Bits and Pieces

Orr Dunkelman

Computer Science Department
University of Haifa, Israel

12 May, 2013
Memoryless Collision Search

Consider the random function \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) as a directed graph:

- Let \( V = \{0,1\}^n \) (i.e., each node has a label of length \( n \)).
- and \( (x, y) \in E \) if \( f(x) = y \).

A collision in \( f(\cdot) \) can be views as two edges \((x_1, y)\) and \((x_2, y)\).
Cycle Finding

- Start from a random node $x_1$, and compute iteratively $x_{i+1} = f(x_i)$.
- After about $\sqrt{2^n}$ steps, you expect to enter a cycle.
- The entry point (unless it is back to $x_1$) suggests a cycle.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- $p_1$ is incremented each time by 1 position $p_1 \leftarrow f(p_1)$, and $p_2$ is incremented each time by 2 positions $p_2 \leftarrow f(f(p_2))$ until they collide.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- At this point, set $p_1$ to $x_1$, and increment both pointers each time by 1 position, they will collide in the entry point to the cycle.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- At this point, set $p_1$ to $x_1$, and increment both pointers each time by 1 position, they will collide in the entry point to the cycle.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1$, $p_2$ initialized both to $x_1$.
- At this point, set $p_1$ to $x_1$, and increment both pointers each time by 1 position, they will collide in the entry point to the cycle.
Floyd’s Cycle Finding Algorithm

- Start with two pointers $p_1, p_2$ initialized both to $x_1$.
- At this point, set $p_1$ to $x_1$, and increment both pointers each time by 1 position, they will collide in the entry point to the cycle.
Analysis of Floyd’s Cycle Finding Algorithm

- This method is also known as the $\rho$-method.
- Let the tail’s $(x_1 \leadsto x_3)$ length be $\ell$, and let the cycle’s length be $r$. Then if the two pointers collide after $t$ steps:

  \[ t - \ell = 2t - \ell \mod r \Rightarrow t \equiv 0 \mod r \]

- Then, after $\ell$ more steps, the pointer $p_2$ is in position $2t + \ell$, which means, it did $2t$ steps inside the cycle, which means that it points to the entry point.

- The algorithm does not work when $x_1$ is the start of the cycle, or when the cycle is of length 1 (the former is easily solved by picking a different starting point, the latter offers a fixed-point).
Differential Cryptanalysis

- Introduced by Biham and Shamir [BS90].
- Studies the development of differences through the encryption function.
- A differential characteristics $\Omega_P \rightarrow \Omega_C$ with probability $p$:

$$\Omega_P \xrightarrow{R_1} \Omega_1 \xrightarrow{R_2} \Omega_2 \xrightarrow{R_3} \Omega_C$$
Performing A Differential Attack

To attack more than \( r \) rounds with an \( r \)-round differentials:

- Pick \( T \) plaintexts which generate \( O(1/p) \) pairs of plaintexts with input difference \( \Omega_P \).
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference \( \Omega_C \).
- Analyze these pairs and find the subkeys they suggest.
Performing A Differential Attack

To attack more than $r$ rounds with an $r$-round differentials:

- Pick $T$ plaintexts which generate $O(1/p)$ pairs of plaintexts with input difference $\Omega_P$.
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference $\Omega_C$.
- Analyze these pairs and find the subkeys they suggest.

\[ P \rightarrow R_1 \rightarrow R_2, R_3, \ldots, R_{15} \rightarrow R_{16} \rightarrow C \]
Performing A Differential Attack

To attack more than $r$ rounds with an $r$-round differentials:

- Pick $T$ plaintexts which generate $O(1/p)$ pairs of plaintexts with input difference $\Omega_P$.
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference $\Omega_C$.
- Analyze these pairs and find the subkeys they suggest.

\[ P \rightarrow R_1 \rightarrow R_2, R_3, \ldots, R_{15} \rightarrow R_{16} \rightarrow C \]
Performing A Differential Attack

To attack more than $r$ rounds with an $r$-round differentials:

- Pick $T$ plaintexts which generate $O(1/p)$ pairs of plaintexts with input difference $\Omega_P$.
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference $\Omega_C$.
- Analyze these pairs and find the subkeys they suggest.
Performing A Differential Attack

To attack more than \( r \) rounds with an \( r \)-round differentials:

- Pick \( T \) plaintexts which generate \( O(1/p) \) pairs of plaintexts with input difference \( \Omega_P \).
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference \( \Omega_C \).
- Analyze these pairs and find the subkeys they suggest.
Performing A Differential Attack

To attack more than $r$ rounds with an $r$-round differentials:

- Pick $T$ plaintexts which generate $O(1/p)$ pairs of plaintexts with input difference $\Omega_P$.
- Ask for the encryption of these plaintexts.
- Identify among the ciphertexts pairs which may have difference $\Omega_C$.
- Analyze these pairs and find the subkeys they suggest.

\[
P \xrightarrow{\Omega_P?} R_2, R_3, \ldots, R_{15} \xrightarrow{\Omega_C?} C
\]
Impossible Differential Cryptanalysis

- Introduced by Biham, Biryukov and Shamir [BBS99].
- Uses differentials with probability $0$.
- Whenever a subkey suggests that a pair "satisfies" the differential, it is necessarily wrong one, and can be discarded.
Impossible Differential Cryptanalysis

- Introduced by Biham, Biryukov and Shamir [BBS99].
- Uses differentials with probability 0.
- Whenever a subkey suggests that a pair ”satisfies” the differential, it is necessarily wrong one, and can be discarded.
- The attack has to discard a large set of (sub)keys, thus it has a lower bound on the time complexity of the attack.
Generic Attack Algorithm

- Let the number of possible subkeys be $N_S$.
- Pick $T$ plaintexts which generate enough pairs of plaintexts with “input difference” $\Omega_P$ and “output difference” $\Omega_C$ to discard most of (or all) the $N_S - 1$ wrong subkeys.
- Ask for the encryption of these plaintexts.
- Identify pairs which may have “output difference” $\Omega_C$ and “input difference” $\Omega_P$.
- Analyze these pairs and discard the subkeys they suggest.

\[
P \xrightarrow{R_1} R_2, R_3, \ldots, R_{15} \xrightarrow{R_{16}} C
\]
Generic Attack Algorithm

- Let the number of possible subkeys be $N_S$.
- Pick $T$ plaintexts which generate enough pairs of plaintexts with "input difference" $\Omega_P$ and "output difference" $\Omega_C$ to discard most of (or all) the $N_S - 1$ wrong subkeys.
- Ask for the encryption of these plaintexts.
- Identify pairs which may have "output difference" $\Omega_C$ and "input difference" $\Omega_P$.
- Analyze these pairs and discard the subkeys they suggest.
Generic Attack Algorithm

- Let the number of possible subkeys be $N_S$.
- Pick $T$ plaintexts which generate enough pairs of plaintexts with “input difference” $\Omega_P$ and “output difference” $\Omega_C$ to discard most of (or all) the $N_S - 1$ wrong subkeys.
- Ask for the encryption of these plaintexts.
- Identify pairs which may have “output difference” $\Omega_C$ and “input difference” $\Omega_P$.
- Analyze these pairs and discard the subkeys they suggest.
Generic Attack Algorithm

- Let the number of possible subkeys be $N_S$.
- Pick $T$ plaintexts which generate enough pairs of plaintexts with “input difference” $\Omega_P$ and “output difference” $\Omega_C$ to discard most of (or all) the $N_S - 1$ wrong subkeys.
- Ask for the encryption of these plaintexts.
- Identify pairs which may have “output difference” $\Omega_C$ and “input difference” $\Omega_P$.
- Analyze these pairs and discard the subkeys they suggest.
Finding Impossible Differentials

- Any random permutation has many impossible differentials.
- This follows from the fact that for any non-zero input difference there are at most $2^{n-1}$ output differences.
Finding Impossible Differentials

- Any random permutation has many impossible differentials.
- This follows from the fact that for any non-zero input difference there are at most $2^{n-1}$ output differences.
- The only problem is that finding such impossible differentials requires constructing the difference distribution table of the entire cipher.
Finding Impossible Differentials

- Any random permutation has many impossible differentials.
- This follows from the fact that for any non-zero input difference there are at most $2^{n-1}$ output differences.
- The only problem is that finding such impossible differentials requires constructing the difference distribution table of the entire cipher.
- and usually they are of little cryptanalytic use.
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions $(\alpha, 0) \not\rightarrow (\alpha, 0)$. 
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions $(\alpha, 0) \not\Rightarrow (\alpha, 0)$. 
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions $(\alpha, 0) \not\rightarrow (\alpha, 0)$.
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions $(\alpha, 0) \not\Rightarrow (\alpha, 0)$. 
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\xrightarrow{\oplus} (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).
The Miss-in-the-Middle approach is based on taking two probability 1 truncated differentials that cannot exist.

Consider for example the 5-round impossible differential for any Feistel with bijective round functions \((\alpha, 0) \not\rightarrow (\alpha, 0)\).