The Retracing Boomerang Attack

Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir
EUROCRYPT 2020

Gautam Singh

Indian Institute of Technology Hyderabad

July 21, 2025

Introduction

1. Broke the record for 5-round AES when it was published.

Introduction

1. Broke the record for 5-round AES when it was published.
2. Brings the attack complexity down to $2^{16.5}$ encryptions.

Introduction

1. Broke the record for 5-round AES when it was published.
2. Brings the attack complexity down to $2^{16.5}$ encryptions.
3. Uncovers a hidden relationship between boomerang attacks and two other cryptanalysis techniques: yoyo game and mixture differentials.

The Boomerang Attack

1. Typically split the encryption function as $E = E_1 \circ E_0$, with differential trails for each sub-cipher.

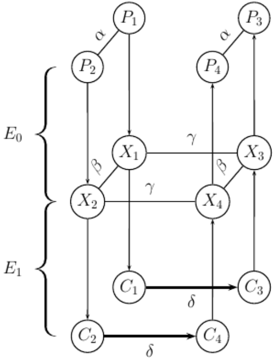

Figure 1: The boomerang attack.

The Boomerang Attack

1. Typically split the encryption function as $E = E_1 \circ E_0$, with differential trails for each sub-cipher.
2. We can build a distinguisher that can distinguish E from a truly random permutation in $\mathcal{O}((pq)^{-2})$ plaintext pairs.

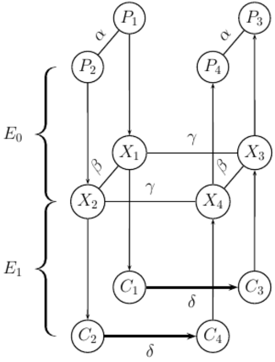

Figure 1: The boomerang attack.

The Boomerang Distinguisher

Algorithm 1 The Boomerang Attack Distinguisher

1: Generate $(pq)^{-2}$ plaintext pairs (P_1, P_2) such that $P_1 \oplus P_2 = \alpha$.
2: **for all** pairs (P_1, P_2) **do**
3: Ask for the encryption of (P_1, P_2) to (C_1, C_2).
4: Compute $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$. ▷ δ-*shift*
5: Ask for the decryption of (C_3, C_4) to (P_3, P_4).
6: **if** $P_3 \oplus P_4 = \alpha$ **then**
7: **return** This is the cipher E
8: **return** This is a random permutation

The Yoyo Game

1. Similar to boomerang, starts by encrypting (P_1, P_2) to (C_1, C_2), then modifying them to (C_3, C_4) and decrypting them.

The Yoyo Game

1. Similar to boomerang, starts by encrypting (P_1, P_2) to (C_1, C_2), then modifying them to (C_3, C_4) and decrypting them.
2. Unlike the boomerang attack, this *continues* in the yoyo game.

The Yoyo Game

1. Similar to boomerang, starts by encrypting (P_1, P_2) to (C_1, C_2), then modifying them to (C_3, C_4) and decrypting them.
2. Unlike the boomerang attack, this *continues* in the yoyo game.
3. All pairs of intermediate values (X_{2l+1}, X_{2l+2}) satisfy some property (such as zero difference in some part).

The Yoyo Game

1. Similar to boomerang, starts by encrypting (P_1, P_2) to (C_1, C_2), then modifying them to (C_3, C_4) and decrypting them.
2. Unlike the boomerang attack, this *continues* in the yoyo game.
3. All pairs of intermediate values (X_{2l+1}, X_{2l+2}) satisfy some property (such as zero difference in some part).
4. Probabilities are low with large l. Still, the yoyo technique has been used to attack AES reduced to 5 rounds.

Mixture

Definition 1 (Mixture)

Suppose $P_i \triangleq (\rho_1^i, \rho_2^i, \ldots, \rho_t^i)$. Given a plaintext pair (P_1, P_2), we say (P_3, P_4) is a *mixture counterpart* of (P_1, P_2) if for each $1 \leq j \leq t$, the quartet $(\rho_j^1, \rho_j^2, \rho_j^3, \rho_j^4)$ consists of two pairs of equal values or of four equal values. The quartet (P_1, P_2, P_3, P_4) is called a *mixture*.

Mixture

Definition 1 (Mixture)

Suppose $P_i \triangleq (\rho_1^i, \rho_2^i, \ldots, \rho_t^i)$. Given a plaintext pair (P_1, P_2), we say (P_3, P_4) is a *mixture counterpart* of (P_1, P_2) if for each $1 \leq j \leq t$, the quartet $(\rho_j^1, \rho_j^2, \rho_j^3, \rho_j^4)$ consists of two pairs of equal values or of four equal values. The quartet (P_1, P_2, P_3, P_4) is called a *mixture*.

1. If (P_1, P_2, P_3, P_4) is a mixture, then XOR of the intermediate values (X_1, X_2, X_3, X_4) is zero.

Mixture

Definition 1 (Mixture)

Suppose $P_i \triangleq (\rho_1^i, \rho_2^i, \ldots, \rho_t^i)$. Given a plaintext pair (P_1, P_2), we say (P_3, P_4) is a *mixture counterpart* of (P_1, P_2) if for each $1 \leq j \leq t$, the quartet $(\rho_j^1, \rho_j^2, \rho_j^3, \rho_j^4)$ consists of two pairs of equal values or of four equal values. The quartet (P_1, P_2, P_3, P_4) is called a *mixture*.

1. If (P_1, P_2, P_3, P_4) is a mixture, then XOR of the intermediate values (X_1, X_2, X_3, X_4) is zero.
2. $X_1 \oplus X_3 = \gamma \implies X_2 \oplus X_4 = \gamma$. Hence, for $\gamma \xrightarrow{q} \delta$ in E_1, $C_1 \oplus C_3 = C_2 \oplus C_4 = \delta$ with probability q^2.

The SimpleSWAP Algorithm

Algorithm 2 is a simple method to generate mixture counterparts.

Algorithm 2 Swaps the first word where texts are different and returns one word.

function SIMPLESWAP(x^0, x^1) $\triangleright$ $x^0 \neq x^1$
  $x'^0, x'^1 \leftarrow x^0, x^1$
  for i from 0 to 3 **do**
    if $x_i^0 \neq x_i^1$ **then**
      $x_i'^0, x_i'^1 \leftarrow x_i^1, x_i^0$
      return x'^0, x'^1

The Retracing Boomerang Framework

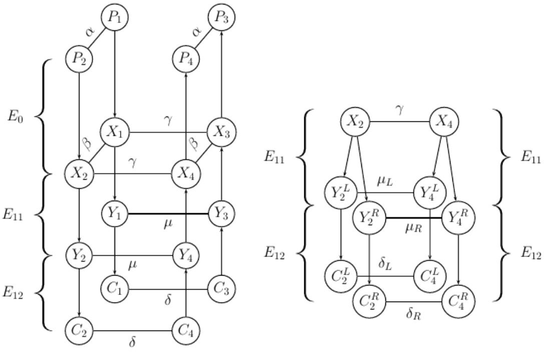

Figure 2: The retracing boomerang attack.

The Retracing Boomerang Attack

1. The *retracing boomerang* framework consists of a *shifting* type and a *mixing* type.

The Retracing Boomerang Attack

1. The *retracing boomerang* framework consists of a *shifting* type and a *mixing* type.
2. Both attacks use the setup shown in Figure 2.

The Retracing Boomerang Attack

1. The *retracing boomerang* framework consists of a *shifting* type and a *mixing* type.
2. Both attacks use the setup shown in Figure 2.
3. Although the additional split looks restrictive, it applies for a wide class of block ciphers such as SASAS constructions.

The Retracing Boomerang Attack

1. The *retracing boomerang* framework consists of a *shifting* type and a *mixing* type.
2. Both attacks use the setup shown in Figure 2.
3. Although the additional split looks restrictive, it applies for a wide class of block ciphers such as SASAS constructions.
4. It is assumed that E_{12} can be split into two parts of size b and $n-b$ bits, call these functions E_{12}^{L} and E_{12}^{R}, with characteristic probabilities q_2^L and q_2^R respectively.

The Shifting Retracing Boomerang Attack

1. Check if $C_1^L \oplus C_2^L = 0$ or δ_L. *Discard all pairs not satisfying this relation*. This is a $(b-1)$-bit filtering.

The Shifting Retracing Boomerang Attack

1. Check if $C_1^L \oplus C_2^L = 0 \text{ or } \delta_L$. *Discard all pairs not satisfying this relation*. This is a $(b-1)$-bit filtering.
2. δ-shift is performed on the filtered ciphertext pairs to get (C_3, C_4). This ensures $\{C_1, C_3\} = \{C_2, C_4\}$.

The Shifting Retracing Boomerang Attack

1. Check if $C_1^L \oplus C_2^L = 0 \text{ or } \delta_L$. *Discard all pairs not satisfying this relation.* This is a $(b-1)$-bit filtering.
2. δ-shift is performed on the filtered ciphertext pairs to get (C_3, C_4). This ensures $\{C_1, C_3\} = \{C_2, C_4\}$.
3. *If one of these pairs satisfies $\delta_L \xrightarrow{q_2^L} \mu_L$, the other pair will too!*. Increases the probability of the boomerang distinguisher by $(q_2^L)^{-1}$.

The Shifting Retracing Boomerang Attack

1. Check if $C_1^L \oplus C_2^L = 0 \text{ or } \delta_L$. *Discard all pairs not satisfying this relation*. This is a $(b-1)$-bit filtering.
2. δ-shift is performed on the filtered ciphertext pairs to get (C_3, C_4). This ensures $\{C_1, C_3\} = \{C_2, C_4\}$.
3. *If one of these pairs satisfies $\delta_L \xrightarrow{q_2^L} \mu_L$, the other pair will too!*. Increases the probability of the boomerang distinguisher by $(q_2^L)^{-1}$.
4. Any possible characteristic of E_{12}^L has probability at least 2^{-b+1}, thus overall probability increases by a factor of at most 2^{b-1}. On the other hand, filtering only leaves 2^{-b+1} of the pairs, so *no apparent gain?*

The Shifting Retracing Boomerang Attack

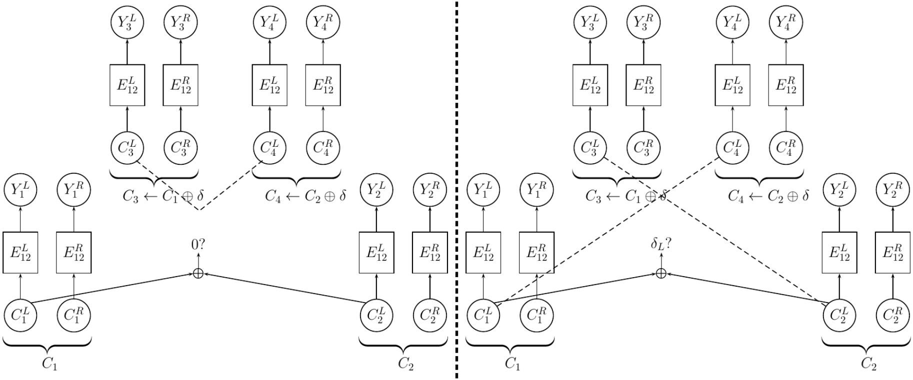

Figure 3: A shifted quartet (dashed lines indicate equality).

The Mixing Retracing Boomerang Attack

1. In shifting attack, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$ forced using a δ-shift.

The Mixing Retracing Boomerang Attack

1. In shifting attack, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$ forced using a δ-shift.
2. Each ciphertext shifted by $(C_1^L \oplus C_2^L, 0)$, thus

$$C_3 = (C_3^L, C_3^R) = (C_1^L \oplus (C_1^L \oplus C_2^L), C_1^R) = (C_2^L, C_1^R), \tag{1}$$
$$C_4 = (C_4^L, C_4^R) = (C_2^L \oplus (C_1^L \oplus C_2^L), C_2^R) = (C_1^L, C_2^R). \tag{2}$$

Here, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$.

The Mixing Retracing Boomerang Attack

1. In shifting attack, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$ forced using a δ-shift.
2. Each ciphertext shifted by $(C_1^L \oplus C_2^L, 0)$, thus

$$C_3 = (C_3^L, C_3^R) = (C_1^L \oplus (C_1^L \oplus C_2^L), C_1^R) = (C_2^L, C_1^R), \tag{1}$$
$$C_4 = (C_4^L, C_4^R) = (C_2^L \oplus (C_1^L \oplus C_2^L), C_2^R) = (C_1^L, C_2^R). \tag{2}$$

   Here, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$.
3. Further, $C_1^R = C_3^R$ and $C_2^R = C_4^R$. *Additional* gain of $(q_2^R)^{-2}$ for total probability $(pq_1)^2 q_2^L$, *better than shifting!*

The Mixing Retracing Boomerang Attack

1. In shifting attack, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$ forced using a δ-shift.
2. Each ciphertext shifted by $(C_1^L \oplus C_2^L, 0)$, thus

$$C_3 = (C_3^L, C_3^R) = (C_1^L \oplus (C_1^L \oplus C_2^L), C_1^R) = (C_2^L, C_1^R), \quad (1)$$
$$C_4 = (C_4^L, C_4^R) = (C_2^L \oplus (C_1^L \oplus C_2^L), C_2^R) = (C_1^L, C_2^R). \quad (2)$$

   Here, $\{C_1^L, C_2^L\} = \{C_3^L, C_4^L\}$.

3. Further, $C_1^R = C_3^R$ and $C_2^R = C_4^R$. *Additional* gain of $(q_2^R)^{-2}$ for total probability $(pq_1)^2 q_2^L$, *better than shifting!*
4. Similar to the core step used in the yoyo attack on AES.

The Mixing Retracing Boomerang Attack

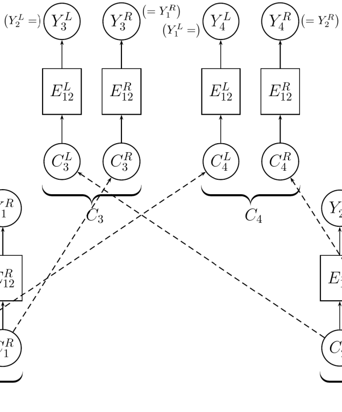

Figure 4: A mixture quartet of ciphertexts (dashed lines indicate equality).

Advantages of Shifting Retracing Attack

1. **Using structures**

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**
   - In mixing, the output difference of E_{12}^L is arbitrary.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**
   - In mixing, the output difference of E_{12}^L is arbitrary.
   - Usually no good combination between characteristics of $(E_{12}^L)^{-1}$ and $(E_{11})^{-1}$. For instance, in the yoyo attack, E_{11} is empty.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**
   - In mixing, the output difference of E_{12}^{L} is arbitrary.
   - Usually no good combination between characteristics of $(E_{12}^{L})^{-1}$ and $(E_{11})^{-1}$. For instance, in the yoyo attack, E_{11} is empty.
3. **Construction of 'friend pairs'**

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**
   - In mixing, the output difference of E_{12}^L is arbitrary.
   - Usually no good combination between characteristics of $(E_{12}^L)^{-1}$ and $(E_{11})^{-1}$. For instance, in the yoyo attack, E_{11} is empty.
3. **Construction of 'friend pairs'**
   - 'Friend pairs' are pairs which satisfy a common property.

Advantages of Shifting Retracing Attack

1. **Using structures**
   - Shifting applies the same δ-shift to all pairs of ciphertexts.
   - Filtering applied first reduces the data complexity. No filtering in mixing since shift is based on ciphertexts.
   - With shifting, one can obtain all ciphertexts, shift them by δ and then decrypt, simultaneously checking for the filter and condition between P_3 and P_4 using a hash table.
2. **Combination with E_{11}**
   - In mixing, the output difference of E_{12}^L is arbitrary.
   - Usually no good combination between characteristics of $(E_{12}^L)^{-1}$ and $(E_{11})^{-1}$. For instance, in the yoyo attack, E_{11} is empty.
3. **Construction of 'friend pairs'**
   - 'Friend pairs' are pairs which satisfy a common property.
   - More 'friend pairs' can be constructed in the shifting variant.

Description of AES

1. Byte ordering shown after *SB* in Figure 5 (column major).

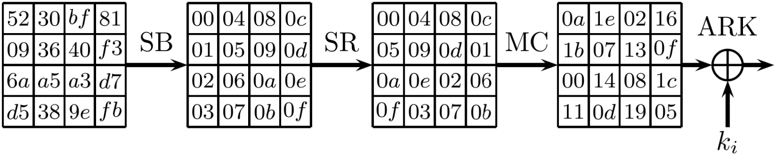

Figure 5: An AES round.

Description of AES

1. Byte ordering shown after *SB* in Figure 5 (column major).
2. j-th byte of a state X_i is denoted as $X_{i,j}$ or $(X_i)_j$.

$$\begin{array}{|c|c|c|c|}\hline 52 & 30 & bf & 81 \\ \hline 09 & 36 & 40 & f3 \\ \hline 6a & a5 & a3 & d7 \\ \hline d5 & 38 & 9e & fb \\ \hline \end{array} \xrightarrow{\text{SB}} \begin{array}{|c|c|c|c|}\hline 00 & 04 & 08 & 0c \\ \hline 01 & 05 & 09 & 0d \\ \hline 02 & 06 & 0a & 0e \\ \hline 03 & 07 & 0b & 0f \\ \hline \end{array} \xrightarrow{\text{SR}} \begin{array}{|c|c|c|c|}\hline 00 & 04 & 08 & 0c \\ \hline 05 & 09 & 0d & 01 \\ \hline 0a & 0e & 02 & 06 \\ \hline 0f & 03 & 07 & 0b \\ \hline \end{array} \xrightarrow{\text{MC}} \begin{array}{|c|c|c|c|}\hline 0a & 1e & 02 & 16 \\ \hline 1b & 07 & 13 & 0f \\ \hline 00 & 14 & 08 & 1c \\ \hline 11 & 0d & 19 & 05 \\ \hline \end{array} \xrightarrow{} \underset{\uparrow \atop k_i}{\overset{\text{ARK}}{\oplus}} \rightarrow$$

Figure 5: An AES round.

Description of AES

1. Byte ordering shown after *SB* in Figure 5 (column major).
2. j-th byte of a state X_i is denoted as $X_{i,j}$ or $(X_i)_j$.
3. Denote by W, Z and X the states before MC in round 0, at the input to round 1 and before MC in round 2 respectively.

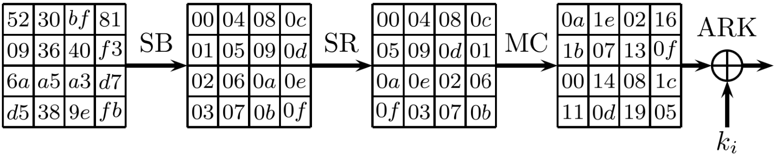

Figure 5: An AES round.

Description of AES

1. Byte ordering shown after SB in Figure 5 (column major).
2. j-th byte of a state X_i is denoted as $X_{i,j}$ or $(X_i)_j$.
3. Denote by W, Z and X the states before MC in round 0, at the input to round 1 and before MC in round 2 respectively.
4. The l-th shifted column (resp. l-th inverse shifted column) refers to application of SR (resp. SR^{-1}) to the l-th column.

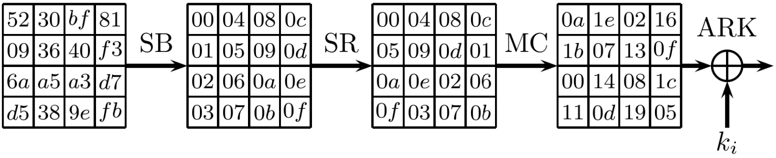

Figure 5: An AES round.

Description of AES

1. Byte ordering shown after SB in Figure 5 (column major).
2. j-th byte of a state X_i is denoted as $X_{i,j}$ or $(X_i)_j$.
3. Denote by W, Z and X the states before MC in round 0, at the input to round 1 and before MC in round 2 respectively.
4. The l-th shifted column (resp. l-th inverse shifted column) refers to application of SR (resp. SR^{-1}) to the l-th column.
5. Round subkeys are $k_{-1}, k_0, \ldots$.

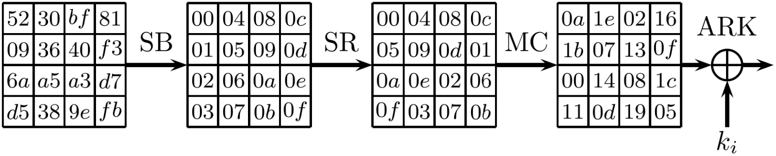

Figure 5: An AES round.

Summary of Yoyo Attack on Five Round AES

1. Decomposes AES as $E = E_{12} \circ E_{11} \circ E_0$ where E_0 is the first 2.5 rounds, E_{11} is the MC of round 2 and E_{12} is the last 2 rounds.

Summary of Yoyo Attack on Five Round AES

1. Decomposes AES as $E = E_{12} \circ E_{11} \circ E_0$ where E_0 is the first 2.5 rounds, E_{11} is the MC of round 2 and E_{12} is the last 2 rounds.
2. Truncated differential characteristic for E_0: zero input difference in three inverse shifted columns and zero output difference in a single shifted column with probability $4 \cdot 2^{-8} = 2^{-6}$.

Summary of Yoyo Attack on Five Round AES

1. Decomposes AES as $E = E_{12} \circ E_{11} \circ E_0$ where E_0 is the first 2.5 rounds, E_{11} is the MC of round 2 and E_{12} is the last 2 rounds.
2. Truncated differential characteristic for E_0: zero input difference in three inverse shifted columns and zero output difference in a single shifted column with probability $4 \cdot 2^{-8} = 2^{-6}$.
3. For E_{12}, 1.5 rounds of AES can be taken as four 32-bit super S-boxes.

Summary of Yoyo Attack on Five Round AES

1. Decomposes AES as $E = E_{12} \circ E_{11} \circ E_0$ where E_0 is the first 2.5 rounds, E_{11} is the MC of round 2 and E_{12} is the last 2 rounds.
2. Truncated differential characteristic for E_0: zero input difference in three inverse shifted columns and zero output difference in a single shifted column with probability $4 \cdot 2^{-8} = 2^{-6}$.
3. For E_{12}, 1.5 rounds of AES can be taken as four 32-bit super S-boxes.
4. Attack inverse shifted columns of k_{-1}. Friend pairs used to get more information.

Meet in the Middle Improvement on Yoyo Attack

1. Denote the value of byte m before MC operation of round 0 by W_m, and WLOG let $l = 0$. Then,

$$Z_0 = 02_x \cdot W_0 \oplus 03_x \cdot W_1 \oplus 01_x \cdot W_2 \oplus 01_x \cdot W_3. \tag{3}$$

Meet in the Middle Improvement on Yoyo Attack

1. Denote the value of byte m before MC operation of round 0 by W_m, and WLOG let $l = 0$. Then,

$$Z_0 = 02_x \cdot W_0 \oplus 03_x \cdot W_1 \oplus 01_x \cdot W_2 \oplus 01_x \cdot W_3. \tag{3}$$

2. Adversary guesses $k_{-1,\{0,5\}}$ by computing the following for $j = 1, 2, 3$ and storing the concatenated 24-bit value in a hash table.

$$02_x \cdot ((W_3^j)_0 \oplus (W_4^j)_0) \oplus 03_x \cdot ((W_3^j)_1 \oplus (W_4^j)_1) \tag{4}$$

Meet in the Middle Improvement on Yoyo Attack

1. Denote the value of byte m before MC operation of round 0 by W_m, and WLOG let $l = 0$. Then,

$$Z_0 = 02_x \cdot W_0 \oplus 03_x \cdot W_1 \oplus 01_x \cdot W_2 \oplus 01_x \cdot W_3. \tag{3}$$

2. Adversary guesses $k_{-1,\{0,5\}}$ by computing the following for $j = 1, 2, 3$ and storing the concatenated 24-bit value in a hash table.

$$02_x \cdot ((W_3^j)_0 \oplus (W_4^j)_0) \oplus 03_x \cdot ((W_3^j)_1 \oplus (W_4^j)_1) \tag{4}$$

3. We need $Z_0 = 0$ to satisfy the truncated differential characteristic. Meet in the Middle (MITM) methods are used to narrow down candidates for k_{-1}.

Meet in the Middle Improvement on Yoyo Attack

1. Denote the value of byte m before MC operation of round 0 by W_m, and WLOG let $l = 0$. Then,

$$Z_0 = 02_x \cdot W_0 \oplus 03_x \cdot W_1 \oplus 01_x \cdot W_2 \oplus 01_x \cdot W_3. \tag{3}$$

2. Adversary guesses $k_{-1,\{0,5\}}$ by computing the following for $j = 1, 2, 3$ and storing the concatenated 24-bit value in a hash table.

$$02_x \cdot ((W_3^j)_0 \oplus (W_4^j)_0) \oplus 03_x \cdot ((W_3^j)_1 \oplus (W_4^j)_1) \tag{4}$$

3. We need $Z_0 = 0$ to satisfy the truncated differential characteristic. Meet in the Middle (MITM) methods are used to narrow down candidates for k_{-1}.
   - Specific choice of plaintexts based on DDT of AES S-boxes.

Meet in the Middle Improvement on Yoyo Attack

1. Denote the value of byte m before MC operation of round 0 by W_m, and WLOG let $l = 0$. Then,

$$Z_0 = 02_x \cdot W_0 \oplus 03_x \cdot W_1 \oplus 01_x \cdot W_2 \oplus 01_x \cdot W_3. \tag{3}$$

2. Adversary guesses $k_{-1,\{0,5\}}$ by computing the following for $j = 1, 2, 3$ and storing the concatenated 24-bit value in a hash table.

$$02_x \cdot ((W_3^j)_0 \oplus (W_4^j)_0) \oplus 03_x \cdot ((W_3^j)_1 \oplus (W_4^j)_1) \tag{4}$$

3. We need $Z_0 = 0$ to satisfy the truncated differential characteristic. Meet in the Middle (MITM) methods are used to narrow down candidates for k_{-1}.
   - Specific choice of plaintexts based on DDT of AES S-boxes.
   - Eliminating key bytes using friend pairs.

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \tag{5}$$

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \tag{5}$$

2. Constrain $(P_1)_5 \oplus (P_2)_5 = 01_x$ to detect right key bytes efficiently.

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \qquad (5)$$

2. Constrain $(P_1)_5 \oplus (P_2)_5 = 01_x$ to detect right key bytes efficiently.
3. DDT row of AES S-box for input difference 01_x along with input pair(s) for each output difference computed and stored in memory.

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \qquad (5)$$

2. Constrain $(P_1)_5 \oplus (P_2)_5 = 01_x$ to detect right key bytes efficiently.
3. DDT row of AES S-box for input difference 01_x along with input pair(s) for each output difference computed and stored in memory.
4. For each (P_1, P_2) and for each guess of $k_{-1,0}$, use (5) to compute the output difference of the SB operation in byte 5.

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \tag{5}$$

2. Constrain $(P_1)_5 \oplus (P_2)_5 = 01_x$ to detect right key bytes efficiently.
3. DDT row of AES S-box for input difference 01_x along with input pair(s) for each output difference computed and stored in memory.
4. For each (P_1, P_2) and for each guess of $k_{-1,0}$, use (5) to compute the output difference of the SB operation in byte 5.
5. Lookup to find inputs that can lead to this difference and retrieve possible values of $k_{-1,5}$ corresponding to the guessed $k_{-1,0}$.

Specific Choice of Plaintexts

1. Choose plaintexts with non-zero difference *only in bytes 0 and 5.* Here, $(Z_1)_0 = (Z_2)_0$ leaves 2^8 candidates for $k_{-1,\{0,5\}}$, given by

$$02_x \cdot ((W_1)_0 \oplus (W_2)_0) \oplus 03_x \cdot ((W_1)_1 \oplus (W_2)_1) = 0. \qquad (5)$$

2. Constrain $(P_1)_5 \oplus (P_2)_5 = 01_x$ to detect right key bytes efficiently.
3. DDT row of AES S-box for input difference 01_x along with input pair(s) for each output difference computed and stored in memory.
4. For each (P_1, P_2) and for each guess of $k_{-1,0}$, use (5) to compute the output difference of the SB operation in byte 5.
5. Lookup to find inputs that can lead to this difference and retrieve possible values of $k_{-1,5}$ corresponding to the guessed $k_{-1,0}$.
6. Obtain 2^8 candidates for $k_{-1,\{0,5\}}$ in about 2^8 operations per pair.

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

3. If equality holds in byte 10, then $k_{-1,15}$ is isolated for a fixed $k_{-1,\{0,5\}}$ and has only 2^8 possible values.

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

3. If equality holds in byte 10, then $k_{-1,15}$ is isolated for a fixed $k_{-1,\{0,5\}}$ and has only 2^8 possible values.
4. Requires 2^9 simple operations and leaves 2^8 candidates for $k_{-1,\{0,5,15\}}$.

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

3. If equality holds in byte 10, then $k_{-1,15}$ is isolated for a fixed $k_{-1,\{0,5\}}$ and has only 2^8 possible values.
4. Requires 2^9 simple operations and leaves 2^8 candidates for $k_{-1,\{0,5,15\}}$.
5. Similar MITM procedure followed with another friend pair to obtain the unique value of $k_{-1,\{0,5,10,15\}}$ by isolating $k_{-1,10}$.

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

3. If equality holds in byte 10, then $k_{-1,15}$ is isolated for a fixed $k_{-1,\{0,5\}}$ and has only 2^8 possible values.
4. Requires 2^9 simple operations and leaves 2^8 candidates for $k_{-1,\{0,5,15\}}$.
5. Similar MITM procedure followed with another friend pair to obtain the unique value of $k_{-1,\{0,5,10,15\}}$ by isolating $k_{-1,10}$.
6. Perform 2^8 operations for each pair (P_1, P_2) and for each value of l. Total time complexity of about 2^{16} operations.

Eliminating Key Bytes Using Friend Pairs

1. To reduce the number of candidates for $k_{-1,\{10,15\}}$, the boomerang process is used to return multiple friend pairs (P_3^j, P_4^j).
2. In particular, we choose one such pair for which

$$(P_3^j)_{10} \oplus (P_4^j)_{10} = 0 \quad \text{or} \quad (P_3^j)_{15} \oplus (P_4^j)_{15} = 0. \tag{6}$$

3. If equality holds in byte 10, then $k_{-1,15}$ is isolated for a fixed $k_{-1,\{0,5\}}$ and has only 2^8 possible values.
4. Requires 2^9 simple operations and leaves 2^8 candidates for $k_{-1,\{0,5,15\}}$.
5. Similar MITM procedure followed with another friend pair to obtain the unique value of $k_{-1,\{0,5,10,15\}}$ by isolating $k_{-1,10}$.
6. Perform 2^8 operations for each pair (P_1, P_2) and for each value of l. Total time complexity of about 2^{16} operations.
7. Each pair requires 2^7 friend pairs to find one that satisfies (6) with high probability. Total data complexity is increased to about 2^{15}.

Attack Algorithm

1. **Precomputation:** Compute DDT row of AES S-box for input difference 01_x, along with actual inputs for each output difference.

Attack Algorithm

1. **Precomputation:** Compute DDT row of AES S-box for input difference 01_x, along with actual inputs for each output difference.
2. **Online Phase:** Take 64 pairs (P_1, P_2) with $(P_1)_5 = 00_x$, $(P_2)_5 = 01_x$, $(P_1)_0 \neq (P_2)_0$ and all other corresponding bytes equal.

Attack Algorithm

1. **Precomputation:** Compute DDT row of AES S-box for input difference 01_x, along with actual inputs for each output difference.
2. **Online Phase:** Take 64 pairs (P_1, P_2) with $(P_1)_5 = 00_x$, $(P_2)_5 = 01_x$, $(P_1)_0 \neq (P_2)_0$ and all other corresponding bytes equal.
3. For each plaintext pair, create 2^7 friend pairs (P_1^j, P_2^j) such that for each j, $P_1^j \oplus P_2^j = P_1 \oplus P_2$ and $(P_1^j)_{\{0,5,10,15\}} = (P_1)_{\{0,5,10,15\}}$.

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
   1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
    1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.
    2. Use the boomerang process to obtain pairs (P_3, P_4) and (P_3^j, P_4^j).

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
   1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.
   2. Use the boomerang process to obtain pairs (P_3, P_4) and (P_3^j, P_4^j).
   3. Find a j for which (6) is satisfied. Perform an MITM attack on column 0 of round 0 using (P_3^j, P_4^j) to obtain 2^8 candidates for $k_{-1,\{0,5,15\}}$.

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
   1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.
   2. Use the boomerang process to obtain pairs (P_3, P_4) and (P_3^j, P_4^j).
   3. Find a j for which (6) is satisfied. Perform an MITM attack on column 0 of round 0 using (P_3^j, P_4^j) to obtain 2^8 candidates for $k_{-1,\{0,5,15\}}$.
   4. Perform another MITM attack on column 0 of round 0 using two plaintext pairs $(P_3^{j'}, P_4^{j'})$. This gives a possible value for $k_{-1,\{0,5,10,15\}}$.

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
   1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.
   2. Use the boomerang process to obtain pairs (P_3, P_4) and (P_3^j, P_4^j).
   3. Find a j for which (6) is satisfied. Perform an MITM attack on column 0 of round 0 using (P_3^j, P_4^j) to obtain 2^8 candidates for $k_{-1,\{0,5,15\}}$.
   4. Perform another MITM attack on column 0 of round 0 using two plaintext pairs $(P_3^{j'}, P_4^{j'})$. This gives a possible value for $k_{-1,\{0,5,10,15\}}$.
   5. If contradiction, go to the next value of l. If contradiction for all l, discard this pair and go to the next pair.

Attack Algorithm

4. For each plaintext pair (P_1, P_2) and for each $l \in \{0, 1, 2, 3\}$, do the following. ($l = 0$ taken below)
    1. Use (5) to compute and store all 2^8 candidates for $k_{-1,\{0,5\}}$ in a table.
    2. Use the boomerang process to obtain pairs (P_3, P_4) and (P_3^j, P_4^j).
    3. Find a j for which (6) is satisfied. Perform an MITM attack on column 0 of round 0 using (P_3^j, P_4^j) to obtain 2^8 candidates for $k_{-1,\{0,5,15\}}$.
    4. Perform another MITM attack on column 0 of round 0 using two plaintext pairs $(P_3^{j'}, P_4^{j'})$. This gives a possible value for $k_{-1,\{0,5,10,15\}}$.
    5. If contradiction, go to the next value of l. If contradiction for all l, discard this pair and go to the next pair.
5. Using a pair (P_1, P_2) for which no contradiction occurred, perform MITM attacks on columns 1, 2 and 3 of round 0 using the fact that $Z_3 \oplus Z_4$ equals 0 in the l-th inverse shifted column to recover k_{-1}.

Attack Analysis

1. Attack succeeds if data contains a pair that satisfies the truncated differential characteristic of E_0 and if a friend pair has zero difference in either byte 10 or 15.

Attack Analysis

1. Attack succeeds if data contains a pair that satisfies the truncated differential characteristic of E_0 and if a friend pair has zero difference in either byte 10 or 15.
2. Increasing the number of initial pairs and friend pairs per initial pair boosts success probability. With 64 pairs and 128 friend pairs per initial pair, the probability of success is $(1 - e^{-1})^2 \approx 0.4$

Attack Analysis

1. Attack succeeds if data contains a pair that satisfies the truncated differential characteristic of E_0 and if a friend pair has zero difference in either byte 10 or 15.
2. Increasing the number of initial pairs and friend pairs per initial pair boosts success probability. With 64 pairs and 128 friend pairs per initial pair, the probability of success is $(1 - e^{-1})^2 \approx 0.4$
3. Another way to boost succees probability is to find other ways to cancel terms in (3). For instance, if there exist j, j' such that $\{(P_3^j)_{10}, (P_4^j)_{10}\} = \{(P_3^{j'})_{10}, (P_4^{j'})_{10}\}$, we can take the XOR of (3) to cancel the effect of $k_{-1,10}$, thus increasing the success probability even when there is no pair that satisfies (6).

Attack Analysis

4. Data complexity is $2 \cdot 2^6 \cdot 2^7 = 2^{14}$ chosen plaintexts and 2^{14} adaptively chosen ciphertexts.

Attack Analysis

4. Data complexity is $2 \cdot 2^6 \cdot 2^7 = 2^{14}$ chosen plaintexts and 2^{14} adaptively chosen ciphertexts.
5. Structures reduce the data complexity to slightly above 2^{14} adaptively chosen ciphertexts and plaintexts, but success probability slightly reduced due to additional dependencies between analyzed pairs.

Attack Analysis

4. Data complexity is $2 \cdot 2^6 \cdot 2^7 = 2^{14}$ chosen plaintexts and 2^{14} adaptively chosen ciphertexts.
5. Structures reduce the data complexity to slightly above 2^{14} adaptively chosen ciphertexts and plaintexts, but success probability slightly reduced due to additional dependencies between analyzed pairs.
6. Memory complexity of the attack remains at 2^9, like yoyo attack.

Attack Analysis

4. Data complexity is $2 \cdot 2^6 \cdot 2^7 = 2^{14}$ chosen plaintexts and 2^{14} adaptively chosen ciphertexts.
5. Structures reduce the data complexity to slightly above 2^{14} adaptively chosen ciphertexts and plaintexts, but success probability slightly reduced due to additional dependencies between analyzed pairs.
6. Memory complexity of the attack remains at 2^9, like yoyo attack.
7. Time complexity dominated by MITM attacks that take 2^{16} operations each. Taking one AES operation equivalent to 80 S-box lookups and adding it to the number of queries gives us a total of $2^{16.5}$ encryptions.