Quantum Differential and Linear Cryptanalysis

Marc Kaplan\textsuperscript{1,2}  Gaëtan Leurent\textsuperscript{3}
Anthony Leverrier\textsuperscript{3}  María Naya-Plasencia\textsuperscript{3}

\textsuperscript{1}LTCI, Télécom ParisTech

\textsuperscript{2}School of Informatics, University of Edinburgh

\textsuperscript{3}\text{Inria Paris}

FSE 2017
Motivation

What would be the impact of quantum computers on symmetric cryptography?

- Some physicists think they can build quantum computers
- NSA thinks we need quantum-resistant crypto (or do they?)
Motivation

What would be the impact of quantum computers on symmetric cryptography?

- Some physicists think they can build quantum computers
- NSA thinks we need quantum-resistant crypto (or do they?)
Expected impact of quantum computers

- Some problems can be solved much faster with quantum computers
  - Up to exponential gains
  - But we don’t expect to solve all NP problems

Impact on public-key cryptography

- RSA, DH, ECC broken by Shor’s algorithm
  - Breaks factoring and discrete log in polynomial time
  - Large effort to develop quantum-resistant algorithms (e.g. NIST)

Impact on symmetric cryptography

- Exhaustive search of a $k$-bit key in time $2^{k/2}$ with Grover’s algorithm
  - Common recommendation: double the key length (AES-256)
- Encryption modes are secure
- Authentication modes broken w/ superposition queries [Crypto ’16]

Kaplan, Leurent, Leverrier & Naya-Plasencia    Quantum Differential and Linear Cryptanalysis
Expected impact of quantum computers

- Some problems can be solved much faster with quantum computers
  - Up to exponential gains
  - But we don’t expect to solve all NP problems

Impact on public-key cryptography

- RSA, DH, ECC broken by Shor’s algorithm
  - Breaks factoring and discrete log in polynomial time
  - Large effort to develop quantum-resistant algorithms (e.g. NIST)

Impact on symmetric cryptography

- Exhaustive search of a $k$-bit key in time $2^{k/2}$ with Grover’s algorithm
  - Common recommendation: double the key length (AES-256)

- Encryption modes are secure
  - [Unruh & al, PQC’16]

- Authentication modes broken w/ superposition queries [Crypto ’16]
Expected impact of quantum computers

- Some problems can be solved much faster with quantum computers
  - Up to exponential gains
  - But we don’t expect to solve all NP problems

Impact on public-key cryptography

- RSA, DH, ECC broken by Shor’s algorithm
  - Breaks factoring and discrete log in polynomial time
  - Large effort to develop quantum-resistant algorithms (e.g. NIST)

Impact on symmetric cryptography

- Exhaustive search of a $k$-bit key in time $2^{k/2}$ with Grover’s algorithm
  - Common recommendation: double the key length (AES-256)
- Encryption modes are secure \[ \text{[Unruh & al, PQC’16]} \]
- Authentication modes broken w/ superposition queries \[ \text{[Crypto ’16]} \]
Expected impact of quantum computers

- Some problems can be solved much faster with quantum computers
  - Up to exponential gains
  - But we don’t expect to solve all NP problems

Impact on public-key cryptography

- RSA, DH, ECC broken by Shor’s algorithm
  - Breaks factoring and discrete log in polynomial time
  - Large effort to develop quantum-resistant algorithms (e.g. NIST)

Impact on symmetric cryptography

- Exhaustive search of a $k$-bit key in time $2^{k/2}$ with Grover’s algorithm
  - Common recommendation: double the key length (AES-256)
- Encryption modes are secure [Unruh & al, PQC’16]
- Authentication modes broken w/ superposition queries [Crypto ’16]
Overview of the talk

Main question

Is AES secure in a quantum setting?

- Symmetric design are evaluated with cryptanalysis:
  - Differential (truncated, impossible, ...)
  - Linear
  - Integral
  - Algebraic
  - ...

- We should study quantum cryptanalysis!

- Start with classical techniques
  - Do we get a quadratic speedup?
  - Do we need a quantum encryption oracle?
  - How are different cryptanalysis techniques affected?
Security notions: Classical

- **PRF security**: given access to $P/P^{-1}$, distinguishing $E$ from random
- **Classical setting**: classical computations
- **Classical security**: classical queries
- Cipher broken by adversary with
  - data $\ll 2^n$
  - time $\ll 2^k$
  - success $> 3/4$
Security notions: Quantum Q1

- **PRF security**: given access to $P/P^{-1}$, distinguishing $E$ from random
- **Quantum setting**: quantum computations
- **Classical security**: classical queries
- **Cipher broken by adversary with**
  - data $\ll 2^n$
  - time $\ll 2^{k/2}$
  - success $> 3/4$
Security notions: Quantum Q2

- **PRF security**: given access to $P/P^{-1}$, distinguishing $E$ from random

- **Quantum setting**: quantum computations

- **Quantum security**: quantum (superposition) queries

- Cipher broken by adversary with
  - data $\ll 2^n$
  - time $\ll 2^{k/2}$
  - success $> 3/4$

\[
\sum_x \psi_x |x\rangle |0\rangle \quad \text{cipher / random} \quad \sum_x \psi_x |x\rangle |P(x)\rangle
\]
About the models

**Q1 model: classical queries**

- Build a quantum circuit from classical values
- Example: breaking RSA with Shor’s algorithm

**Q2 model: superposition queries**

- Access quantum circuit implementing the primitive with a secret key
- Example: breaking CBC-MAC with Simon’s algorithm

- The Q2 model is *very strong* for the adversary
  - Simple and clean generalisation of classical oracle
  - Aim for security in the strongest (non-trivial) model
  - A Q2-secure block cipher is useful for security proofs of modes
Outline

Introduction
  Quantum Computing

Brute-force
  Grover’s algorithm

Differential
  Distinguisher
  Last-round attack

Truncated differential
  Distinguisher
  Last-round attack

Conclusion
## Grover’s algorithm

- Search for a marked element in a set $X$
- Set of marked elements $M$, with $|M| \geq \varepsilon \cdot |X|$

<table>
<thead>
<tr>
<th>Classical algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: loop</td>
</tr>
</tbody>
</table>
| 2: $x \leftarrow \text{Setup}()$ | ▶ Pick a random element in $X$, cost $S$
| 3: if $\text{Check}(x)$ then | ▶ Check if it is marked, cost $C$
| 4: return $x$ |

- $1/\varepsilon$ repetitions expected
- Complexity $(S + C)/\varepsilon$
Grover’s algorithm

- Search for a marked element in a set $X$
- Set of marked elements $M$, with $|M| \geq \varepsilon \cdot |X|$

**Grover Algorithm (as a quantum walk)**

Quantum algorithm to find a marked element using:

- **Setup**: builds a uniform superposition of inputs in $X$
- **Check**: applies a control-phase gate to the marked elements

- Only $1/\sqrt{\varepsilon}$ repetitions needed
- Complexity $(S + C)/\sqrt{\varepsilon}$

- Can produce a uniform superposition of $M$
- Can provide an oracle without measuring (nesting)
- Variant to measure $\varepsilon$ (quantum counting)
**Grover’s algorithm**

- **Search for a marked element** in a set $X$
- **Set of marked elements** $M$, with $|M| \geq \varepsilon \cdot |X|$ 

**Grover Algorithm (as a quantum walk)**

Quantum algorithm to find a marked element using:

- **Setup**: builds a uniform superposition of inputs in $X$
- **Check**: applies a control-phase gate to the marked elements

- Only $1/\sqrt{\varepsilon}$ repetitions needed
- Complexity $(S + C)/\sqrt{\varepsilon}$

- Can produce a uniform superposition of $M$
- Can provide an oracle without measuring (nesting)
- Variant to measure $\varepsilon$ (quantum counting)
Brute-force attack

- We can use Grover’s algorithm for a quantum brute-force key search

1. Capture a few known plaintext/ciphertext: $C_i = E_{\kappa^*}(P_i)$
2. **Setup**: builds a uniform superposition of $\{0, 1\}^k$
3. **Check**($\kappa$): test whether $C_i = E_{\kappa}(P_i)$

- Complexity $O(2^{k/2})$
  - Quadratic gain
- Uses the **Q1** model
  - Classical data ($C_i, P_i$)
  - Quantum circuit independent of the secret key $\kappa^*$

$S = 1, \varepsilon = 2^{-k}, C = 1$
# Outline

## Introduction
- Quantum Computing

## Brute-force
- Grover’s algorithm

## Differential
- Distinguisher
- Last-round attack

## Truncated differential
- Distinguisher
- Last-round attack

## Conclusion
Differential distinguisher: classical

- Assume a differential $\delta_{\text{in}}, \delta_{\text{out}}$ given, with

$$h := -\log Pr_x[E(x \oplus \delta_{\text{in}}) = E(x) \oplus \delta_{\text{out}}] \ll n,$$

**Classical algorithm: search for right pairs**

1. **for** $0 \leq i < 2^h$ **do**
2. $x \leftarrow \text{Rand}()$
3. **if** $E(x \oplus \delta_{\text{in}}) = E(x) \oplus \delta_{\text{out}}$ **then**
4. **return** cipher
5. **return** random

- Complexity $O(2^h)$
Differential distinguisher: quantum

- Assume a differential $\delta_{\text{in}}, \delta_{\text{out}}$ given, with

$$h := -\log\Pr_x[E(x \oplus \delta_{\text{in}}) = E(x) \oplus \delta_{\text{out}}] \ll n,$$

Quantum algorithm: Grover search for right pair

1. **Setup**: builds a uniform superposition of $\{0, 1\}^n$, $S = 1$

2. **Check($x$)**: test whether $E(x \oplus \delta_{\text{in}}) = E(x) \oplus \delta_{\text{out}}$, $\varepsilon = 2^{-h}, C = 1$

- Complexity $O(2^{h/2})$
  - Quadratic gain
- Uses the Q2 model
  - Superposition queries to $E$ with secret key
Last-Round attack: classical

Classical algorithm

1: for $0 \leq i < 2^h$ do
2: \( x \leftarrow \text{RAND()}) \)
3: \( \triangleright \) Filter possible output differences
4: if \( E(x) \oplus E(x \oplus \delta_{\text{in}}) \in \mathcal{D}_{\text{fin}} \) then
5: Find last key candidates for \((x, x \oplus \delta_{\text{in}})\)
6: Try all possibilities for remaining key bits

\[ p = 2^{-h} \]

\[ p = 2^{-h_{\text{out}}} \]

\[ \mathcal{D}_{\text{fin}} \]

\( \triangleright \) Finding partial key candidates costs \( C_{k_{\text{out}}} \)

\( \triangleright \) Between 1 and \( 2^{k_{\text{out}}} \)

\[ T = 2^h + 2^{h-n+\Delta_{\text{fin}}} \cdot \left( C_{k_{\text{out}}} + 2^{k-h_{\text{out}}} \right) \]
Last-Round attack: quantum Q2

Quantum algorithm: Grover search for right pair

1. **Setup**: builds a uniform superposition of 
   \[ X = \{ x : E(x) \oplus E(x \oplus \delta_{\text{in}}) \in D_{\text{fin}} \} \]
   using nested Grover algorithm 
   \[ S = 2^{(n-\Delta_{\text{fin}})/2} \]

2. **Check**\((x)\): Find last key cand. for \((x, x \oplus \delta_{\text{in}})\)
   Run nested Grover over remaining key bits
   \[ \varepsilon = 2^{n-h-\Delta_{\text{fin}}}, C = C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2} \]

- Repeat key recovery with right pair
- Finding partial key candidates costs \(C^*_{k_{\text{out}}}\)
  - Between 1 and \(2^{k_{\text{out}}/2}\)
  - \[ T = 2^{h/2} + 2^{(h-n+\Delta_{\text{fin}})/2} \cdot (C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2}) \]
Last-Round attack: quantum Q1

- Previous attack uses superposition queries
- Alternatively, make $2^h$ classical queries
  - Interesting if $2^h < 2^{k/2}$
  - E.g. AES-256

Quantum algorithm: Grover search for right pair

1. Setup: builds superposition of classical data using quantum memory
   \[ S = 1 \]
2. Check(x): same as Q2
   \[ \varepsilon = 2^{n-h-\Delta_{\text{fin}}}, \quad C = C^*_k + 2^{(k-h_{\text{out}})/2} \]

- \[ T = 2^h + 2^{(h-n+\Delta_{\text{fin}})/2} \cdot \left( C^*_k + 2^{(k-h_{\text{out}})/2} \right) \]
Outline

Introduction
  Quantum Computing

Brute-force
  Grover’s algorithm

Differential
  Distinguisher
  Last-round attack

Truncated differential
  Distinguisher
  Last-round attack

Conclusion
Truncated differential distinguisher: classical

- Assume vector spaces $\mathcal{D}_{\text{in}}, \mathcal{D}_{\text{out}}$ given (dim. $\Delta_{\text{in}}, \Delta_{\text{out}}$), with

$$h := -\log \Pr_{x, \delta \in \mathcal{D}_{\text{in}}}[E(x \oplus \delta) \oplus E(x) \in \mathcal{D}_{\text{out}}] \ll n - \Delta_{\text{out}},$$

**Classical algorithm (using structures)**

1. for $0 \leq i < 2^{h - 2\Delta_{\text{in}}}$ do
2. $x \leftarrow \text{Rand}()$
3. $L \leftarrow \{E(x \oplus \delta) : \delta \in \mathcal{D}_{\text{in}}\}$
4. if $\exists y_1, y_2 \in L$ s.t. $y_1 \oplus y_2 \in \mathcal{D}_{\text{out}}$ then
5. return cipher
6. return random

- Complexity $O(2^{h - \Delta_{\text{in}}})$
Truncated differential distinguisher: quantum

- Assume vector spaces $\mathcal{D}_{\text{in}}, \mathcal{D}_{\text{out}}$ given (dim. $\Delta_{\text{in}}, \Delta_{\text{out}}$), with

$$h := -\log \mathbb{P}_{x,\delta \in \mathcal{D}_{\text{in}}} [E(x \oplus \delta) \oplus E(x) \in \mathcal{D}_{\text{out}}] \ll n - \Delta_{\text{out}},$$

Quantum algorithm: Grover search for structure with right pair

1. **Setup**: builds a uniform superposition of $\{0,1\}^n$

2. **Check**($x$): test whether $\exists y_1, y_2 \in x \oplus \mathcal{D}_{\text{in}}$ s.t. $y_1 \oplus y_2 \in \mathcal{D}_{\text{out}}$

$$\varepsilon = 2^{-h + 2\Delta_{\text{in}}}, \quad C = ?$$
Finding collisions

- Finding \( y_1, y_2 \in L \) s.t. \( y_1 \oplus y_2 \in D_{out} \): truncate and find collisions

Classical algorithm

1. Sort(\( L \))
2. for \( 0 < i < |L| \) do
3. if \( L[i] = L[i + 1] \) then return \( L[i] \)
4. return \( \perp \)

- Complexity \( \tilde{O}(N) \)

Quantum algorithmic: Ambainis’ element distinctness

- Quantum walk algorithm to find collisions
- Complexity \( O(N^{2/3}) \) — less than quadratic speedup!
- Uses memory \( O(N^{2/3}) \)
Finding collisions

- Finding $y_1, y_2 \in L$ s.t. $y_1 \oplus y_2 \in D_{\text{out}}$: truncate and find collisions

Classical algorithm

1: Sort($L$)
2: for $0 < i < |L|$ do
3: if $L[i] = L[i + 1]$ then return $L[i]$
4: return $\bot$

- Complexity $\tilde{O}(N)$

Quantum algorithmic: Ambainis’ element distinctness

- Quantum walk algorithm to find collisions
- Complexity $O(N^{2/3})$ — less than quadratic speedup!
- Uses memory $O(N^{2/3})$
Truncated differential distinguisher: quantum

- Assume vector spaces $D_{\text{in}}, D_{\text{out}}$ given ($\text{dim. } \Delta_{\text{in}}, \Delta_{\text{out}}$), with

$$h := -\log \Pr_{x, \delta \in D_{\text{in}}} [E(x \oplus \delta) \oplus E(x) \in D_{\text{out}}] \ll n - \Delta_{\text{out}},$$

Quantum algorithm: Grover search for structure with right pair

1. **Setup**: builds a uniform superposition of $\{0, 1\}^n$  
2. **Check**$(x)$: test whether $\exists y_1, y_2 \in x \oplus D_{\text{in}}$ s.t. $y_1 \oplus y_2 \in D_{\text{out}}$

$$\varepsilon = 2^{-h+2\Delta_{\text{in}}}, C = 2^{2\Delta_{\text{in}}/3}$$

- Complexity $O(2^{h/2-\Delta_{\text{in}}/3})$ — less than quadratic speedup
- Uses the Q2 model
  - Superposition queries to $E$ with secret key
Last-Round attack: classical

Classical algorithm

1: for $0 \leq i < 2^{h-2\Delta_{in}}$ do
2: \hspace{1em} $x \leftarrow \text{RAND}()$
3: \hspace{1em} $L \leftarrow \{E(x \oplus \delta) : \delta \in D_{in}\}$
4: \hspace{1em} ▷ Filter possible output differences
5: \hspace{1em} if $\exists y_1, y_2 \in L$ s.t. $y_1 \oplus y_2 \in D_{out}$ then
6: \hspace{1em} \hspace{1em} Find last key candidates for $(y_1, y_2)$
7: \hspace{1em} \hspace{1em} Try all possibilities for remaining key bits

Finding partial key candidates costs $C_{k_{out}}$

▷ Between 1 and $2^{k_{out}}$

$T = 2^{h-\Delta_{in}} + 2^{h-n+\Delta_{fin}} \cdot (C_{k_{out}} + 2^{k-h_{out}})$
Last-Round attack: quantum $Q2$

Assume each structure has pairs with difference in $D_{\text{fin}}$

Q2 algo: Grover search for structure with right pair

1. **Setup**: unif. superposition
   \[ S = 1, \varepsilon = 2^{2\Delta_{\text{in}} - h} \]
2. **Check($x$)**: Grover search over pairs in $x \oplus D_{\text{in}}$
   1. **Setup**: Ambainis to find pairs with output in $D_{\text{fin}}$
   \[ S' = 2^{(n - \Delta_{\text{fin}})/3} \]
   2. **Check($x_1, x_2$)**: Find last key candidates
      Run nested Grover over remaining key bits,
      \[ \varepsilon' = 2^{-2\Delta_{\text{in}} + (n - \Delta_{\text{fin}})}, C' = C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2} \]
   \[ C = 2^{\Delta_{\text{in}} - (n - \Delta_{\text{fin}})/6} + 2^{\Delta_{\text{in}} + (\Delta_{\text{fin}} - n)/2} \left( C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2} \right) \]

\[ T = 2^{h/2 - (n - \Delta_{\text{fin}})/6} + 2^{(h - n + \Delta_{\text{fin}})/2} \left( C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2} \right) \]
**Last-Round attack: quantum Q1**

- Alternatively, use classical queries
- Filter pairs with output in $D_{\text{fin}}$ classically

**Q1 algo: Grover search for structure with right pair**

1. **Setup:** builds superposition of classical data using quantum memory $S = 1$
2. **Check**($x_1, x_2$): Find last key candidates
Run nested Grover over remaining key bits

$$\varepsilon = 2^{n-h-\Delta_{\text{fin}}}, C = C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2}$$

$$T = 2^{h-\Delta_{\text{in}}} + 2^{(h-n+\Delta_{\text{fin}})/2} \cdot \left(C^*_{k_{\text{out}}} + 2^{(k-h_{\text{out}})/2}\right)$$
Summary: simplified complexities

- **Simple differential distinguisher**
  \[ D_C = 2^h \quad D_{Q1} = 2^h = D_C \quad D_{Q2} = 2^{h/2} = \sqrt{D_C} \]
  \[ T_C = 2^h \quad T_{Q1} = 2^h = T_C \quad T_{Q2} = 2^{h/2} = \sqrt{T_C} \]

- **Simple differential LR attack**
  \[ D_C = 2^h \quad D_{Q1} = 2^h = D_C \quad D_{Q2} = 2^{h/2} = \sqrt{D_C} \]
  \[ T_C = 2^h + C_k \quad T_{Q1} = 2^h + C^*_k \quad T_{Q2} = 2^{h/2} + C^*_k \approx \sqrt{T_C} \]

- **Truncated differential distinguisher**
  \[ D_C = 2^{h-\Delta_{in}} \quad D_{Q1} = 2^{h-\Delta_{in}} = D_C \quad D_{Q2} = 2^{h/2-\Delta_{in}/3} > \sqrt{D_C} \]
  \[ T_C = 2^{h-\Delta_{in}} \quad T_{Q1} = 2^{h-\Delta_{in}} = T_C \quad T_{Q2} = 2^{h/2-\Delta_{in}/3} > \sqrt{T_C} \]

- **Truncated differential LR attack Assuming > 1 filtered pairs / structure**
  \[ D_C = 2^{h-\Delta_{in}} \quad D_{Q1} = 2^{h-\Delta_{in}} = D_C \quad D_{Q2} = 2^{h/2-(n-\Delta_{fin})/6} > \sqrt{D_C} \]
  \[ T_C = 2^{h-\Delta_{in}} + C_k \quad T_{Q1} = 2^{h-\Delta_{in}} + C^*_k \quad T_{Q2} = 2^{h/2-(n-\Delta_{fin})/6} + C^*_k \approx \sqrt{T_C} \]
Concrete examples

- Truncated differential attacks have less than quadratic speedup
- Can become worse than Grover key search (not an attack)
- The best quantum attack is not always a quantum version of the best classical attack

### LAC (reduced LBlock, $n = 64$)

- Differential with probability $2^{-61.5}$
  - Classical distinguisher with complexity $2^{62.5}$
  - Quantum distinguisher with complexity $2^{31.75}$
- Truncated differential with $\Delta_{\text{in}} = 12, \Delta_{\text{out}} = 20, 2^h = 2^{-44} + 2^{-55.3}$
  - Classical distinguisher with complexity $2^{60.9}$
  - Quantum distinguisher with complexity $2^{33.4}$
Concrete examples

- Truncated differential attacks have less than quadratic speedup
- Can become worse than Grover key search (not an attack)
- The best quantum attack is not always a quantum version of the best classical attack

**KLEIN-64 ($n = 64$)**

- Truncated differential with $h = 69.5$, $\Delta_{in} = 16$, $\Delta_{fin} = 32$, $k = 64$, $k_{out} = 32$, $h_{out} = 45$
  - Classical attack with complexity $2^{58.2}$
  - Quantum attack with complexity $> 2^{32}$
Concrete examples

- Truncated differential attacks have less than quadratic speedup
- Can become worse than Grover key search (not an attack)
- The best quantum attack is not always a quantum version of the best classical attack

**KLEIN-96 (n = 64)**

- Truncated differential with $h = 78$, $\Delta_{\text{in}} = 32$, $\Delta_{\text{fin}} = 32$, $k = 96$, $k_{\text{out}} = 48$, $h_{\text{out}} = 52$
  - Classical attack with complexity $2^{90}$
  - Q2 attack with complexity $2^{47.3}$
  - Q1 attack with complexity $2^{47.96}$
Conclusions

- We fixed some mistakes from the ToSC version
  - Updated version on arXiv:1510.05836

- Quantification of classical attacks using Grover and Ambainis
  - Differential, truncated differential and linear cryptanalysis

- “It’s complicated”
- Up to quadratic speedup
  - If key search is the best classical attack,
    Grover key search is the best quantum attack

- Data complexity can only be reduced using quantum queries
- Cipher with $k > n$ are most likely to see quadratic speedup
  - Attacks with classical queries (Q1 model) possible
Bonus slide: Linear cryptanalysis

- **Linear distinguisher**

\[
D_C = \frac{1}{\varepsilon^2} \quad D_{Q1} = \frac{1}{\varepsilon^2} = D_C \quad D_{Q2} = \frac{1}{\varepsilon} = \sqrt{D_C} \\
T_C = \frac{1}{\varepsilon^2} \quad T_{Q1} = \frac{1}{\varepsilon^2} = T_C \quad T_{Q2} = \frac{1}{\varepsilon} = \sqrt{T_C}
\]

- **Linear attack with \(\ell\) \(r\)-round distinguishers (Matsui 1)**

\[
D_C = \frac{1}{\varepsilon^2} \quad D_{Q1} = \frac{\ell}{\varepsilon^2} > D_C \quad D_{Q2} = \frac{\ell}{\varepsilon} > \sqrt{D_C} \\
T_C = \frac{\ell}{\varepsilon^2} + 2^{k-\ell} \quad T_{Q1} = \frac{\ell}{\varepsilon^2} + 2^{(k-\ell)/2} \quad T_{Q2} = \frac{\ell}{\varepsilon} + 2^{(k-\ell)/2} > \sqrt{T_C}
\]

- **Last-round linear attack (Matsui 2)**

\[
D_C = \frac{1}{\varepsilon^2} \quad D_{Q1} = \frac{1}{\varepsilon^2} = D_C \quad D_{Q2} = 2^{k_{out}/2}/\varepsilon > \sqrt{D_C} \\
T_C = C_k \quad T_{Q1} = \frac{1}{\varepsilon^2} + \sqrt{C_k} \quad T_{Q2} = \sqrt{C_k} = \sqrt{T_C}
\]