
  秘密鍵をs =(102 106)^T, 行列を$A = \begin{pmatrix} 71 & 188 \\ 20 & 102 \end{pmatrix}$, ノイズをe =(-1 0)^Tとする。
  このとき公開鍵b = A s + e=(A)である。
- M=1の暗号化はr= (0 1)^Tを乱数としてu = A^T r = (B)
  v = b^T r + (q//2)M = (C)としてEnc(M)= c = (u, v)である。
- 復号はDec(c) = v - s^T u = (D)この値はq//2=(E)に近いのでDec(c)=(F)となる。

答え

- (A) (184 2)^T, (B) (20 102)^T, (C) 130, (D) 128, (E) 128, (F) 1