New Generic Attacks which are Faster than Exhaustive Search

Christophe De Cannière, Itai Dinur, and Adi Shamir

Katholieke Universiteit Leuven
Weizmann Institute of Science
Ecole Normale Supérieure

February 24, 2009

Question

Every n-bit block cipher $c = E(k, p)$ can be broken in 2^n operations by exhaustive search.

Question:
Can we do (slightly) better?

Let us try to use Cube Attacks

Classical Cube Attack

$$E : (k, v) \mapsto c$$

- Attacker controls public value v (chosen plaintext)
- **Goal:** recover secret key k
- Attack exploits non-randomness of E
- Will not work if $E(k, v)$ is random function of degree $2 \cdot n$

k, v, p, and c are all n-bit words

Let us try to use Cube Attacks

How about related keys?

$$E : (k \oplus v, 0) \mapsto c$$

- Attacker controls public value v (related key)
- **Goal:** recover secret key k

k, v, p, and c are all n-bit words

Observation

- **Main observation:** k_i and v_i never appear together in the same monomial of $E(k \oplus v, 0)$

$$E(k \oplus v, 0) = \cdots + v_1 v_3 v_4 v_6 (k_2 + k_8 k_5 k_7 + k_2 k_7 k_8) + \cdots$$

Observation

- **Main observation:** k_i and v_i never appear together in the same monomial of $E(k \oplus v, 0)$

$$E(k \oplus v, 0) = \cdots + v_1 v_3 v_4 v_6 (k_2 + k_8 k_5 k_7 + k_2 k_7 k_8) + \cdots$$

$\Rightarrow$ Summing over the cube $v_1 v_3 v_4 v_6$ eliminates k_1, k_3, k_4, and k_6 from the corresponding superpoly.

Observation

- **Main observation:** k_i and v_i never appear together in the same monomial of $E(k \oplus v, 0)$

$$E(k \oplus v, 0) = \cdots + v_1 v_3 v_4 v_6 (k_2 + k_8 k_5 k_7 + k_2 k_7 k_8) + \cdots$$

$\Rightarrow$ Summing over the cube $v_1 v_3 v_4 v_6$ eliminates k_1, k_3, k_4, and k_6 from the corresponding superpoly.

$\Rightarrow$ Generic attack: works even when $E(k \oplus v, 0)$ has degree n (highest possible degree)

Attacking $E(k \oplus v, p)$

Precomputation

- Fix plaintext p (e.g., $p = 0$)
- Sum over cube $v_{m+1} v_{m+2} \cdots v_n$ and obtain superpoly $g_i(k_1, k_2 \cdots k_m)$ for each of the n ciphertext bits c_i

$$v_1 v_2 \cdots v_m v_{m+1} \cdots v_{n-2} v_{n-1} v_n$$
$$k_1 k_2 \cdots k_m k_{m+1} \cdots k_{n-2} k_{n-1} k_n$$

- **Cost:** 2^n function evaluations

Attacking $E(k \oplus v, p)$

Precomputation

- System of n nonlinear equations in m variables

$$
\begin{aligned}
g_1 &= \quad k_2 + k_4 \quad \cdots + k_1 k_2 \cdots k_m \\
g_2 &= k_1 + k_5 k_7 \quad \cdots + k_1 k_2 \cdots k_m \\
&\vdots \\
g_n &= \quad k_3 + k_5 k_7 \quad \cdots
\end{aligned}
$$

$$\underbrace{\hspace{8em}}_{2^m \text{ monomials}}$$

Attacking $E(k \oplus v, p)$

Precomputation

- m linear expressions in n variables

$$\begin{array}{rcccccccc}
k_1 & = & g_1 & + g_2 & & + g_7 & & \cdots & \\
k_2 & = & & g_2 & & & + g_8 & \cdots & + g_n \\
& \vdots & & & & & & & \\
k_m & = & & & g_5 & & + g_8 & \cdots & \\
& & \multicolumn{7}{c}{\longleftarrow\;\; n \text{ variables} \;\;\longrightarrow}
\end{array}$$

- If $m = \log n \Rightarrow$ linearize and solve
- Store these $\log n$ vectors of n bits in memory

Attacking $E(k \oplus v, p)$

Online Phase

- Compute $g_1 \cdots g_n$ by summing the ciphertexts over the cube $v_{m+1} v_{m+2} \cdots v_n$
- Recover $k_1, k_2 \cdots k_m$ using precomputed vectors
- Exhaustively search for remaining $n - m$ key bits
- **Cost:** $2^{n-m} + 2^{n-m} = \frac{2}{n} 2^n$ function evaluations

Attacking $E(k \oplus v, p)$

Online Phase

- Compute $g_1 \cdots g_n$ by summing the ciphertexts over the cube $v_{m+1} v_{m+2} \cdots v_n$
- Recover $k_1, k_2 \cdots k_m$ using precomputed vectors
- Exhaustively search for remaining $n - m$ key bits
- **Cost:** $2^{n-m} + 2^{n-m} = \frac{2}{n} 2^n$ function evaluations

$\Rightarrow$ **What did we achieve:** reduction of time by a factor $n/2$ in exchange for $\log n$ words of memory (each n bits long)

Standard TMTO Attack

Precomputation

- Compute $E(x, 0)$ for each

$$x = x_1 \cdots x_m 0000000 \cdots 00$$

- Store these $2^m = n$ words in memory

Online Phase

- Compute $E(k \oplus v, 0)$ for each

$$v = 00 \cdots 0 v_{m+1} v_{m+2} \cdots v_n$$

- Check for match in memory
- If match: $k = v \oplus x$

Standard TMTO Attack

- **Cost:** $2^{n-m} = \frac{1}{n}2^n$ function evaluations

$\Rightarrow$ **What did we achieve:** reduction of time by a factor n in exchange for n words of memory (of n bits each)

Tweaked TMTO Attack

Precomputation

- Compute $E(x, 0)$ for each

$$x = x_1 \cdots x_m 0000000 \cdots 00$$

- Store the m first bits of these $2^m = n$ words in memory

Online Phase

- Compute $E(k \oplus v, 0)$ for each

$$v = 00 \cdots 0 v_{m+1} v_{m+2} \cdots v_n$$

- Check for match in memory (in m first bits)
- If match (very likely): recompute $E(x, p)$
- If match remains: $k = v \oplus x$

Tweaked TMTO Attack

- **Cost:** $2 \cdot 2^{n-m} = \frac{2}{n} 2^n$ function evaluations

$\Rightarrow$ **What did we achieve:** reduction of time by a factor $n/2$ in exchange for n words of memory (of $\log n$ bits each)

Summary

	cube attack	plain TO	tweaked TO
Function evaluations	$\frac{2}{n}2^n$	$\frac{1}{n}2^n$	$\frac{2}{n}2^n$
Memory (n-bit words)	$\log n$	n	$\log n$
Precomputation	2^n	n	n
Memory accesses	1	$\frac{1}{n}2^n$	$\frac{1}{n}2^n$