Differential Fault Attack (DFA) on AES

A **Differential Fault Attack (DFA)** is a cryptanalytic technique that exploits faults induced in a cryptographic algorithm to extract secret information, typically the encryption key. This attack is particularly effective against block ciphers and has been widely studied in the context of both classical and quantum-resistant cryptography.

Fault attacks are a subset of *side-channel attacks* where an adversary deliberately injects faults into a cryptographic device and observes the faulty output to recover secret information. This differs from purely mathematical cryptanalysis because it exploits physical vulnerabilities in the implementation rather than weaknesses in the cipher's design.

A **Differential Fault Attack (DFA)** is one such fault attack that compares correct ciphertexts with faulty ciphertexts to deduce information about the secret key.

How DFA Works

The fundamental idea of DFA is based on the fact that an induced fault in an intermediate round of encryption propagates in a structured manner, leading to differences in the final ciphertext. By analyzing these differences, an attacker can extract key bits.

Steps of a Differential Fault Attack

1. **Choose a plaintext** P and encrypt it under a secret key K to obtain a correct ciphertext C.
2. **Induce a fault** during the encryption process (typically in the last few rounds of the cipher).
3. **Obtain a faulty ciphertext** C' as a result of the fault.
4. **Compute the difference** between the correct ciphertext C and the faulty ciphertext C'.
5. **Analyze differential propagation** through the last rounds to deduce the secret key.

Most DFA techniques target the last rounds of a block cipher, particularly the **Substitution-Permutation Network (SPN)** or **Feistel structure**, for the following reasons:

- The last few rounds have fewer non-linear operations, making it easier to backtrack differences.
- Faults in the last rounds do not propagate too much, allowing the attacker to recover key bytes more easily.
- The attack often requires only a few faulted ciphertexts (typically less than 100, sometimes even 10-20) to break the full key.

Example: DFA on AES

Consider AES, which has 10, 12, or 14 rounds depending on key length. DFA on AES typically targets the **last round before the MixColumns step** (since the final round does not have MixColumns). The attack exploits the fact that:

- A single fault in the **SubBytes or ShiftRows step** causes predictable differences in only a few bytes of the ciphertext.
- By analyzing these differences over multiple faulty encryptions, the attacker can recover parts of the round key.
- Using the recovered last-round key, the full key can be computed via key schedule inversion.

Fault Models Used in DFA

The effectiveness of DFA depends on the type of faults induced. Some common fault models include:

Bit-Flipping Faults

- A single bit in an intermediate state is flipped randomly or in a controlled manner.
- Example: A fault is injected into a single bit of a register during AES encryption.

Byte/Nibble Faults

- A single byte (or nibble) is corrupted, leading to localized errors in the ciphertext.
- This is common in software implementations where faults affect registers or memory cells.

Random Faults

- Faults are injected randomly into the cipher state, making them harder to exploit but still useful with statistical techniques.

Stuck-at Faults

- A particular bit or byte is forced to a fixed value (0 or 1).
- These are useful in differential analysis since the faulty value is deterministic.

Timing-Based Faults

- The attacker manipulates timing constraints (e.g., lowering voltage) to cause incorrect computations in the cipher.

AES Structure Relevant to DFA

AES-128 consists of 10 rounds, each including the following transformations:

- **SubBytes**: Byte-wise substitution using an S-Box.
- **ShiftRows**: Row-wise permutation.
- **MixColumns**: Column-wise mixing (except in the last round).
- **AddRoundKey**: XOR with the round key.

Since the final round of AES does not contain MixColumns, faults injected before or during this round propagate in a controlled manner, making analysis easier.

Attack Procedure

Step 1: Obtain Correct Ciphertext

Encrypt a plaintext P using the unknown secret key K to obtain the correct ciphertext C:

$$C = AES_K(P) \tag{1}$$

Step 2: Induce a Fault in the Last Round

Induce a fault in the state matrix during the SubBytes or ShiftRows stage of the last round, resulting in a faulty ciphertext C':

$$C' = AES_K^{\text{faulty}}(P) \tag{2}$$

The fault typically affects only one byte and propagates through the last round.

Step 3: Compute the Difference

Define the XOR difference between correct and faulty ciphertexts:

$$\Delta C = C \oplus C' \tag{3}$$

This difference propagates backward through the last round:

$$\Delta C = SBox^{-1}(C \oplus K_{10}) \oplus SBox^{-1}(C' \oplus K_{10}) \tag{4}$$

where K_{10} is the last round key and $SBox^{-1}$ is the AES inverse S-Box.

Step 4: Recover Last Round Key Bytes

Since only one byte is affected by the fault, each byte of K_{10} can be recovered independently:

1. Iterate over all 256 possible values of a key byte $K_{10}[i]$.
2. Compute the inverse S-Box output for both $C[i]$ and $C'[i]$.
3. Identify the correct key byte satisfying:

   $$SBox^{-1}(C[i] \oplus K_{10}[i]) \oplus SBox^{-1}(C'[i] \oplus K_{10}[i]) = \varepsilon \tag{5}$$

   where ε is the faul injected.
4. Repeat this process for all 16 bytes of K_{10}.

Step 5: Recover Full AES Key

Once K_{10} is obtained, the original AES key K is derived by inverting the AES key schedule:

$$K_{i-1} = \text{KeyExpansion}^{-1}(K_i) \tag{6}$$

By iterating this process backward from K_{10}, we can fully recover K.

Practical Considerations

- **Number of Faults Required**: Around 10-20 faulty ciphertexts are sufficient for full key recovery.
- **Fault Model**: The attack assumes a single-byte fault in the last round, common in laser-induced attacks on hardware.
- **Computational Complexity**: The attack requires at most 16×256 operations, which is practical.

Example Calculation

Example Setup

Assume a correct ciphertext byte:

$$C[i] = 0x3F \tag{7}$$

A faulty ciphertext byte:

$$C'[i] = 0x1B \tag{8}$$

XOR difference:

$$\Delta C[i] = C[i] \oplus C'[i] = 0x3F \oplus 0x1B = 0x24 \tag{9}$$

Key Byte Recovery

For all 256 possible values of $K_{10}[i]$:

1. Compute the inverse S-Box for $C[i] \oplus K_{10}[i]$ and $C'[i] \oplus K_{10}[i]$.
2. Check which key candidate satisfies:

$$SBox^{-1}(C[i] \oplus K_{10}[i]) \oplus SBox^{-1}(C'[i] \oplus K_{10}[i]) = \Delta C[i] \tag{10}$$

3. If a single key candidate remains, it is correct.