The AH provides integrity and sender authentication for the IP packets. The header is computed by hashing the payload (IP packet) together with the secret key.

The ESP protocol provides privacy by encrypting the payload. The payload is either the entire IP packet (tunnel mode) or the upper layer protocol data (transport mode).
\[ 273 = 7 \cdot 39 = 3 \cdot 7 \cdot 13 \]

\[ \phi(273) = (3-1)(7-1)(13-1) = 2 \cdot 6 \cdot 12 = 12^2 = 144 \]

(BTW, twelve dozens = 144 are a "gross" which is an old measure — this has nothing to do w/ this problem)

Key space = (\# poss. "a") \times (\# poss. "b")

= 144 \times 273 = 39,312
Rijndael applies 4 transformations to the data in the 1st round:

a) ByteSub
b) ShiftRow
c) MixColumn
d) AddRoundKey

d) **ByteSub:**

Same S-Box operation is applied to every byte.

There are 16 bytes w/ value FF₁₀ = 1111 1111₂

1. step: inverse of \(x^8 + x^4 + x^3 + x + 1 \in GF(2^8)\)

\[ P(x) = x^8 + x^4 + x^3 + x + 1 \]

\[ \bar{x} \cdot EA \]

(1) \( P(x) = [x+1](x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1) + [x^4 + x^3 + x] \), via long division

\[ t_2(x) = t_0 - q_1 t_1 = 0 - q_1 = -(x+1) = x+1 \]

(2) \( A(x) = [x^3 + x + 1](x^4 + x^3 + x) + [1] \), via long division

\[ t_2(x) = t_1 - q_2 t_2 = 1 - [x^3 + x + 1](x+1) = (x^4 + x^3 + x) + (x^3 + x + 1) + 1 \]
\[ = x^4 + x^3 + x^2 \]

(3) \( x^4 + x^3 + x = [x^4 + x^3 + x] + [0] \)

\[ \Rightarrow A^{-1}(x) = x^4 + x^3 + x^2 \mod P(x) \]
Each column of the new state is computed as

\[
\begin{bmatrix}
C_0 \\
C_1 \\
C_2 \\
C_3
\end{bmatrix} =
\begin{bmatrix}
X & x+1 & 1 & 1 \\
4 & X & x+1 & 1 \\
1 & 1 & x & x+1 \\
X+1 & 1 & -1 & x
\end{bmatrix}
\begin{bmatrix}
B(x) \\
B(x) \\
B(x) \\
B(x)
\end{bmatrix}
\]

\[C_0 = (X + (x+1) + 1 + 1) = B(x) = B(x)\]
\[C_1 = (1 + x + (x+1) + 1) = B(x) = B(x)\]
\[C_2 = B(x) = B(x)\]
\[C_3 = B(x) = B(x)\]

\(\Rightarrow\) state after MixColumn still consists of 16 Bytes w/ value \(B = 16_{\text{h}}\)

8) Add Round Key

Since subkey is all-one, all bits are flipped

\[-B \oplus \underbrace{1111 \ 1111}_{1110} = 1110 \ 1001_2 = E9\]

\(\Rightarrow\) output after final round

\[E9 \ E9 \ E9 \ldots \ E9\]

16 times
cont

2. Step: Mult. of $A^{-1}(x)$ by matrix + vector add.

\[ A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

\[ B = ByteSub(A) = \begin{bmatrix} 10 \\ 01 \\ 11 \\ 00 \end{bmatrix} \]

\[ B = H \cdot A + V = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

\[ B = 16_h = (0001 \ 0110) \leftrightarrow B(x) = x^4 + x^2 + x \]

\[ \Rightarrow \text{state cons. of 16 (}=128/8) \] identical bytes:

16, 16, 16, ... 16_h

3) ShiftRows transformation does not change data, since it's only a byte re-ordering.

8) MixColumns

performs mult. of $B(x)$ w/ $1$, $x$, and $x+1$

\[ B(x) = x^4 + x^2 + x = 0001 \ 0110 \quad \text{[not needed]} \]

\[ x \cdot B(x) = x^5 + x^3 + x^2 = 0010 \ 1100 \quad \text{[\textbackslash]} \]

\[ (x+1) \cdot B(x) = (x^5 + x^3 + x^2) + (x^4 + x^2 + x) = x^5 + x^4 + x^3 + x = 0011 \ 1010 \]
$x = y^a \mod n$

$a = b^{-1} \mod \phi(n)$

$\phi(n) = 100 \cdot 106 = 10600 \; \; n = 10607$

Extended EA: w/ $\phi(n)$, b

$10600 = 2 \cdot 4497 + 1606 \; \; t_2 = t_0 - q_1 b_1 = -q_1 = -2$

$4497 = 2 \cdot 1606 + 1285 \; \; t_3 = t_1 - q_2 t_2 = 1 - 2(-2) = 5$

$1606 = 1 \cdot 1285 + 321 \; \; t_4 = t_2 - q_3 t_3 = -2 - 1(5) = -7$

$1285 = 4 \cdot 321 + 1 \; \; t_5 = t_3 - q_4 t_4 = 5 - 4(-7) = 33$

$321 = 321 \cdot 1 + 0$

$a = 33 \equiv 4497^{-1} \mod 10600$

$[\text{Check: } 33 \cdot 4497 = 148401 \equiv 1 \mod 10600]$}

$7411^{33} = 7411^{100001}$

$7411^2 \equiv 1747 \mod n$

$7411^4 = 7411^{100} \equiv 4435 \mod n$

$7411^8 = 7411^{1000} \equiv 485 \mod n$

$7411^{16} = 7411^{10000} \equiv 8278 \mod n$

$7411^{32} = 7411^{100000} \equiv 8904 \mod n$

$7411^{33} = 8904 \cdot 7411 \equiv 2 \mod n$
\[ a^{774} \equiv a^{774 \mod 10} \equiv a^4 \mod 131 \]

\[ 70^4 = (70^2)^2 = 53^2 = 58 \mod 131 \]
6) a)  
1) Decryption of y using symmetric alg. w/ key k:
\[ d_k(y) = x \| H(k \| x) \]
2) Concatenate k and x, where k is the 2nd secret key of the receiver.
3) Compute hash of k \| x
4) Compare computed hash value with the one received by decryption in 1)

b) 
1) Decrypt as in 1a)
\[ d_k(y) = x \| \text{sig}(H(x)) \]
2) Feed x(x) and sig(H(x)) into verification algorithm and check whether signature is valid. Ver. alg. needs public key of sender.
Protocol A

1) Confidentiality — yes, given through encryption

2) Integrity — yes, data manipulation (by Oscar) will result in a corrupted y value, which will most likely not lead to a valid pair \( (x', H(k_y || x')) \)

3) Non-repudiation — NO, both Alice and Bob can generate valid messages:

\[ f_{k_y}(x || H(k_y || x)) \]

A neutral 3rd party can not decide who generated the message

Protocol B

1) Confidentiality — yes, as above

2) Integrity — yes, manipulation of y will most likely result in a pair

\( (x', \text{sig}_{k_{D}}()) \)

For which the verification will not check and

3) Non-repudiation — yes, only sender can generate messages w/ valid signatures