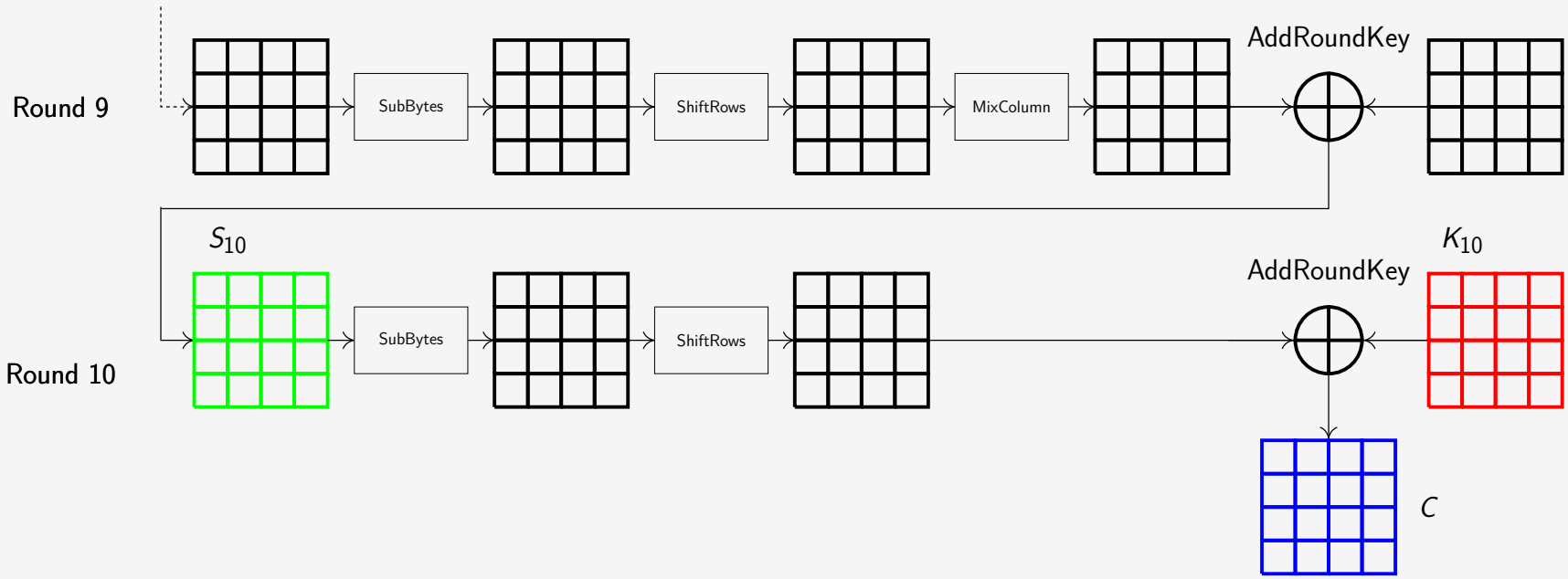
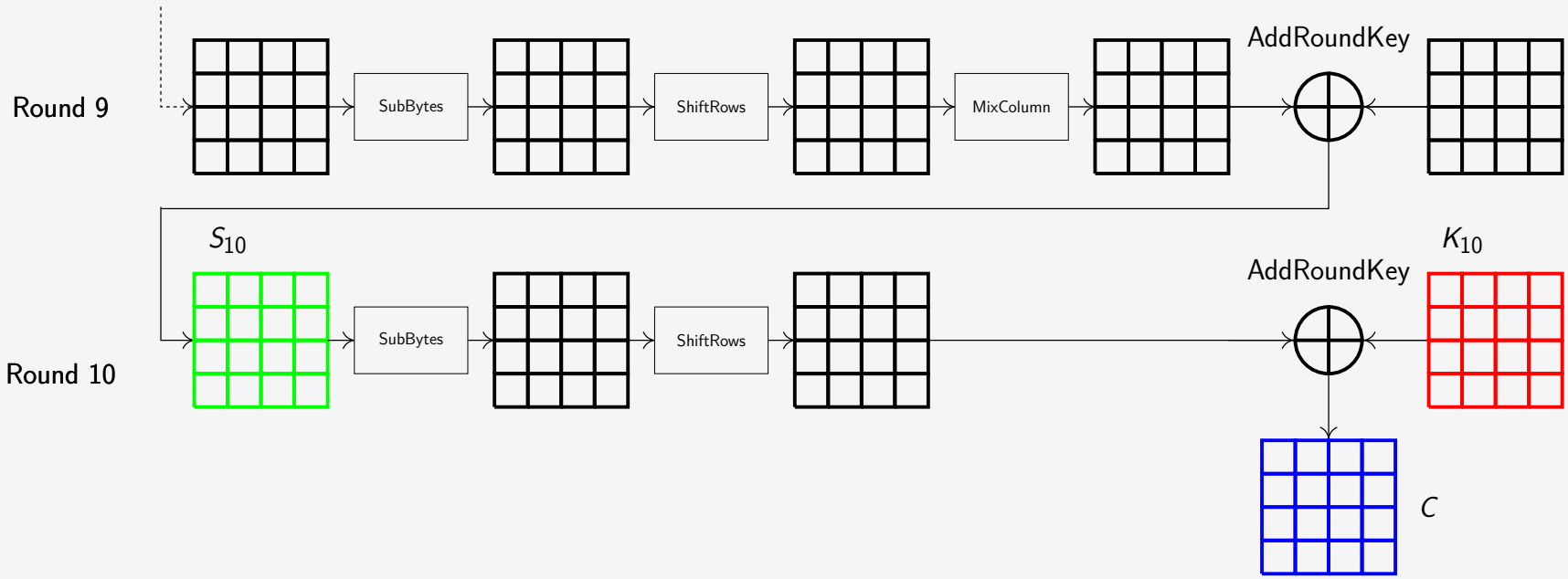
AES-128 Assignment 2



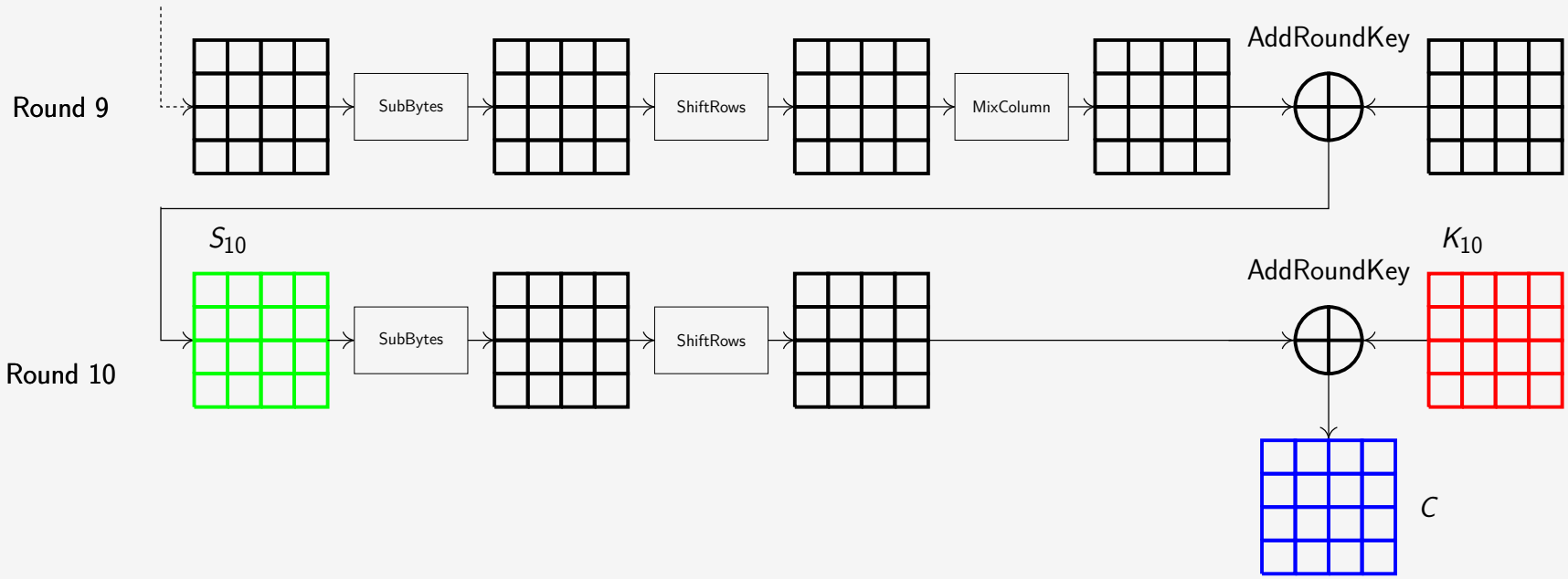
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?

AES-128 Assignment 2



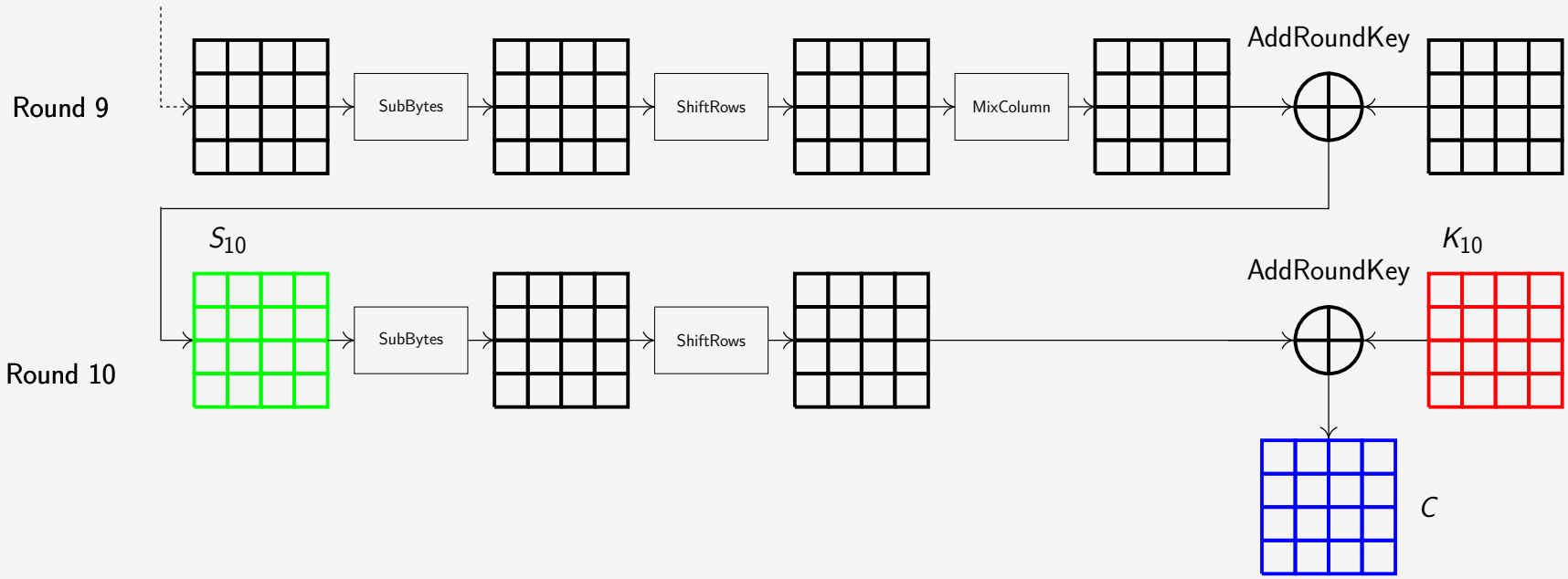
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code

AES-128 Assignment 2



- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$

AES-128 Assignment 2



- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$
- $HW(S_{10}) = HW(InvSubBytes[InvShiftRows(K_{10} \oplus C)])$

What to do: tip 1

- The only difference with the previous task is that here you have Hamming weights instead of the pure values
- Acquire several encryptions (lets say 10) with different data
- Your working formula is $HW(S_{10}) = HW(InvSubBytes[InvShiftRows(K_{10} \oplus C)])$
- I suggest to take every 0 ciphertext byte for all 10 encryptions
- Make a loop of key candidates from 0x00 till 0xFF and compute the formula above for a given key and 10 ciphertext bytes (at position 0) - this is called a model
- Then take the first sample from each of 10 traces and check if they are equal to the model of your key candidates
- Repeat the process for all 16 positions in the trace and all key candidates
- Only one key candidate and one position shall fully match your model with the given leakage
- Try to do the same process for the next ciphertext byte, etc.

What to do: tip 2

```
num_enc = 10

traces = np.zeros((num_enc, 16), dtype=np.uint8)
ctexts = np.zeros((num_enc, 16), dtype=np.uint8)

for iEnc in range(num_enc):
    ptext = '%032x' % random.randrange(16**32)
    output, ctext, trace = binary_aes128_encrypt(ptext, verbose=False)
    ctexts[iEnc,:] = ctext
    traces[iEnc,:] = trace

#Define numpy structure to keep the recovered key
key = np.zeros((16), dtype=np.uint8)

#Predefine all key candidates as a matrix num_enc x 256
key_cand = np.matlib.repmat(np.arange(256).astype(np.uint8), num_enc,1)

for iByte in range(16):
    #Compute Sbox output
    sbox_out = np.bitwise_xor(key_cand, ctexts[:,iByte].reshape(num_enc,1))

    #Apply inverse Sbox: invSbox[k ^ c[i, j]]
    sbox_in  = sca_training.invSbox[sbox_out]

    #Compute the Hamming weight (i.e., a model of the leakage): HW(invSbox[k ^ c[i, j]])
    hw_sbox_in = sca_training.HW_uint8[sbox_in]

    for iSample in range(16):
        best_keys = np.where(np.sum(hw_sbox_in==traces[:,iSample].reshape(num_enc,1), axis=0) == num_enc)[0]

        if len(best_keys) == 0:
            print('No key candidates remained for ciphertext byte', iByte, 'and sample index', iSample)

        if len(best_keys) == 1:
            print('Only one key candidate remained for ciphertext byte', iByte, 'and sample index', iSample)
            key[iByte] = best_keys[0]

        if len(best_keys) > 1:
            print('Several key candidates remained for ciphertext byte', iByte, 'and sample index', iSample)
            print(best_keys)
    print('\n')
#Get master key using the provided binary
key_schedule = sca_training.inverse_key_expansion(key)
```

Thank you!

Roman Korkikian, Nicolas Oberli

Side-channels and Fault Attacks
February 23rd, 2023 - June 29th, 2023