# STUDI DAN ANALISIS PENGGUNAAN *KEY-SCHEDULE* PADA ALGORITMA IDEA (INTERNATIONAL DATA ENCRYPTION ALGORITHM)

**Nitia Rahmi – 13504068**
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institute Teknologi Bandung
Jl. Ganesa 10, Bandung
**Email** : if14068@students.if.itb.ac.id

***Abstract*** *Key-schedule merupakan algoritma untuk memperluas kunci master yang relative pendek (sekitar 40-256 bit) menjadi kunci yang panjangnya ratusan hingga ribuan bit. [3] Key-schedule digunakan dalam beberapa algoritma cipher blok, salah satunya IDEA (International Data Encryption Algorithm) . Pada algoritma kriptografi, beberapa serangan akibat penggunaan key-schedule mungkin terjadi baik berupa meet in the middle attack, linier factor, weak keys, detectable key classes, dan related key attack. Dalam makalah ini studi dan analisis akan lebih ditekankan pada bentuk penyerangan related key karena related key attack dianggap paling berpotensi sebagai dampak penggunaan key-schedule. Sebagai studi kasus, akan analisis terhadap algoritma IDEA. Pada akhir makalah, akan diajukan prinsip desain key-schedule agar lebih kuat terhadap serangan yang mungkin.*

***Kata Kunci :*** *IDEA* (*International Data Encryption Algorithm*), *serangan, key-schedule, related key*

## 1. PENDAHULUAN

Pada algoritma cipher blok dengan jumlah putaran yang banyak, dibutuhkan kunci-kunci yang banyak pula. Padahal kunci yang dimasukkan pengguna panjangnya reatif pendek, yaitu sekitar 40-256 bit. Untuk itu, pada beberapa algoritma cipher blok diterapkan pembangkit kunci internal (*key-schedule*). *Key-schedule* merupakan algoritma untuk memperluas kunci master yang relative pendek (sekitar 40-256 bit) menjadi kunci yang panjangnya ratusan hingga ribuan bit. Pada dasarnyta algoritma pembangkit kunci internal (*key-schedule*) dibuat serumit mungkin tetapi tetap dapat diproses agar teks yang telah dienkripsi dapat didekripsi. Pada algoritma kriptografi yang menerapkan algoritma *key-schedule* yang sederhana, maka akan mudah bagi kriptanalis untuk memecahkan cipherteks dengan berbagai bentuk serangan. Algoritma IDEA termasuk algoritma cipher blok dengan *key-schedule* yang cukup sederhana.

Algoritma IDEA ditetapkan sebagai pengganti DEA, yang dalam beberapa hal sudah terbukti tidak aman lagi, terutama dalam pengiriman data melalui internet. Algoritma IDEA diciptakan do ETH, Institut Teknologi Federal Swiss pada tahun 1992 oleh Xue Jia Lai dan James Massey. Pada tahun 1999, algoritma ini digunakan secara luas di Eropa untuk pengamanan sistem email PGP (*Pretty Good Privacy*). Algoritma IDEA sudah dipatenkan dan dipegang oleh Ascom, Swiss. Namun, untuk penggunaan non-komersial, algortima ini masih gratis.

Dalam kaitannya dengan penggunaan pembangkit kunci internal (*key-schedule*), algoritma IDEA menggunakan *key-schedule* untuk membangkitkan kunci internal yang panjangnya 832 -bit (52x16 bit) dari kunci eksternal yang panjangnya 128 -bit.

## 2. BENTUK SERANGAN PADA *KEY-SCHEDULE*

Ketika banyak serangan tidak dapat mematahkan suatu algoritma , para kriptanalis akan menggunakan kelemahan teoretis yang bisa dieksplotasi dari algoritma tersebut. Dengan demikian titik lemah proses enlkripsi dapat diketahui. Berikut ini akan dipaparkan beberapa jenis serangan khususnya pada penggunaan *key-schedule* pada algoritma cipher blok.

- *Meet-in-the-Middle Attack*
  *Meet-in-the-Middle Attack* terjadi ketika pada blok pertama dari cipherteks tergantung pada bit-bit kunci yang berbeda dengan blok kedua dari cipherteks sehingga penyerang dapat mematahkan / menyerang kedua blok ini secara terpisah (*independent*) kemudian bekerja mematahkan enkripsi ganda tersebut dengan cipher blok yang ada dan dua kunci yang berbeda.

- *Linear Factors*
  *Linear Factors* adalah himpunan bit-bit kunci yang komplemenya pada operasi XOR dantar kunci dengan blok cipherteks tidak mengalami perubahan (hasil sebelum XOR sama dengan setelah XOR).

- *Weak Keys*

*Weak Keys* (kunci lemah) adalah kunci yang menyebabkan tidak adanya perbedaan antara enkripsi dan dekripsi. Dekripsi terhadap cipherteks tetap menghasilkan plainteks semula, namun enkripsi dua kali berturut-turut terhadap plainteks akan kembali menghasilkan plainteks tersebut. [2] Jika jumlah kunci lemah cukup kecil, maka kunci tersebut tidak menjamin aspek confidentiality dari tujuan kriptografi, yaitu menjaga kerahasiaan pesan dab menyimpan data dengan menyembunyikan informasi lewat teknik-teknik enkripsi. Walaupun beberapa mode hash menggunakan cipher blok dimana penyerang dapat memilih kunci masukan dalam percobaannya untuk menemukan tabrakan/kekacauan, dalam mode ini cipher blok sebaiknya tidak mengandung kunci lemah maupun kunci semi lemah.

- *Detectable Key Classes*
  Salah satu cara untuk mengurangi kunci-kunci efektif yang mungkin adalah dengan membagi himpunan kunci tersebut ke dalam kelas, kemudian menemukan serangan yang mampu membuka kelas dimana kunci itu berada. Dengan kata lain, mengidentifikasi kunci dengan kelas spesifik sangat kecil, terkadang juga diacu sebagai kunci lemah. Misalnya IDEA beberapa kelas kunci yang bisa dideteksi hanya dengan enkripsi dua chosen-plaintext (plainteks tertentu yang telah dipilih). [3]

- *Attack on One-Wayness*
  Key-schedule disebut *one-way* , jika diberikan beberapa upa kunci (*subkeys*) untuk beberapa putaran, memungkinkan penyerang untuk mendapatkan berbagai informasi baru tentang *master key* atau tentang upa-kunci pada putaran yang lain.

- *Related Key Attack*
  *Related Key Attack* adalah salah satu bentuk serangan dimana penyerang memperlakukan cipherteks seperti kotak hitam (*black box*). Kriptanalis memiliki cipherteks yang dienkripsi dengan dua buah kunci yang berbeda. Kriptanalis tidak mengetahui kedua kunci tersebut, namun ia mentahui hubungan kedua kunci tersebut [3]. Winternitz dan Hellman memperlihatkan bahwa dengan enkripsi satu plainteks yang dipilih, dengan kunci dibawah $2^n$ dengan $n \leq k$, satu menyimpan nilai kunci dengan $2^{k-n}$ percobaan enkripsi, jika cipherteks menggunakan kunci k-bits. [4] Penyerangan seperti ini dapat dengan mudah diperluas menjadi bentuk penyerangan berdasarkan peluang kunci diketahui (*probabilistic known-key attack*) dengan kompleksitas yang sama dan memperlihatkan bahwa beberapa cipher yang mempunyai kekuatan lebih dari $2^{k/2}$ mampu melawan *naïve black-box attack* , jika queri *related key* tidak lebih mahal daripada queri *chosen plaintext*.

  Kriptanalisis *related-key* cukup kuat, tetapi hanya merupakan bentuk penyerangan yang sangat teoretik. Walaupun demikian, tidaklah aman untuk membiarkan jenis penyerangan ini, karena makin lama dapat dipercaya bahwa penyerangan *related-key* dapat dipraktekkan pada beberapa aplikasi nyata kriptografi.

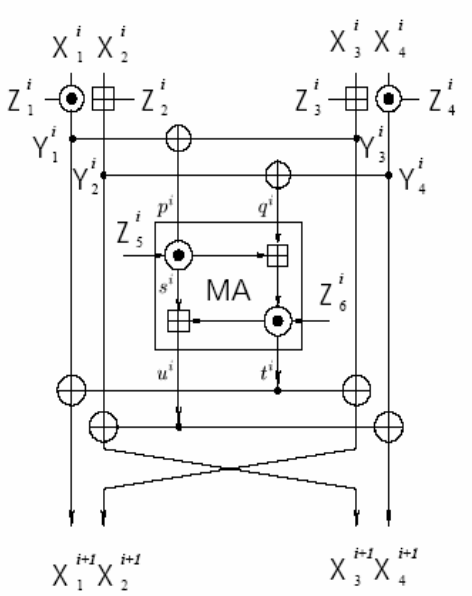
## 3. DESKRIPSI UMUM ALGORITMA IDEA

IDEA merupakan algoritma kriptografi simetri yang beroperasi dengan blok yang berukuran 64 -bit. Dalam prosesnya satu blok dibagi menjadi empat, sehingga setiap sub-blok panjangnya 16 bit. IDEA mengenkripsi blok plainteks dalam delapan putaran untuk menghasilkan blok cipherteks. Dalam proses enkripsinya, IDEA menggunakan sebuah kunci utama yang panjangnya 128 -bit, yang kemudian dengan menggunakan pembangkit kunci internal (*key - schedule*) kunci utama tersebut diperluas menjadi 52 upa-kunci yang masing-masing panjangnya 16bit.

Dalam prosesnya, untuk setiap putaran, digunakan 6 buah upa-kunci. Karena ada delapan putaran, maka ada 48 buah upa-kunci. Pada putaran kesembilan, keempat sub-blok yang ada ditransformasi dengan upa-kunci dan ada empat sub-blok sehingga membutuhkan 4 upa-kunci untuk proses transformasi terakhir ini. Dengan demikian jumlah upa-kunci adalah 52 buah.

### 3.1 Enkripsi pada IDEA

Berikut ini gambaran operasi enkripsi pada IDEA untuk satu putaran. Karena enkripsi pada IDEA terdiri dari 8 putaran, maka proses yang ditunjukkan seperti gambar di bawah ini diulang sebanyak delapan kali. Pada gambar 1 dapat dilihat bahwa pada proses enkripsi algoritma IDEA terdapat tiga operasi yang berbeda untuk pasangan sub-blok 16 –bit, yaitu XOR (prnambahan modulo 2), penambahan modulo $2^{16}$, dan perkalian dengan modulo $2^{16}+1$.




**Gambar 1 Satu Putaran Enkripsi dengan IDEA**

Proses enkripsi pada algoritma IDEA , yaitu blok plainteks dengan panjang 64-bit dibagi empat sub-blok yang masing-masing panjangnya 16-bit : $X_1$, $X_2$, $X_3$, $X_4$ sehingga X = ($X_1$, $X_2$, $X_3$, $X_4$ ). Keempat sub-blok tersebut kemudian ditransformasi melalui delapan putaran sehingga menjadi sub-blok 16-bit : $Y_1$, $Y_2$, $Y_3$, $Y_4$ sebagai cipherteks sehingga Y=( $Y_1$, $Y_2$, $Y_3$, $Y_4$) dengan Y panjangnya 64-bit, dengan Y berada di bawah kendali 52 upa-kunci yang panjang tiap upa-kunci 16-bit, dibentuk dari kunci utama yang panjangnya 128-bit.

### 3.2 Dekripsi pada IDEA

Untuk proses dekripsi hampir sama dengan enkripsi, dengan 52 upa-kunci yang merupakan hasil turunan 52 upa-kunci pada proses enkripsi. Pada proses ini, diambil invers dari operasi XOR, penambahan dengan modulo $2^{16}$, dan perkalian dengan modulo $2^{16}+1$, tergantung pada operasi yag dibuat pada fase cipher. Setiap upa-kunci adalah salah satu dari penambahan atau perkalian yang berkorespondensi dengan upa-kunci enkripsi [4]

## 4. PEMBENTUKAN UPA-KUNCI (*SUB-KEY*) PADA AGORITMA IDEA ATAU *KEY-SCHEDULE*

Sebanyak 52 upa-kunci yang masing-masing panjangnya 16-bit untuk proses enkripsi dibangkitkan dari kunci utama yang panjanganya 128-bit. Kunci utama dimasukkan oleh pengguna. Pada tiap putaran, digunakan enam buah upa-kunci sehingga dalam delapan putaran digunakan 48 buah upa-kunci. Empat buah upa-kunci sisanya digunakan untuk trasformasi akhir. Proses pembangkitan kuncinya (*key-schedule*) adalah sebagai berikut. Kunci utama yang panjangnya 128-bit dipartisi tiap 16-bit sehingga dihasilkan delapan bagian dengan panjang masing-masing bagian 16-bit. Delapan bagian ini menjadi delapan kunci pertama proses enkripsi. Kemudian blok kunci yang panjangnya 128-bit dirotasi dari kiri sepanjang 25-bit untuk dipartisi lagi menjadi 8 sub-blok kunci 16-bit berikutnya. Proses partisi dan rotasi diulangi terus sampai diperoleh 52 buah upa-kunci , yang panjang tiap upa-kuncinya 16-bit. Pada tabel berikut ditunjukkan proses pembangkitan kunci pada algoritma IDEA.

**Tabel 1 penggunaan knci di tiap putaran**

| Round | $Z_1^i$ | $Z_2^i$ | $Z_3^i$ | $Z_4^i$ | $Z_5^i$ | $Z_6^i$ |
|---|---|---|---|---|---|---|
| $i=1$ | 0–15 | 16–31 | 32–47 | 48–63 | 64–79 | 80–95 |
| $i=2$ | 96–111 | 112–127 | 25–40 | 41–56 | 57–72 | 73–88 |
| $i=3$ | 89–104 | 105–120 | 121–8 | 9–24 | 50–65 | 66–81 |
| $i=4$ | 82–97 | 98–113 | 114–1 | 2–17 | 18–33 | 34–49 |
| $i=5$ | 75–90 | 91–106 | 107–122 | 123–10 | 11–26 | 27–42 |
| $i=6$ | 43–58 | 59–74 | 100–115 | 116–3 | 4–19 | 20–35 |
| $i=7$ | 36–51 | 52–67 | 68–83 | 84–99 | 125–12 | 13–28 |
| $i=8$ | 29–44 | 45–60 | 61–76 | 77–92 | 93–108 | 109–124 |
| $i=9$ | 22–37 | 38–53 | 54–69 | 70–85 | | |

Pada setiap operasi pada kunci 128-bit, 8 kunci dengan panjang 16-bit diperoleh. Namun hanya enam kunci yang digunakan pada tiap putaran. Kunci yang tersisa disimpan untuk putaran berikutnya.

## 5. PENYERANGAN *RELATED-KEY* SEBAGAI AKIBAT PENGGUNAAN *KEY-SCHEDULE* , STUDI KASUS : ALGORITMA IDEA

Seperti yang telah disebutkan pada bagian 3 bahwa pada proses enkripsi algoritma IDEA terdapat tiga relasi non trivial berupa tiga operasi yang berbeda untuk pasangan sub-blok 16 –bit, yaitu XOR (prnambahan modulo 2), penambahan modulo $2^{16}$, dan perkalian dengan modulo $2^{16}+1$. Dengan menggunakan tiga relasi ini dan teknik-teknik lain, ditemukan penyerangan linier (*linier attack*) pada putaran ke-5 IDEA dengan menggunakan $2^{19}$ plainteks yang diketahui dan memiliki kompleksitas waktu enkripsi $2^{103}$ . Serangan jenis ini merupakan jenis serangan yang cukup terkenal pada algoritma IDEA. [1]

Selain penyerangan linier seperti yang dijelaskan di atas, algoritma IDEA juga rentan terhadap serangan *related*-key akibat penggunaan *key-schedule*. Seperti yang telah dipaparkan pada bagian 4, bahwa upa-kunci yang jumlahnya 52 buah dengan panjang masing-masing kunci 16-bit, dibangkitkan dari kunci utama yang panjangnya 128-bit. Dimana pada tiap putaran menggunakan 6 buah upa-kunci yang panjangnya 96-bit, lalu pada putaran final dilakukan transformasi dengan upa-kunci yang total panjangnya 64-bit. Dari sini dapat kita lihat bahwa *key-shedule*

pada algoritma IDEA sangat sederhana. Untuk membangkitkan (6n+4) 16-bit upa-kunci untuk n kali putaran IDEA, 128-bit kunci utama dipenuhi dengan upa-kunci ini. Karena *key-schedule* yang sangat sedehana inilah maka penyerangan diferensial dengan *chosen-key* pada putaran ketiga menjadi mungkin. Serangan ini dapat menyimpan 32-bit kunci (sama dengan 2 buah upa-kunci, dengan panjang tiap upa-kunci 16-bit) dengan menggunakan 6 plainteks yang dipilih (*chosen plaintext*), yaitu dua plainteks dengan kunci pertama, dan empat plainteks sisanya dengan kunci kedua. Penyimpanan 64-bit kunci lainnya membutuhkan $2^{17}$ *chosen plaintext* yang lain dengan 3 buah kunci. Dengan demikian sama saja dengan membiarkan penyimpanan 32-bit kunci dengan *exhaustive search*.

Dilakukan pengujian pada kata kedua dan ketiga pada tahap intermediate ekripsi, yait didapatkan relasi antara nilai pada kedua kata ini dan keluaran pada layer MA yang hanya menggunakan operasi XOR dan penjumlahan modulo tetapi bukan perkalian modulo. Misal ada plainteks P = = ($P_1$, $P_2$, $P_3$, $P_4$) dan menghasilkan cipherteks C = ($C_1$, $C_2$, $C_3$, $C_4$) maka [1]:

Ket simbol : □ = perasi penjumlahan modulo $2^{16}$
○ = operasi XOR

$((((((((((((((( P_2 \square Z^1_2) \circ u^1) \square Z^2_3) \circ t^2) \square Z^2_3) \circ u^3) \square Z^4_3) \circ t^4) \square Z^5_2) \circ u^5) \square Z^6_3) \circ t^6) \square Z^7_2) \circ u^7) \square Z^8_3) \circ t^8) \square Z^9_2) = C_2$

(1)

akan sama dengan :
$((((((((((((((( P_3 \square Z^1_3) \circ t^1) \square Z^2_2) \circ u^2) \square Z^3_3) \circ t^3) \square Z^4_2) \circ u^4) \square Z^5_3) \circ t^5) \square Z^6_2) \circ u^6) \square Z^7_3) \circ t^7) \square Z^8_2) \circ u^8) \square Z^9_3) = C_3$

(2)

Sekarang jika kita lihat nilai least significant bit (LSB) dari kata itu, maka operasi penambahan modulo akan menghasilkan hasil yang sama dengan operasi XOR, sehingga persamaan di ayas dapat ditulis :

$LSB(P_2 \circ Z^1_2 \circ u^1 \circ Z^2_3 \circ t^2 \circ Z^2_3 \circ u^3 \circ Z^4_3 \circ t^4 \circ Z^5_2 \circ u^5 \circ Z^6_3 \circ t^6 \circ Z^7_2 \circ u^7 \circ Z^8_3 \circ t^8 \circ Z^9_2) = LSB(C_2)$

(3)

Dan

$LSB(P_3 \circ Z^1_3 \circ t^1 \circ Z^2_2 \circ u^2 \circ Z^3_3 \circ t^3 \circ Z^4_2 \circ u^4 \circ Z^5_3 \circ t^5 \circ Z^6_2 \circ u^6 \circ Z^7_3 \circ t^7 \circ Z^8_2 \circ u^8 \circ Z^9_3) = LSB(C_3)$

(4)

Karena $u^i = t^i \square s^i$ maka $LSB(u^i) = LSB(t^i \square s^i)$ sehingga dapat pula disimpulkan bahwa $LSB(u^i \circ t^i) = LSB(s^i)$. Dengan demikian, XOR dari kedua persamaan di atas :

$LSB(P_2 \circ P_3 \circ Z^1_2 \circ Z^1_3 \circ s^1 \circ Z^2_2 \circ Z^2_3 \circ s^2 \circ Z^3_2 \circ Z^3_3 \circ s^3 \circ Z^4_2 \circ Z^4_3 \circ s^4 \circ Z^5_2 \circ Z^5_3 \circ s^5 \circ Z^6_2 \circ Z^6_3 \circ s^6 \circ Z^7_2 \circ Z^7_3 \circ s^7 \circ Z^8_2 \circ Z^8_3 \circ s^8 \circ Z^9_2 \circ Z^9_3) = LSB(C_2 \circ C_3)$.

(5)

Misalnya ada plainteks P1dan P2, maka perbedaan hasil operasi XOR kedua plainteks tersebut dalam nilai intermediate X, ditandai dengan $\Delta X$ adalah :

$LSB(P^1_2 \circ P^1_3 \circ P^2_2 \circ P^2_3 \circ \Delta S^1 \circ \Delta S^2 \circ \Delta S^3 \Delta S^4 \circ \Delta S^5 \circ \Delta S^6 \circ \Delta S^7 \circ \Delta S^8) = LSB(C^1_2 \circ C^1_3 \circ C^2_2 \circ C^2_3)$

(6)

Persamaan (6) di atas merupakan dasar dari semua serangan *related-key* yang mungkin pada IDEA , yaitu serangan *known-plaintekxt 5-round* , serangan *related-key* pada *7.5-round*, dan serangan *related-key rectangle* pada *7-round* IDEA.

Pada tabel berikut ditunjukkan beberapa pengujian serangan pada algoritma IDEA.

**Tabel 2 Pengujian Serangan pada IDEA**

| Putaran | Tipe Serangan | Kompleksitas | | Jumlah upa-kunci yang dipengaruhi |
|---|---|---|---|---|
| | | Data | Waktu | |
| 2 | differential | $2^{10}$ CP | $2^{42}$ | Semua |
| 2.5 | differential | $2^{10}$ CP | $2^{106}$ | Semua |
| 3 | Differential-linear | $2^{29}$ CP | $2^{44}$ | Semua |
| 3.5 | Linear | 103 KP/CP | $2^{97}$ | Semua |
| 3.5 | Square | $2^{22}$ CP | $2^{66}$ | Semua |
| 4 | Impossible Differential | $2^{37}$ CP | $2^{70}$ | Semua |
| 4 | Linear | 114 KP | $2^{114}$ | Semua |
| 4 | Square | $2^{23}$ CP | $2^{98}$ | Semua |
| 4.5 | Impossible Differential | $2^{64}$ CP | $2^{112}$ | Semua |
| 5 | Meet-in-The-Middle-Attack | $2^{24}$ CP | $2^{126}$ | Semua |
| 6.5 | Related-Key Rectangle | $2^{59.8}$ RK-CP | $2^{88.1}$ | Semua |
| 2.5 (*) | Linear | $2^{18}$ CP | $2^{18}$ | Semua |
| 3 | Linear | $2^{19}$CP | $2^{48.5}$ | Semua |
| 4.5 | Linear | 16CP | $2^{103}$ | Semua |
| 5 | Linear | $2^{19}$ KP | $2^{103}$ | Semua |
| 7.5 | Related-Key Linear | $2^{43.5}$ RK-KP | $2^{115.1}$ | Semua |
| 7 | Related-Key Rectangle | $2^{65}$ RK-CP | $2^{104.2}$ | Semua |

Keterangan tabel 2 : KP - Known Plaintext
CP – Chosen Plaintext
RK – Related Key
Kompleksitas waktu adalah jumlah unit ekripsi
*pembedaan dalam serangan

Serangan chosen plaintext yang paling dipublikasi pada algoritma IDEA adalah serangan pada putaran kelima IDEA yang membutuhkan $2^{24}$ *chosen plaintext* yang telah dienkripsi dan mempunyai kompleksitas waktu enkripsi $2^{126}$. Serangan *related-key* yang paling dipublikasi adalah serangan pada putaran ke 6.5 IDEA yang membutuhkan $2^{57.8}$ *chosen plaintext* yang telah dienkripsi di bawah empat *related-key* dan mempunyai kompleksitas waktu enkripsi $2^{88.1}$. [1] Berkaitan dengan serangan pada varian pengurangan putaran, beberapa kelas kunci lemah (*wek key*) di seluruh IDEA ditemukan. Kelas kunci lemah terbesar, yang diidentifikasi dengan teknik boomerang, mengandung $2^{64}$ kunci, dan pengujiannya membutuhkan $2^{16}$ *adaptive chosen-plaintext* dan cipherteks , serta mempunyai kompleksitas waktu $2^{16}$.

## 6. DESAIN *KEY-SCHEDULE* YANG KUAT

Seperti yang telah dijelaskan sebelumnya, bahwa kekuatana kriptografi algoritma IDEA terletak pada kombiasi tiga kelompok operasi yang *incompatible* : XOR (penjumlahan modulo 2), penambahan modulo $2^{16}$, dan perkalian modulo dengan $2^{16}+1$. Ketiga operasi tersebut penting untuk keamanan cipherteks. Tentu saja, jika miltiplikasi (perkalian dengan modulo $2^{16}+1$) tersebut dibuang, maka cipherteks dapat dengan mudah dipatahkan dengan memeriksa *least significant bits* (LSB) dari tiap kata selama proses enkripsi. Jika operasi XOR dihilangkan, maka

cipherteks hanya diturunkan dari operasi penambahan modulo $2^{16}$ dan emnjadi sangat mudah dipatahkan dengan menggunakan beberapa *known-plaintext*. Jadi, dapat disimpulkan bahwa jika salah satu dari ketiga operasi tersebut dibuang, maka cipherteks dapat dengan mudah dipatahkan.

Dengan melihat berbagai serangan yang mungkin dari penggunaan key-schedule seperti yang terlah dibahas dalam bagian sebelumnya, maka disimpulkan ada dua macam pendekatan untuk mencegah *related-key attack* yaitu dengan menambahkan proteksi beripa protokol tingkat tinggi, dan dengan memperkuat *key-schedule* agar tidak dapat dipatahkan dengan *related-key attack*.

Untuk pendekatan pertama, yaitu penggunaan protokol tingkat tinggi, maka buatlah protokol pergantian kunci yang tidak hanya menjamin *confidentiality* tetapi juga menjamin integritas saat pembentukan upa-upa kunci dari kunci utama.

Untuk pendekatan kedua, yaitu perkuat desain *key-schedule,* diajukan prinsip yaitu jangan atau minimalkan penggunaan *key-schedule* linier. Seperti kita lihat pada tabel 1, bahwa pada beberapa putaran algoritma IDEA sangat rentan terhadap serangan pada linier *key-schedule*. Setiap perubahan bit pada pembentukan upa-kunci seharusnya mempengaruhi pembentukan upa-kunci berikutnya, dan *key-schedule* sebaiknya didesain untuk kuat terhadap *differential attack*.

Jika anda menggunakan akan menerapkan algoritma *key-schedule* yang mungkin rentan terhadap berbagai serangan (seperti yang dipaparkan pada bagian sebelumnya), maka **prinsipnya** : masukkan kunci

utama (yang dari pengguna) ke suatu fungsi hash yang kuat, sebelum kunci tersebut diperluas dengan algoritma key-schedule.

Dengan demikian, serangan seperti yang dijelaskan pada bagian sebelumnya, menjadi dipesulit.

## 7. KESIMPULAN

Algoritma IDEA sebenarnya cukup kuat, namun rentan terhadap berbagai bentuk serangan yang berhubungan dengan algoritma *key-schedule* yang diterapkannya. Untuk itu telah diajukan pendekatan dalam desain key-schedule agar lebih kuat, salah satunya bahwa dalam mendesain protokol pergantian kunci , aspek integritas juga perlu diperhatikan, tidak hanya *confidentiality*. Walaupun *key-schedule* linier pada IDEA sangat rentan terhadap serangan, secara keseluruhan algoritma ini tetap dapat bertahan terhadap berbagau serangan tersebut [1].

## DAFTAR REFERENSI


[1] Biham, Eli. Orr Dunkelman, dan Nathan Keller. *New Cryptanalytic Result on IDEA*, Israel.

[2] Munir, Rinaldi. Diktat Kuliah IF5054 Kriptografi. STEI. 2006

[3] Kelsey,John. Bruce Sneier, dan David Wagner. *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Berkeley.

[4] R. Winternitz, M.Hellman, "*Chosen-key Attack on Block Cipher*", v.11, n.1, 1987.

[5] Schneier, Bruce. *Applied Cryptography 2nd* . John Wiley & Sons.1996.