How to Efficiently and Simultaneously Compute the Probabilities of All Iterative Characteristics

Itai Dinur, Orr Dunkelman, Nathan Keller, Adi Shamir

Computer Science Department
University of Haifa

28th May, 2013

Differential Cryptanalysis

- Introduced by Biham and Shamir [BS90].
- Studies the development of differences through the encryption function.
- Based on differential characteristics $(\Omega_P \rightarrow \Omega_1 \rightarrow \Omega_2 \rightarrow \cdots \rightarrow \Omega_C)$
- Easy to compute differential transitions through linear (affine) operations.
- S-boxes, require to start looking at probabilities ...

Differential Cryptanalysis

- Introduced by Biham and Shamir [BS90].
- Studies the development of differences through the encryption function.
- Based on differential characteristics $(\Omega_P \to \Omega_1 \to \Omega_2 \to \cdots \to \Omega_C)$
- Easy to compute differential transitions through linear (affine) operations.
- S-boxes, require to start looking at probabilities ...

Wait for tomorrow's IACR distinguished lecture by Eli!

Difference Distribution Tables

- The difference distribution table of an S-box counts how many pairs with a given input difference lead to a given output difference.
- It is an essential part of identifying high-probability transitions for constructing the differential characteristics, later used in the attack.
- (Sometimes, the table also stores the actual pairs, and not only their number).

The Difference Distribution Tables of DES $S1$

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
								⋮								
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

The Diagonal of the Difference Distribution Table

- The diagonal contains all the differences that are copied to themselves.

The Diagonal of the Difference Distribution Table

- The diagonal contains all the differences that are copied to themselves.
- In other words, iterative differences.

The Diagonal of the Difference Distribution Table

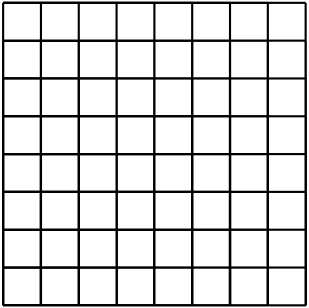
- The diagonal contains all the differences that are copied to themselves.
- In other words, iterative differences.
- These differences are extremely important in block cipher cryptanalysis (wait for Eli's talk, read Eli's & Adi's book, or just ask a friendly cryptanalyst).

The Diagonal of the Difference Distribution Table

- The diagonal contains all the differences that are copied to themselves.
- In other words, iterative differences.
- These differences are extremely important in block cipher cryptanalysis (wait for Eli's talk, read Eli's & Adi's book, or just ask a friendly cryptanalyst).
- Also very useful for the cryptanalysis of hash functions.

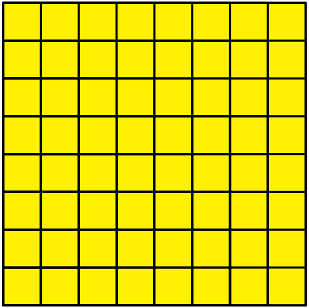
Constructing the Full DDT

- For an n-bit to n-bit S-box $S[\cdot]$, one can run the following algorithm:
  1. Set $DDT[i][j] \leftarrow 0$ for all i, j
  2. For all x
     - For all y: $DDT[x \oplus y][S[x] \oplus S[y]] + +$
- Time: $O(2^{2n})$, Memory: $O(2^n)$



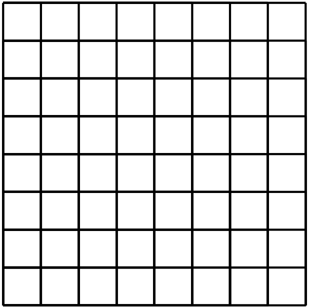
Constructing the Full DDT

- For an n-bit to n-bit S-box $S[\cdot]$, one can run the following algorithm:
  1. Set $DDT[i][j] \leftarrow 0$ for all i, j
  2. For all x
     - For all y: $DDT[x \oplus y][S[x] \oplus S[y]] + +$
- Time: $O(2^{2n})$, Memory: $O(2^n)$



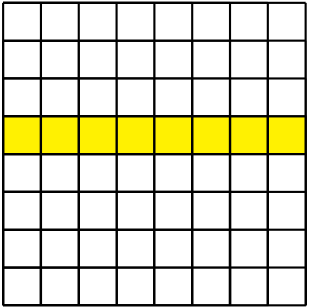
Constructing the Row/Column of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input difference Δ_{IN} one can run the following algorithm:
  1. For all j's set $DDT[\Delta_{IN}][j] \leftarrow 0$
  2. For all x:
     - $DDT[\Delta_{IN}][S[x] \oplus S[x \oplus \Delta_{IN}] + +$
- Time: $O(2^n)$, Memory: $O(2^n)$



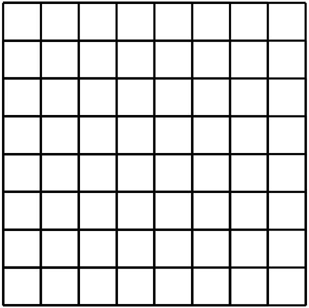
Constructing the Row/Column of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input difference Δ_{IN} one can run the following algorithm:
  1. For all j's set $DDT[\Delta_{IN}][j] \leftarrow 0$
  2. For all x:
     - $DDT[\Delta_{IN}][S[x] \oplus S[x \oplus \Delta_{IN}] + +$
- Time: $O(2^n)$, Memory: $O(2^n)$



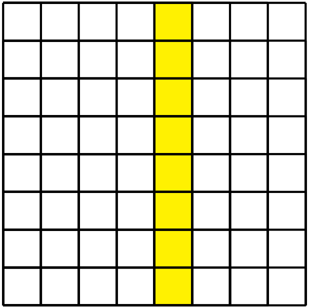
Constructing the Row/Column of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input difference Δ_{OUT} one can run the following algorithm:
  1. For all j's set $DDT[j][\Delta_{OUT}] \leftarrow 0$
  2. For all x:
     - $DDT[S^{-1}[x] \oplus S^{-1}[x \oplus \Delta_{OUT}]][\Delta_{OUT}] + +$
- Time: $O(2^n)$, Memory: $O(2^n)$



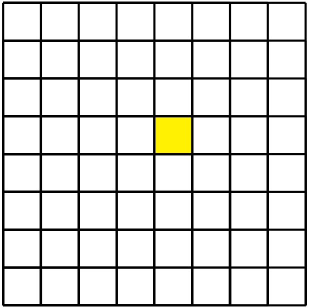
Constructing the Row/Column of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input difference Δ_{OUT} one can run the following algorithm:
  1. For all j's set $DDT[j][\Delta_{OUT}] \leftarrow 0$
  2. For all x:
     - $DDT[S^{-1}[x] \oplus S^{-1}[x \oplus \Delta_{OUT}]][\Delta_{OUT}] + +$
- Time: $O(2^n)$, Memory: $O(2^n)$



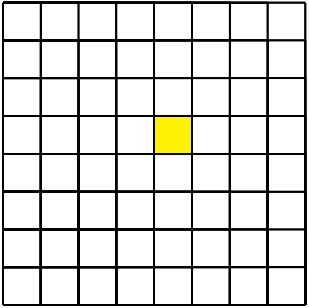
Constructing an Entry of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input/output difference pair $(\Delta_{IN}, \Delta_{OUT})$:
  1. Set $DDT[\Delta_{IN}][\Delta_{OUT}] \leftarrow 0$
  2. For all x:
     - If $S[x] \oplus S[x \oplus \Delta_{IN}] = \Delta_{OUT}]$ $DDT[\Delta_{IN}][\Delta_{OUT}]++$



Constructing an Entry of the DDT

- For an n-bit to n-bit S-box $S[\cdot]$, and an input/output difference pair $(\Delta_{IN}, \Delta_{OUT})$:
  1. Set $DDT[\Delta_{IN}][\Delta_{OUT}] \leftarrow 0$
  2. For all x:
     - If $S[x] \oplus S[x \oplus \Delta_{IN}] = \Delta_{OUT}]$ $DDT[\Delta_{IN}][\Delta_{OUT}]++$
- Time: $O(2^n)$, Memory: $O(1)$



How to Construct the Diagonal

1. Construct the entire DDT ($O(2^{2n})$ time and $O(2^n)$ memory).
2. Construct only the diagonal's entries ($O(2^{2n})$ time and $O(n)$ memory).

Diagonal Entries

- Diagonal entries (Δ, Δ) correspond to $i \oplus j = S[i] \oplus S[j]$.

Diagonal Entries

- Diagonal entries (Δ, Δ) correspond to $i \oplus j = S[i] \oplus S[j]$.
- Or $i \oplus S[i] = j \oplus S[j] \ldots$

Diagonal Entries

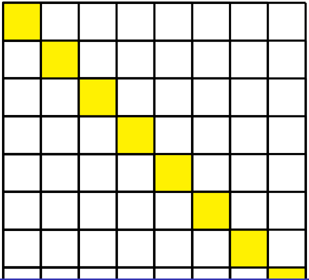
- Diagonal entries (Δ, Δ) correspond to $i \oplus j = S[i] \oplus S[j]$.
- Or $i \oplus S[i] = j \oplus S[j] \ldots$
- By listing all $x \oplus S[x]$, and looking for collisions in these values, we find all iterative differences.

Diagonal Entries

- Diagonal entries (Δ, Δ) correspond to $i \oplus j = S[i] \oplus S[j]$.
- Or $i \oplus S[i] = j \oplus S[j] \ldots$
- By listing all $x \oplus S[x]$, and looking for collisions in these values, we find all iterative differences. In $O(2^n)$ time and memory!

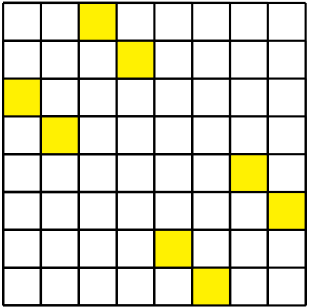
Diagonal Entries

- Diagonal entries (Δ, Δ) correspond to $i \oplus j = S[i] \oplus S[j]$.
- Or $i \oplus S[i] = j \oplus S[j] \ldots$
- By listing all $x \oplus S[x]$, and looking for collisions in these values, we find all iterative differences. In $O(2^n)$ time and memory!
- Which is exactly **the same time complexity as computing just $DDT[1][1]$!**



Shifted Diagonal Entries

- Similarly, one can compute the entries of the form $(\Delta, \Delta \oplus v)$ for a fixed v.
- Just look for values of $x \oplus S[x]$ which are shifted by v.



Summary

- Presented a new methodology to efficiently construct the diagonal of DDTs.
- Same time complexity for a diagonal as for an entry!
- Useful for 32-bit structures (Super S-box of AES, lightweight crypto).

Summary

- Presented a new methodology to efficiently construct the diagonal of DDTs.
- Same time complexity for a diagonal as for an entry!
- Useful for 32-bit structures (Super S-box of AES, lightweight crypto).
- Also works with additive differences (use $x + S[x]$).
- And allows finding the actual pairs as well.

Questions?

$E\nu\chi\alpha\rho\iota\sigma\tau\omega$!

Thank you for your attention!