Dissection: A New Paradigm for Solving Bicomposite Search Problems

Itai Dinur

École normale supérieure, Paris, France

Joint work with Orr Dunkelman, Nathan Keller and Adi Shamir
A block cipher is a set of $2^n$ permutations indexed by an $n$-bit key $K$.

Each permutation works on a space of $n$ bits.

- Maps $n$-bit plaintexts to $n$-bit ciphertexts.
Encryption and Decryption

- Given a key $K$ and a plaintext $P$ it is easy to encrypt, i.e. compute the ciphertext $C=E_K(P)$
- Given a key $K$ and a ciphertext $C$ it is easy to decrypt, i.e. compute the plaintext $P=D_K(C)$
Single Encryption

• The Basic Cryptanalytic Problem:
  • **Input**: a list of plaintext-ciphertext pairs \((P_1, C_1), (P_2, C_2), (P_3, C_3), \ldots\)
  • **Goal**: find all keys \(K\) such that
    \[ C_1 = E_K(P_1), \ C_2 = E_K(P_2), \ldots \]

• Exhaustive Search:
  • For each \(n\)-bit value of \(K\)
    • Perform trial encryptions i.e., test whether \(C_1 = E_K(P_1)\), if so test whether \(C_2 = E_K(P_2)\) ...
  • **Time**: \(2^n\), **Memory**: \textit{constant}
Double Encryption

- $C = E_{K_2}(E_{K_1}(P))$ with independent keys $n$-bit keys $K_1, K_2$
- Suggested following concerns about the small keys size of DES
MITM Attack (Hellman, Merkle ‘81)

- For each \( n \)-bit value of \( K_1 \)
  - Partially encrypt \( P_1 \) and store the \( n \)-bit suggestions for \( X \) in a sorted list
- For each \( n \)-bit value of \( K_2 \)
  - Partially decrypt \( C_1 \) and look for matches in the list
  - For each of the \( \approx 2^n \) matches test the full key
- Time \( 2^n \), memory \( 2^n \) (ignoring logarithmic factors)
Triple Encryption

- **Triple-DES** was used as a de-facto encryption standard from 1998 until 2001 (and even today...)
- A trivial extension of the **MITM** attack (by guessing $K_3$) breaks triple encryption in time $2^{2n}$ and memory $2^n$
  - Still the best known algorithm for triple encryption
Multiple Encryption

- $r$-fold encryption: $E_{K_r}(E_{K_{r-1}}(...(E_{K_1}(P))))$ with independent keys $K_1, K_2, ..., K_r$
- An extension of MITM breaks $r$-fold encryption in time $T$ and memory $M$ such that $TM=2^{rn}=N$ (provided $M \leq 2^{[r/2]n}$)
- Suggests an optimal time-memory tradeoff of $TM=N$
Improved Attack on 4-Fold Encryption with $M=2^n$

For each $n$-bit value of $X_2$
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$
  - Given $P_1, X_2$ obtain $\approx 2^n$ suggestions for $K_1, K_2$ using a 2R MITM attack
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$
  - Given $P_1,X_2$ obtain $\approx 2^n$ suggestions for $K_1,K_2$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and store the triplet in a sorted list
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$:
  - Given $P_1, X_2$ obtain $\approx 2^n$ suggestions for $K_1, K_2$ using a 2R MITM attack.
  - For each suggestion, obtain $Y_2$ and store the triplet in a sorted list.
- Given $X_2, C_1$ obtain $\approx 2^n$ suggestions for $K_3, K_4$ using a 2R MITM attack.
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$
  - Given $P_1, X_2$ obtain $\approx 2^n$ suggestions for $K_1, K_2$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and store the triplet in a sorted list
  - Given $X_2, C_1$ obtain $\approx 2^n$ suggestions for $K_3, K_4$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and match with the stored list

<table>
<thead>
<tr>
<th>$K_1, K_2$</th>
<th>$Y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>110 101</td>
<td>000</td>
</tr>
<tr>
<td>111 011</td>
<td>010</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>100 110</td>
<td>111</td>
</tr>
</tbody>
</table>
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$
  - Given $P_1, X_2$ obtain $\approx 2^n$ suggestions for $K_1, K_2$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and store the triplet in a sorted list
  - Given $X_2, C_1$ obtain $\approx 2^n$ suggestions for $K_3, K_4$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and match with the stored list

For each of the $\approx 2^n$ matches **test the full key** using $(P_3, C_3)$ and $(P_4, C_4)$
Improved Attack on 4-Fold Encryption with $M=2^n$

- For each $n$-bit value of $X_2$
  - Given $P_1, X_2$ obtain $\approx 2^n$ suggestions for $K_1, K_2$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and store the triplet in a sorted list
  - Given $X_2, C_1$ obtain $\approx 2^n$ suggestions for $K_3, K_4$ using a 2R MITM attack
  - For each suggestion, obtain $Y_2$ and match with the stored list
  - For each of the $\approx 2^n$ matches test the full key using $(P_3, C_3)$ and $(P_4, C_4)$
- Time $2^{2n}$, memory $2^n$ (the same as triple-encryption!)
Increasing r Further

- We obtained $TM = 2^{3n}$ (instead of $2^{4n}$) for $r = 4$
- What happens when we increase $r$ further?
- We first fix $M = 2^n$ and try to minimize $T$
Surprisingly Efficient Attack on 7-Fold Encryption (a 7r attack)

- Split the 7r cipher into two subciphers, a 3r top part and a 4r bottom part

- Guess 2 intermediate encryption values in the middle (one for \((P_1, C_1)\) and one for \((P_2, C_2)\))
  - Apply a 3r attack to the top part and store the \(2^n\) returned suggestions
  - Apply the 4r attack to the bottom part and test the returned keys on the fly
Analysis of the Attack

• We guess $2n$ bits in the middle
  • The top $3r$ attack takes $2^{2n}$ time and $2^n$ memory
  • The bottom $4r$ attack takes $2^{2n}$ time and $2^n$ memory
• The total complexity is $T=2^{4n}$ (instead of $2^{6n}$)
• We obtain $TM=2^{5n}$ (instead of $2^{7n}$)
### Extending the 7r Attack

- Our 7r attack divides the cipher **asymmetrically** into a top and bottom part.

| r | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>$2^n$</td>
<td>$2^n$</td>
<td>$2^{2n}$</td>
<td>$2^{3n}$</td>
<td>$2^{4n}$</td>
<td>$2^{5n}$</td>
<td>$2^{6n}$</td>
<td>$2^{7n}$</td>
<td>$2^{8n}$</td>
</tr>
</tbody>
</table>

- Can be extended recursively by dividing the cipher **asymmetrically** into subciphers.
- The algorithms generalize to any amount of memory.
Dissection Algorithms

• We obtain a new class of algorithms which we call **dissection** algorithms

• We perform “cuts” of different sizes in carefully chosen places of the encryption structure
Composite Problems

• A composite problem
  • We are given the initial value(s) and the final value(s) of a cascade of $r$ steps
  • In each step, one of a list of possible transformations was applied
  • The goal: Find out, which transformation was applied in each step (i.e., find all possible options)

• Clearly, $r$-fold encryption is a composite problem
Application to Knapsacks (Subset-Sum)

- **Modular Knapsack Problem**: 
  - **Input**: A list of \( n \) integers \( \{a_1, a_2, \ldots, a_n\} \) of \( n \) bits each, and a target integer \( S \)
  - **Goal**: Find a vector \( \epsilon = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\} \) where \( \epsilon_i \in \{0, 1\} \) such that \( S = \sum_{1 \leq i \leq n} (\epsilon_i \cdot a_i) \mod 2^n \)

- How do we apply the dissection techniques to the Knapsack problem?
Representing Knapsack as a Block Cipher

\[ P + (\varepsilon_1 \cdot a_1) + (\varepsilon_2 \cdot a_2) + \ldots + (\varepsilon_n \cdot a_n) \pmod{2^n} \]

\[ C = P + \sum_{1 \leq i \leq n} (\varepsilon_i \cdot a_i) \pmod{2^n} \]

- We fix the plaintext to be the \(0\ n\)-bit vector, the ciphertext to be \(S\)
- The knapsack problem reduces to recovering the key of this block cipher, given one plaintext-ciphertext pair
Representing Knapsack as 4-Fold Encryption

- We split the knapsack to 4 independent knapsacks by splitting the generators and defining $S = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4 \pmod{2^n}$
- $X_i = \sum_{1 \leq j \leq i} (\sigma_j)$
Representing Knapsack as 4-Fold Encryption

- **Problem:** In \( r \)-fold encryption, we have \( r \) “small” plaintexts \( \rightarrow \) can efficiently guess intermediate values. Here we have a single “big” plaintext.

- **Solution:** Split the “block cipher” also vertically into \( n/4 \)-bit blocks.

\[
\begin{align*}
\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n/4}\} \\
\{\varepsilon_{n/4+1}, \ldots, \varepsilon_{n/2}\} \\
\{\varepsilon_{n/2+1}, \ldots, \varepsilon_{3n/4}\} \\
\{\varepsilon_{3n/4+1}, \ldots, \varepsilon_n\}
\end{align*}
\]
Representing Knapsack as 4-Fold Encryption

- **Problem**: Dependency between the “vertical” chunks through addition carries

- **Solution**: Guess the intermediate encryption values in their natural order (from right to left)
Representing Knapsack as 4-Fold Encryption

• **Conclusion:** We can apply to knapsacks the algorithm for $r$-fold encryption, for any $r$

• We choose $r$ according to the amount of **available memory**, in order to optimize the running time of the dissection algorithms
Time-Memory Tradeoff for Knapsacks

Becker, Coron and Joux 2011

Schroeppel and Shamir 1981
Examples of Other Composite Problems

• **Rubik’s cube** – find a shortest solution given an initial state

• The matching phase in **rebound attacks** on hash functions

• etc...
Conclusions

• We improved the best known algorithms for multiple encryption
• Our techniques allow us to improve the best known algorithms for the knapsack problem with small memory
• These techniques are applicable to other composite problems that have nothing to do with cryptography
Open Problems

• Are our results optimal?
  • Can you improve our $7r$ attack?

• Our algorithms use the smallest number of P/C pairs. Can you improve the attacks by using slightly more data?

• Find additional applications to dissection algorithms
Thanks for your attention!