Studi Jenis Serangan Umum Pada *Smart Card* dan Pengembangan Keamanan *Smart Card* Untuk Mengatasi Serangan-Serangan Umum

Andree Datta Adwitya – NIM : 13503062

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : datta@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang jenis-jenis serangan pada *smart card* dan studi pengembangan keamanan dari aplikasi *smart card.* Banyak sekali tipe serangan yang dapat dilakukan terhadap *smart card*. Serangan-serangan yang dimaksud dapat ditujukan kepada perangkat keras maupun perangkat lunak dari *smart card*. Dalam makalah ini serangan-serangan tersebut akan dibahas satu per satu. Kode yang tidak sensitif terhadap tenggat waktu maupun kode yang tidak teratur dapat digunakan untuk membuat sirkuit yang *robust* secara asinkron dan sistem yang dapat mengecek dirinya sendiri. Sifat redundan dari skema pemrograman juga memberikan kemungkinan implementasi yang seimbang. Dalam makalah ini pun akan didemonstrasikan bagaimana karakteristik-karakteristik tersebut dapat digunakan untuk membangun fungsi-fungsi *smart card* yang tahan terhadap serangan *side-channel* dan *fault injection*.

Kata kunci: *Asynchronous logic, smart card,*analisis,energi diferensial, analisis pancaran elektromagnetik.

1. Pendahuluan

Smart card adalah sebuah media penyimpan berbagai macam data yang dapat digunakan untuk berbagai aktivitas. Akhir-akhir ini, penggunaan dari *smart card* lebih banyak diprioritaskan dan berkembang di bidang otentikasi dan mekanisme pembayaran, seperti kartu kredit, akses kondisional televisi berbayar, pembayaran kendaraan umum, rekaman medis, identitas pribadi, kartu SIM telepon genggam, kunci sekuritas mobil, dan lain sebagainya.

Keamanan dari sistem seperti sistem *smart card* bergantung kepada kemampuan dari *smart card* itu sendiri untuk melakukan operasi kriptografi sambil tetap menyimpan kunci secara rahasia di dalam kartu. Perhatian lebih diberikan kepada serangan-serangan yang menyediakan kemampuan untuk membuka informasi kunci rahasia yang relatif cukup cepat dan tidak meninggalkan bukti bahwa telah terjadi serangan.

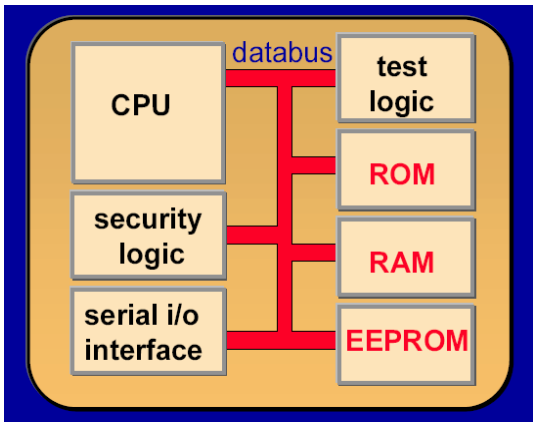
Serangan *non-invasive* atau serangan *side-channel* mengekstraksi kunci dengan menganalisa kebocoran informasi dalam bentuk total penggunaan daya atau energi, emisi elektromagnetik, dan *timing*. Tambahan untuk mengatur kartu, aktivitas sirkuit dapat dipengaruhi secara eksternal dengan memodulasi daya/energi atau *clock pin*. Percobaan lain untuk menyadap aktivitas switching dapat juga dilakukan dengan menggunakan sinyal elektromagnetik (*Electro Magnetic Pulse*). Teknik-teknik seperti diatas sering digunaan dalam usaha untuk merusak kartu dalam cara yang mudah diprediksi dan teratur.

Sebagai pelengkap dari serangan *non-invasive*, seringkali diperlukan asumsi bahwa detil dari desain dapat dicari melalui *reverse engineering*. Dalam kasus ini, hanya inilah cara seorang penyerang dapat mendeteksi dan mempengaruhi aktivitas sirkuit yang membatasi kemampuan mereka untuk mengekstraksi kunci. Ancaman yang saat ini ada yaitu adalah penggunaan injeksi kesalahan optikal. Seorang penyerang dapat sinar laser untuk membuat grup transistor yang diincar untuk bekerja pada waktu yang diinginkan.

Dalam makalah ini akan dibahas semua jenis serangan terhadap *smart card* yang disebutkan diatas, serta dijabarkan pula teknik-teknik untuk improvisasi keamanan *smart card* dengan membatasi derajat secara keras dimana aktivitas sirkuit dapat dimonitor atau dipengaruhi secara eksternal.

1.1 Isi dari *smart card*

Skema dari isi *smart card* dapat dilihat pada gambar 1.



Gambar 1: Skema internal *smart card*

Penjelasan dari Gambar 1.

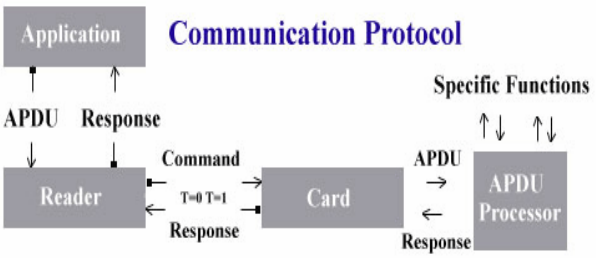
- CPU(*Central Processing Unit*) adalah jantung utama dari chip.
- *Security logic* digunakan untuk mendeteksi kondisi abnormal, seperti kondisi cahaya.
- *Serial i/o interface* adalah antarmuka yang berkomunikasi dengan benda diluar chip.
- *Test logic* berisi sekumpulan prosedur untuk melakukan *self-test* pada chip.
- ROM berisi sistem operasi dari kartu. Besar dari ROM biasanya 16 *kilobyte*, akan tetapi untuk masa depan sudah dikembangkan ROM dengan ukuran 32 dan 64 *kilobyte*.
- RAM dapat dianalogikan seperti kertas coretan untuk CPU yang berguna untuk lalu lintas data dari dan ke CPU. Biasanya ukuran RAM adalah 512 *byte*, namun saat ini sudah dikembangkan RAM dengan ukuran 1 *kilobyte*.
- EEPROM berisi data-data rahasia yang disimpan oleh kartu. Data-data rahasia ini antara lain, kunci-kunci kiptografi, kode PIN, template *biometrik*, dan kode aplikasi. EEPROM biasanya berukuran 8 *kilobyte*, dan untuk masa depan sudah dikembangkan EEPROM berukuran 32 *kilobyte*.
- Databus bertugas untuk menghubungkan seluruh elemen dari chip. Biasanya databus memiliki lebar 8 atau 16 bit.

1.2 Komunikasi *smart card* dengan dunia diluarnya

Sebuah *smart card* dan alat pembaca kartu berkomunikasi melalui paket data kecil bernama APDU (*Application Protocol Data Units*). Karakteristik-karakteristik yang membuat pihak ketiga kesusahan untuk menyerang sistem adalah:

- *Bit rate* yang kecil (9600 bit per detik) menggunakan sebuah jalur transmisi serial dua arah (ISO standard 7816/3).
- Mode *half duplex* untuk mengirimkan informasi (data hanya berjalan dalam satu arah dalam satu waktu)
- Komunikasi mengikuti protokol bayangan seperti dilihat pada gambar 2 dan dijelaskan dibawah.

Akan tetapi, setiap alat eksternal yang berkomunikasi dengan kartu, membuat kartu lebih rentan untuk diserang melalui jalur komunikasi.



Gambar 2: Protokol komunikasi *smart card* dengan *card reader*

Smart card dan alat pembaca kartu protokol otentikasi yang saling berbalasan secara aktif untuk mengidentifikasi lawannya masing-masing. Kartu menciptakan sebuah angka acak

dan mengirim nomor tersebut ke alat pembaca kartu yang akan mengenkripsi angka tersebut dengan kunci enkripsi yang di-*shared* sebelum akhirnya hasil enkripsi tersebut dikirim kembali ke kartu. Kartu kemudian akan membandingkan hasil yang dikembalikan oleh alat pembaca kartu dengan hasil enkripsi dirinya sendiri. Pasangan tersebut nantinya dapat melakukan operasi tersebut sebaliknya.

Setelah komunikasi dapat dibangun, setiap pesan antara pasangan tersebut diverifikasi melalui *message authentication code* (MAC). MAC ini adalah angka yang dihasilkan oleh proses yang melibatkan pesan itu sendiri, kunci enkripsi, dan angka acak. Jika data sudah diubah (untuk segala alasan, termasuk kesalahan ketika transmisi data), pesan harus ditransmisi ulang. Atau ada alternatif lain, yaitu jika chip mempunyai memori yang cukup dan tenaga untuk memproses, data dapat diverifikasi melalui tanda tangan digital.

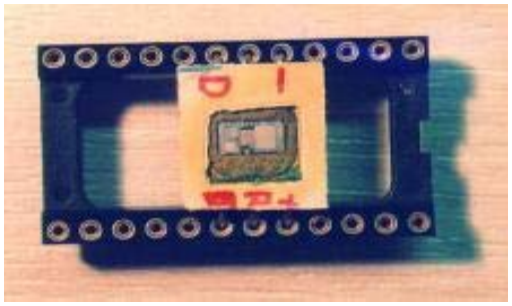
Metode enkripsi yang paling umum adalah DES (*Data Encryption Standard*) simetri, 3DES (*triple* DES), Algoritma kunci publik RSA (algoritma Rivest-Shamir-Adleman's), yang dapat menampung panjang kunci sampai 56, 168, dan 1024 bit. Akan tetapi, kunci-kunci ini bukannya tidak dapat dipecahkan. Seperti dijelaskan oleh Ross Anderson dan Markus Kuhn dalam buku mereka yang berjudul `Design Principles for Tamper-Resistant Smart Card Processors'. Kedua orang ini berhasil memecahkan Dallas DS5002FP *Secure Microcontroller*, yang pada sat itu dikumandangkan oleh satu agensi intelijen di eropa sebagai prosesor paling aman yang pernah tersedia yang dijual secara umum. Mereka menggunakan metode *brute force* dengan sebuah PC yang dimodifikasi dengan perangkat keras ekstra bernilai beberapa ratus dolar amerika.

Kartu dan alat pembaca kartu berkomunikasi melaui sebuah set instruksi spesial. Sebagai contoh, set instruksi Schlumberger Reflex 60 mengandung:

0x60	Gets reader type and activate reader
0x61	Sets reader with ICC communication parameters
0x62	Turns card power ON
0x63	Turns card power OFF
0x64	Sends RESET signal to card
0x65	Gets reader-card status
0x66	Sends one byte to reader
0x67	Sends data block to reader
0x68	Makes reader resend last data block
0x69	Gets reader capabilities
0x6A	Deactivate reader
0x6B	Activate reader-dependent features
0x6C-0x6F	Reserved

Tabel 1: Set instruksi Schlumberger Reflex 60

1.3 Keamanan perangkat keras

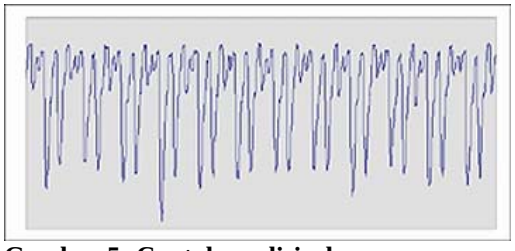


Gambar 3: Perbandingan ukuran *smart card* dengan IC biasa

Seluruh data dan kata kunci dalam kartu disimpan dalam EEPROM dan dapat dihapus atau dimodifikasi dengan suplai tegangan yang tidak biasa. Untuk itu, beberapa processor yang aman mengimplementasikan sensor untuk perubahan lingkungan. Akan tetapi, karena menemukan level sensitivitas yang benar sangat susah dan ketika daya disuplai ke kartu akan terjadi fluktuasi di tegangan, metode ini sangat jarang digunakan. Metode serangan lainnya yang biasanya sukses termasuk memanaskan kontroler sampai kepada suhu tinggi atau memfokuskan sinar ultra violet ke EEPROM, yang akan menyingkirkan kunci keamanan. Serangan fisik secara keras adalah yang paling destruktif ketika kartu dipotong dan prosesor dibuang. Lalu layout dari chip dapat dilihat dengan *reverse engineering*.

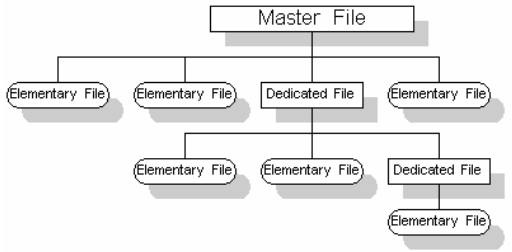
Gambar 4: Chip *smart card* yang sudah dipotong

Analisis daya diferensial adalah sebuah serangan statistikal terhadap algoritma kriptografi yang membandingkan sebuah hipotesis dengan keluaran yang terukur dan seringkali mampu untuk mengekstraksi sebuah kunci enkripsi dari *smart card* ataupun alat komputasi lainnya. Analisis daya sederhana adalah analisis secara langsung dari data daya yang terekam untuk menentukan aksi dan data yang juga cukup berguna.



Gambar 5: Contoh analisis daya

1.4 Keamanan sistem operasi



Gambar 6: Bagan hierarki data dalam *smart card*

Data dalam *smart card* terorganisasi dalam tiga hierarki. Bagan data ini memiliki satu *master file* yang mengandung beberapa *elementary file* dan beberapa *dedicated file*. *Dedicated file* dan *master file* berkorespondensi terhadap direktori dan *elementary file* berkorespondensi terhadap *file*, yang dapat dianalogikan dengan hierarki pada semua sistem operasi umum untuk PC. Akan tetapi, kedua hierarki ini berbeda di *dedicated file* yang juga dapat mengandung data. *Header* dari *dedicated file*, *elementary file* dan *master file* mengandung atribut keamanan yang menyerupai hak *user* yang terasosiasi dengan *file* atau direktori dalam sistem operasi umum. Aplikasi apapun dapat menjelajahi bagan file tersebut, namun hanya dapat berpindah ke node yang lain jika mempunyai hak yang tepat.

1.4.1 Atribut (Hak Akses)

Ada lima level dasar dari hak akses ke *file* untuk *dedicated file* dan *elementary file*. Beberapa sistem operasi menyediakan level yang lebih lanjut. Level dasar dapat dikategorikan terurut menaik dalam tingkat keamanan seperti berikut ini:

1. **Always (ALW):** Akses dari *file* dapat dilakukan tanpa pembatasan apapun.

2. **Card holder verification 1 (CHV1):** Akses hanya mungkin ada jika sebuah nilai valid dari CHV1 disediakan.

3. **Card holder verification 2 (CHV2):** Akses hanya mungkin ada jika sebuah nilai valid dari CHV2 disediakan.

4. **Administrative (ADM):** Alokasi dari level ini dan masing-masing kebutuhan untuk pemenuhan mereka adalah tanggung jawab dari otoritas administrasi yang bersangkutan.

5. **Never (NEV):** Akses terhadap *file* dilarang.

Akan tetapi, menggunakan CHV2 tidak memberikan akses ke *file* yang membutuhkan CHV1. CHV1 dan CHV2 berkorespondensi kepada dua PIN keamanan yang disimpan di dalam kartu: satu adalah sebuah PIN identifikasi user secara umum dan yang satu lagi adalah PIN untuk mengnonaktifkan blokir secara spesifik yang disimpan lebih dahulu di dalam kartu.

1.4.2 PIN (*Personal Identification Number*)

PIN disimpan pada *elementary file* (EF) yang berbeda, contoh EF_{CHV1} and EF_{CHV2}. Sistem operasi memblokir kartu setelah PIN yang

salah dimasukkan beberapa kali secara berurutan. Jumlah kesalahan yang diperbolehkan sudah tetap dan tergantung kepada sistem operasinya. Sekali diblokir, kartu hanya dapat diaktifkan kembali dengan PIN untuk mengnonaktifkan blokir yang spesifik yang disimpan dalam kartu. PIN untuk mengnonaktifkan blokir pun dapat terblokir dengan cara yang sama. Jika hal ini terjadi, kartu dalam kondisi blokir yang tidak dapat dibalikkan dan mungkin harus dihancurkan untuk alasan keamanan.

Jika PIN diblokir, atribut dari setiap *file* diubah sehingga membutuhkan CHV1. Setelah PIN untuk mengnonaktifkan blokir dikeluarkan, atribut dari *file* dikembalikan ke asalnya, *counter* untuk PIN diset kembali ke nilai maksimal, dan *counter* untuk PIN yang mengnonaktifkan blokir diturunkan. Jika *counter* yang terakhir menunjukkan nilai nol, maka PIN tersebut tidak dapat digunakan untuk mengnonaktifkan blokir lagi. Ini memberikan keamanan tambahan untuk *smart card*.

1.5 Keamanan perangkat lunak

Produsen perangkat lunak juga berkontribusi terhadap keamanan *smart card*. Mereka seharusnya menyediakan produk mereka dengan data dan transfer yang dienkripsi secara benar. Untuk membantu mereka mencapai tujuan ini, dikembangkanlah instruksi *hardware-based* atau *OS-based* dan *libraries* untuk mendukung algoritma kriptografi tingkat lanjut.

2. Serangan pada *smart card*

Serangan yang akan dibahas pada makalah ini dibatasi hanya serangan bertipe *side-channel* dan injeksi kesalahan optikal.

2.1 Serangan s*ide-channel*

Serangan *side-channel* pada *smart card* dilakukan untuk membuka kunci rahasia yang ada di dalam *smart card*. Serangan jenis seperti ini sudah cukup lama dilakukan oleh militer. Kebocoran *side-channel* terjadi melalui variasi dependansi data dalam penggunaan sumber daya seperti waktu dan perangkat keras. Untuk sistem kriptografi RSA, pengurangan modular kondisional harus dihilangkan untuk membuat waktu menjadi konstan. Aktivitas bus adalah penyebab dominan dari naik turunnya daya, dengan hubungan yang kuat antara naik turunnya daya tersebut dengan berat Hamming dari data didalam bus. Instruksi dan lokasi memori bergerak melewati bus dan, dalam konteks sumber daya komputasi yang terbatas dari *smart card*, juga potensial data yang berjumlah cukup banyak. Hal ini dapat dipecahkan secara sebagian oleh enkripsi dari bus.

Untuk semua sistem kriptografi popular yang digunakan dalam kehidupan dimana panjang kuncinya bisa dibuat berbeda untuk setiap penggunaan, semakin panjang kunci tersebut, dipercaya semakin hebat pula kekuatan secara matematis dari sistem untuk melawan serangan. Memang, serangan secara *brute force* akan memakan waktu secara eksponensial dalam panjang kunci. Akan tetapi, semakin panjang kunci maka semakin banyak pula komputasi yang dibutuhkan baik pada enkripsi dan dekripsi maupun pada pemberian tanda tangan digital dan verifikasi.

Dalam sistem kriptografi simetris seperti DES, 3-DES, dan AES, panjang dari blok sudah ditetapkan dan jumlah pengulangan proporsional terhadap panjang kunci. Oleh karena itu, kebocoran data pun juga proporsional terhadap panjang kunci dan kekuatan dari implementasi cipher hampir jarang berkurang seiring dengan bertambahnya panjang kunci.

Akan tetapi, kriptografi kunci publik seperti RSA, DSA, ECC, atau Diffie-Hellman, biasanya melibatkan eksponensiasi dalam beberapa bentuk, dimana panjang blok dan eksponen proporsional terhadap panjang kunci. Dengan asumsi bahwa multiplikasi dari argumen dengan panjang dua kali standar membutuhkan waktu empat kali lipat dibandingkan waktu untuk argumen dengan panjang standar dengan perangkat keras yang sama, total waktu untuk dekripsi atau memberi tanda tangan proporsional terhadap kuadrat dari panjang kunci. Konsekuensinya, kebocoran data yang lebih banyak per bit kunci seiirng dengan semakin panjangnya kunci. Dan memang, jika operasi multiplikasi dari eksponensiasi dilakukan secara sequensial dengan menggunakan salah satu algoritma standar, maka akan lebih banyak data per bit eksponen untuk setiap kunci yang lebih panjang dan kekuatan dari implementasinya akan berkurang.

Dua serangan mengilustrasikan bahwa mungkin saja serangan-serangan tersebut memberikan hasil dengan memanjangkan kunci. Yang pertama adalah serangan *timing* yang kesuksesannya bisa dilakukan untuk

kunci yang panjangnya sangat pendek maupun sangat panjang. Sesuai dengan sifatnya, serangan ini mengasumsikan bahwa tidak ada satupun dari anti-serangan yang sudah ada yang bisa menjamin tidak adanya variasi waktu yang bergantung pada data dan membuat variasi random dalam eksponen. Setiap bit kunci ditemukan dengan data yang proporsional terhadap kuadrat dari panjang kunci.

Serangan kedua membutuhkan alat monitoring yang jauh lebih mahal dan menggunakan analisis daya dan/atau *electro-magnetic radiation* (EMR). Serangan ini diaplikasikan ke eksponensiasi tunggal sehingga dapat menghindari kesulitan yang diakibatkan dari penggunaan *exponent-blinding* sebagai anti serangan. Bit-bit kunci ditemukan secara independen menggunakan semua data yang tersedia, dengan data yang proporsional terhadap kuadrat dari panjang kunci. Bit eksponen individual sekarang teridentifikasi dengan benar dengan akurasi yang meningkat cukup cepat untuk mengkompensasi penambahan jumlah bit kunci. Konsekuensinya serangan menjadi lebih mudah dengan panjang kunci yang semakin panjang.

2.1.1 Model keamanan

Konteks untuk kedua serangan diatas pada bagian 2.1 sedikit berbeda, namun untuk kenyamanan dalam kedua kasus kita mengasumsikan skenario yang mirip namun realistis. Untuk setiap serangan, sebuah *smart card* mengerjakan RSA dengan sumber daya yang terbatas dan harus dapat digunakan kembali setelah serangan. Penyerang akhirnya hanya dapat melakukan kegiatan yang terbatas: ia hanya bisa memonitor kebocoran *side-channel*. Ia tidak dapat memilih input apapun maupun membaca input atau output. Dalam kebanyakan sistem kriptografi yang didesain dengan baik, I/O akan tidak terlihat dan penyerang hanya akan dapat melihat paling banyak data yang tidak tersembunyi yang tidak digunakan secara langsung dalam komputasi yang diawasi. Akan tetapi, penyerang diijinkan untuk mengetahui algoritma yang digunakan, mungkin sebagai hasil dari studi destruktif sebelumnya dari kartu-kartu yang identik, informasi dari dalam, dan spesifikasi publik. Tujuan dari penyerang adalah untuk menentukan eksponen rahasia D dengan menggunakan *Chinese Remainder Theorem* ataupun tidak, dan ia dapat menggunakan pengetahuan dari modulus publik M dan eksponen publik E untuk mengkonfirmasi nilai yang keluar. Diasumsikan bahwa algoritma eksponensiasi m-ary digunakan, tetapi argumen yang mirip dapat digunakan untuk algoritma tradisional lainnya.

Serangan *timing* mengasumsikan penggunaan algoritma multiplikasi modular yang menyertakan sebuah pengurangan kondisional yang bersifat final dari modulus. Kita mengasumsikan konsekuensi dari variasi *timing* membuat penyerang mampu merekam hampir semua tahap dalam pengurangan secara akurat. Ia kemudian mengamati sejumlah eksponensiasi dimana eksponen yang tidak tersembunyi digunakan. Kita akan mendemonstrasikan serangan ini dengan implementasi dari metode Montgomery yang akan dijabarkan dalam bagian setelah ini.

Penggunaan daya, dan juga EMR, bervariasi sesuai dengan jumlah dari aktivitas *switching* di dalam sirkuit. Jumlah gerbang rata-rata yang yang dihubungkan dalam *multiplier* mendekati secara linear dari jumlah dari berat Hamming dari input-input, dan hal yang sama berlaku untuk bus-bus yang membawa I/O dari dan ke *multiplier*. Jadi dengan menggunakan kombinasi dari pengukuran daya dan EMR dari kutub yang ditempatkan secara hati-hati, dapat diasumsikan bahwa seorang penyerang dapat mengambil beberapa data, paling sedikit untuk yang berkaitan dengan jumlah dari berat Hamming dari input-input. Masalah dari penyerang adalah untuk mengkombinasikan data-data ini dengan suatu cara yang akan menunjukkan berat Hamming dengan akurasi yang cukup untuk mengurangi jumlah digit dari eksponen. Serangan analisis daya diferensial menunjukkan bagaimana cara ini bisa dilakukan dengan pengamatan dari eksponensiasi tunggal.

2.1.2 Notasi

Seperti diatas, kita mengasumsikan sebuah n-bit, modulus M, dan eksponen privat D untuk sistem kriptografi RSA. Cipherteks C harus dikoenversi menjadi plainteks $C^D \bmod M$ dengan menggunakan k-bit *multiplier* yang kecil dan tunggal. Oleh karena itu, kecuali untuk eksponen, jumlah n-bit X yang terlibat dalam eksponensiasi direpresentasikan menggunakan basis $r = 2^k$ dan (non-redundan) digit x_i $(0 \le i < s)$ dalam *range* $[0,r)$. Sehingga $X = \sum(i=0, s\text{-}1)\ x_i r^i$ dan $n = ks$.

Eksponen D direpresentasikan dengan basis berbeda m, biasanya 2 atau 4, tergantung dari algoritma eksponensiasinya. Eksponensiasi biasanya dilakukan dengan menggunakan

algoritma biner "square-and-multiply", memproses bit-bit eksponen dalam urutan apapun, atau generalisasi dari kasus *most-to-least siginificant*, yang biasa disebut eksponensiasi *m*-ary, dimana *D* direpresentasikan dalam radial *m* menggunakan *t* digit dan beberapa daya dari $C^{(i)} = C^i \bmod M$ $(1 \leq i < m)$ yang diprekomputasi seperti dibawah ini:

THE *m*-ARY (MODULAR) EXPONENTIATION ALGORITHM

```
C(1) ← C ;
For i ← 2 to m-1 do
    C(i) ← C(i-1) × C mod M ;
P ← C(d_{t-1}) ;
For i ← t-2 downto 0 do
Begin
    P ← P^m mod M ;
    If d_i ≠ 0 then P ← P×C(d_i) mod M ;
End ;
```

OUTPUT: $P = C^D \bmod M$ for $D = \sum_{i=0}^{t-1} d_i m^i$

Rumus 1: Algoritma eksponensiasi *m*-ary

Hasil modular disini terlalu besar untuk satu operasi oleh *multiplier* pada *smart card*. Biasanya, digunakan sebuah bentuk dari algoritma Montgomery's *modular multiplication* (MMM). Ini memberikan output yang berhubungan dengan (*A* X *B*) mod *M* melalui sebuah faktor pembanding $R = r^8$ yang ditentukan dari jumlah digit pada input *A*. Hasil dari ini menyertakan sebuah pengurangan kondisional akhir yang mereduksi output menjadi lebih kecil dari *M*, tapi menciptakan variasi dalam waktu yang digunakan.

MONTGOMERY'S MODULAR MULTIPLICATION ALGORITHM (MMM)

```
P := 0 ;
For i := 0 to s-1 do
Begin
    P  := P + a_i×B ;
    q_i := (-p_0 m_0^{-1}) mod r ;
    P  := (P + q_i×M) div r ;
End ;
If P >= M then P := P-M
```

OUTPUT: $P = ABr^{-s} \bmod M$ for $A = \sum_{i=0}^{s-1} a_i r^i < M$ and $B < M$.

Rumus 2: Algoritma Montgomery's *modular multiplication*

Disini m_0^{-1} dibawah mod *r* adalah sisa unik dari modulo *r* dengan properti m_0^{-1} X $m_0 \equiv 1$ mod *r*. Masih mirip dengan yang sebelumnya, r^{-s} yang muncul dibawah mod *M* adalah inverse dari r^s modulo *M*.

2.2 Injeksi kesalahan optikal

Mikrokontroler dan *smart card* yang aman didesain untuk memproteksi baik kerahasiaan maupun integritas dari informasi yang sensitif. Tidaklah cukup untuk mencegah seorang penyerang untuk mencari tahu nilai dari kunci kriptografi yang disimpan. Alat-alat tersebut juga harus tidak bisa membuat bagian dari kunci menjadi nilai yang mudah diketahui, atau untuk memperbaiki kesalahan dalam komputasi yang dapat menyebabkan informasi yang sensitif dapat dibaca. Kesalahan-kesalahan ini mungkin saja kesalahan data, seperti tanda tangan digital yang tidak benar yang dapat membocorkan nilai dari kunci pemberi tanda tangan, atau kesalahan dalam kode, seperti lompatan kondisional yang terlewatkan yang mengurangi jumlah putaran dalam cipher blok. Sampai sekarang, teknik yang paling terkenal untuk memperbaiki kesalahan-kesalahan seperti ini adalah glitching, pengenalan tegangan sesaat kedalam daya dari *clock line* dari chip yang menjadi target. Kebanyakan chip sekarang sudah didesain untuk dapat bertaahan melawan serangan *glitch*.

Serangan dapat bersifat *invasive*, dengan menggunakan alat pengetesan chip seperti *probing station* dan *focused ion beam workstation* untuk mengekstrak data dari chip secara langsung. Atau serangan juga dapat bersifat *non-invasive* yang melibatkan eksploitasi dari emisi elektromagnetik tanpa tujuan, kebocoran desain protocol, dan kerawanan lainnya. Kedua jenis serangan dapat bersifat aktif maupun pasif. Standar serangan *invasive* yang pasif menggunakan *microprobe* untuk memonitor bus dari *smart card* selagi mengeksekusi sebuah program; sedangkan dalam serangan yang aktif, sinyal dapat juga diinjeksikan. Contoh klasik adalah penggunaan jarum *grounded microprobe* dari *clock line* sampai ke *instruction latch* untuk mengnonaktifkan instruksi *jump*. Sebuah serangan *non-invasive* yang pasif adalah menganalisa medan elektromagnetik di sekitar alat yang sedang dites. Sedangkan contoh klasik dari yang bersifat aktif adalah *glitching*.

Sampai sekarang, serangan *invasive* selalu terkait dengan investasi kapital yang relatif besar untuk perlengkapan lab ditambah investasi usaha untuk setiap chip yang diserang. Serangan *non-invasive* seperti analisis daya hanya membutuhkan investasi

kapital yang masih dalam batas kewajaran ditambah investasi usaha dalam mendesain sebuah serangan ke alat dengan tipe tertentu, sehingga biaya yang dikeluarkan per alat yang diserang menjadi relatif lebih rendah. Hal ini menjadikan serangan *non-invasive* lebih atraktif.

Disayangkan untuk para penyerang, banyak produsen chip saat ini telah mengimplemnetasikan pertahanan terhadap bahkan serangan *non-invasive* yang terdahsyat sekalipun. Pertahanan ini mencakupi *randomized clocking* untuk membuat analisis daya menjadi lebih susah, dan sirkuit yang bereaksi terhadap *glitch* dengan mereset prosesor. Sementara itu, serangan *invasive* secara konstan menjadi lebih berat dan mahal seiring dengan semakin kecilnya ukuran fitur dan semakin besarnya kompleksitas dari alat.

Namun ada sebuah kelas serangan baru yang bisa disebut sebagai "semi-invasive". Serangan ini, seperti serangan *invasive*, membutuhkan pemotongan chip to mendapatkan akses ke permukaan chip. Tetapi lapis *passivation* dari chip tetap ada – metode *semi invasive* tidak membutuhkan kontak secara elektrikal dengan dasar metal atau melakukan perusakan mekanis ke silikon.

Serangan *semi invasive* tidak sepenuhnya baru. Serangan *semi invasive* secara teori dapat dilakukan dengan menggunakan media seperti sinar ultra violet, sinar X, laser, medan elektromagnetik, dan pemanasan lokal. Media-media ini dapat digunakan secara terpisah maupun digabungkan satu dengan lainnya. Hanya saja metode ini tidak begitu banyak dieksplorasi.

Ketika transistor semikonduktor ditemukan, transistor memiliki sifat yang lebih sensitif terhadap radiasi ion – baik yang disebabkan oleh ledakan nuklir, isotop radioaktif, sinar X, ataupun sinar kosmis – daripada katup thermionic seperti transistor sebelumnya. Di tahun enam puluhan, ketika eksperimen dengan laser bergelombang denyut, ditemukan bahwa sinar koheren mengakibatkan fenomena yang sama. Laser akhirnya mulai digunakan untuk mensimulasi efek dari radiasi ion pada semikonduktor.

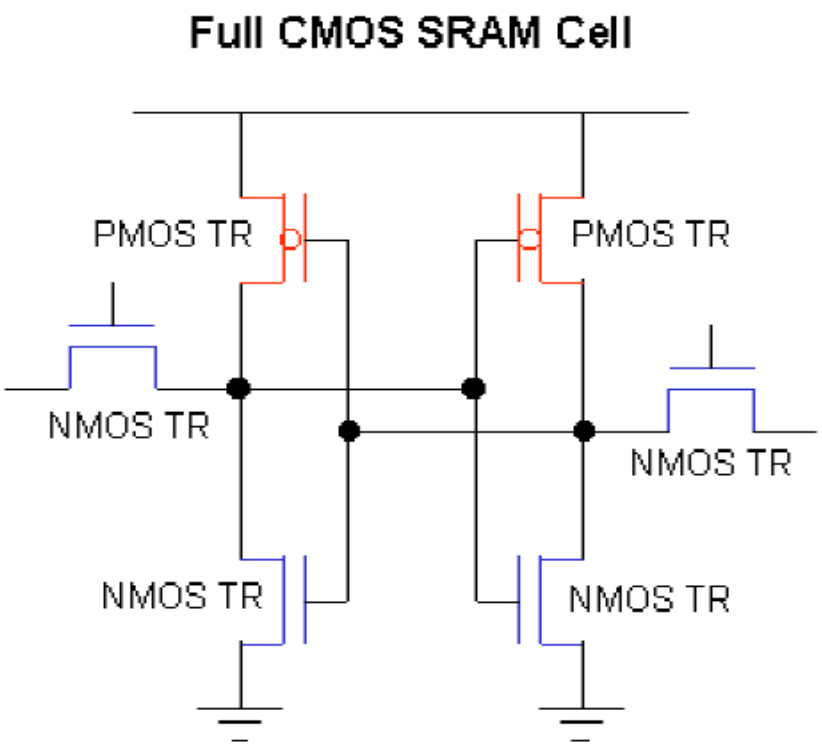
Semenjak saat itu, teknologi berkembang secara pesat. Laser berbasis gas dan benda padat yang mahal sudah digantikan dengan laser semikonduktor yang memiliki biaya rendah. Sebagai hasilnya teknologi telah berpindah dari laboratorium ke alat-alat elektronik yang bersifat konsumtif.

Radiasi laser akan mengionisasi wilayah semikonduktor dari *integrated circuit* (IC) jika energi fotonnya melewati celah frekuensi semikonduktor. Akan tetapi, fokus dari laser terbatas oleh disperse sebesar beberapa mikro meter, dan ini kurang begitu presisi untuk alat semikonduktor modern. Namun, ketika berpindah dari infra merah ke chaya tampak, absorpsi foton naik secara drastis, dan akhirnya memungkinkan untuk menggunakan laser merah dan hijau seiring dengan transistor di chip-chip modern semakin lama semakin tipis. Alat yang lebih kecil juga berarti semakin kecil energi yang dibuthkan untuk mencapai level ionisasi yang sama.

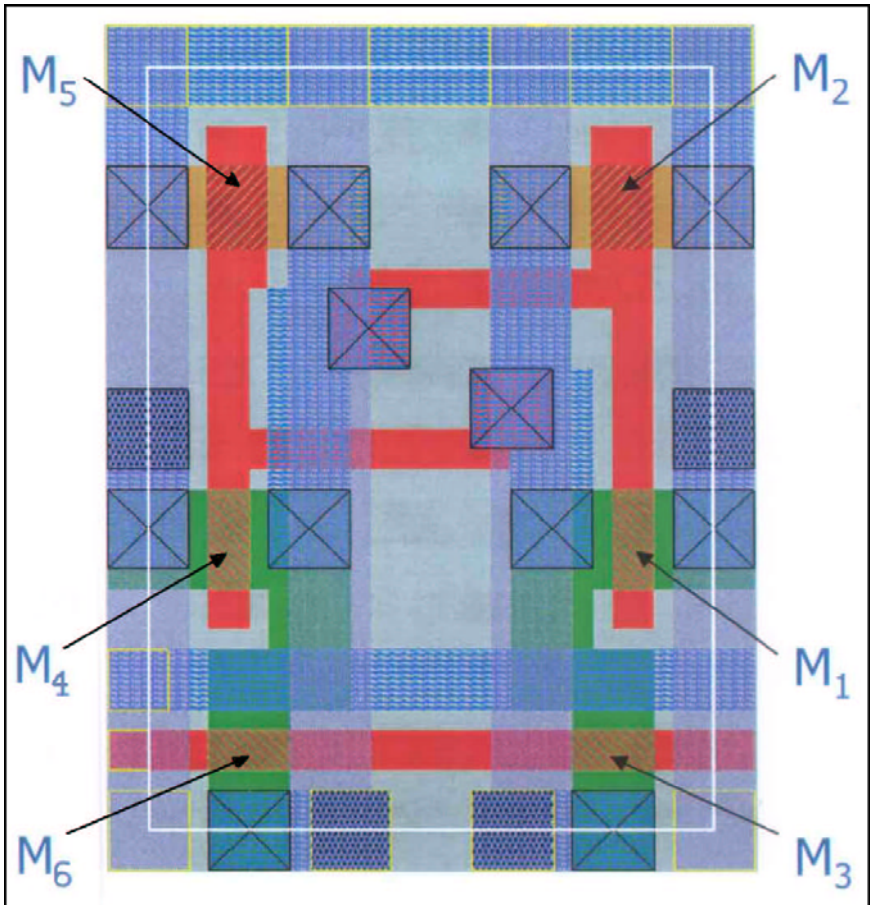
Dalam kasus alat CMOS, ada bahaya ketika mengakses sirkuit, yang akan membuat sirkuit menjadi terbuka dan rusak. Jadi penggunaan radiasi dengan struktur CMOS harus dilakukan dengan kehati-hatian yang sangat mendalam.

Walaupun sudah banyak publikasi tentang penggunaan laser berdenyut untuk mensimulasi radiasi ion, tidak pernah ditemukan informasi yang dipublikasikan tentang bagaimana menggunakan laser tersebut untuk mengontrol atau merubah sifat dari IC.

Eksperimen pertama mentargetkan SRAM. Struktur dari SRAM standar dengan enam transistor ditunjukkan pada gambar 7. Dua pasang dari kanal transistor p- dan n- membuat rangkaian flip-flop, dimana dua kanal transistor n- lainnya digunakan untuk membaca *state* dari transistor dan menulis nilai baru kedalamnya. *Layout* dari sel ditunjukkan pada gambar 8. Transistor M1 dan M3 membuat *inverter* CMOS; berbarengan dengan pasangan sejenis yang lainnya, mereka menciptakan rangkaian flip-flop yang dikontrol oleh transistor M3 dan M6.



Gambar 7: Struktur sel SRAM dengan enam transistor



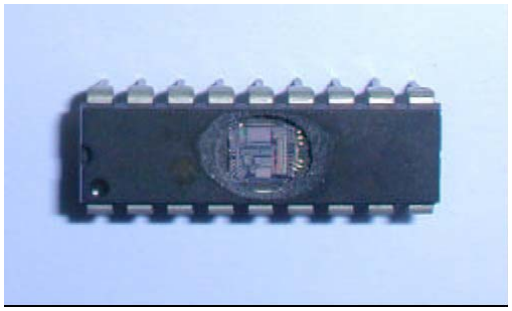
Gambar 8: *Layout* sel SRAM dengan enam transistor

Jika transistor M1 dapat dibuka untuk waktu yang sangat singkat dengan rangsangan dari eksternal, maka itu dapat membuat flip-flop berubah *state*. Dengan membuka transistor M4, *state* dari sel akan berubah menjadi *state* yang sebaliknya. Kesulitan utama yang harus diantisipasi adalah memfokuskan radiasi ion berkurang menjadi beberapa μm^2 dan memilih intensitas yang sesuai.

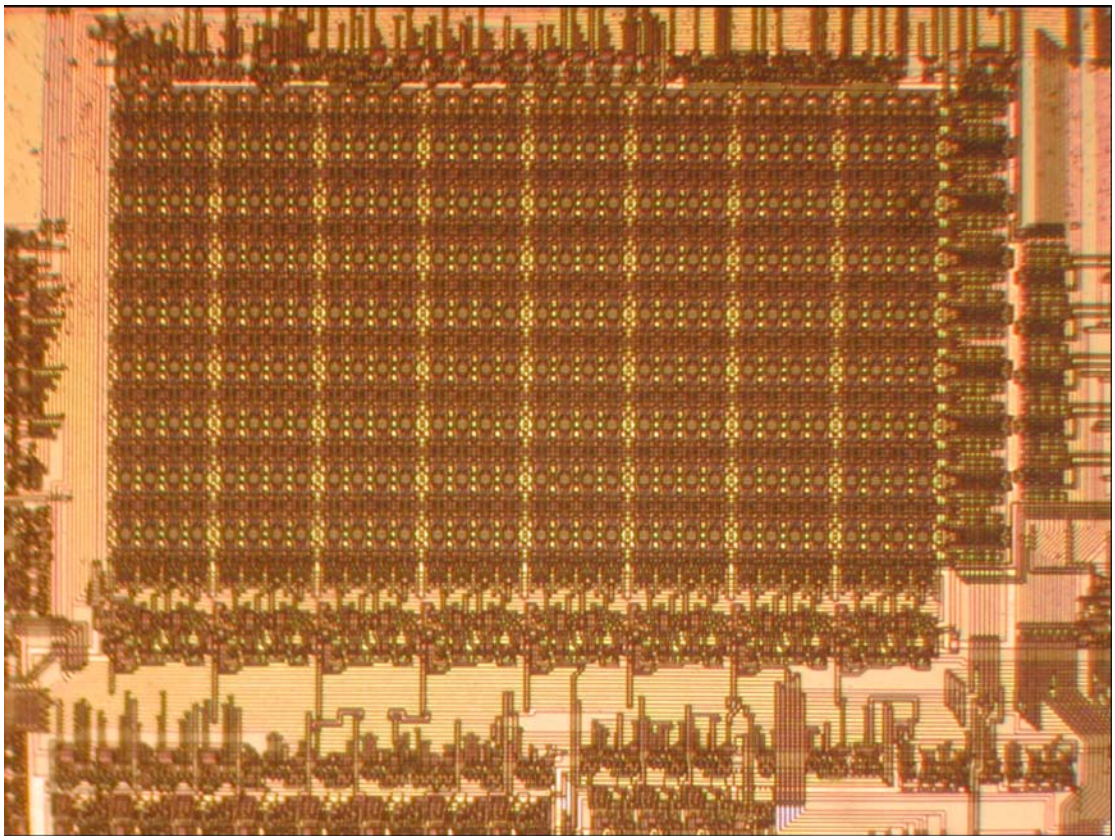
Gambar 9: Mikrokontroler PIC16F84

2.2.1 Metode serangan

Sebagai contoh untuk serangan, dipilih mikrokontroler umum, PIC16F84, yang memiliki memori SRAM 68 byte pada chipnya yang bisa dilihat pada gambar 9. Prosedur standar pembongkaran diterapkan untuk chip ini dan hasil operasinya dapat dilihat pada gambar 10. Larik memori SRAM terletak ditengah dari dibawah inti chip. Area ini ditunjukkan dalam perbesaran 80 kali di gambar 11.



Gambar 10: Chip PIC16F84 yang sudah dibongkar



Gambar 11: Larik memori SRAM dengan perbesaran 80x

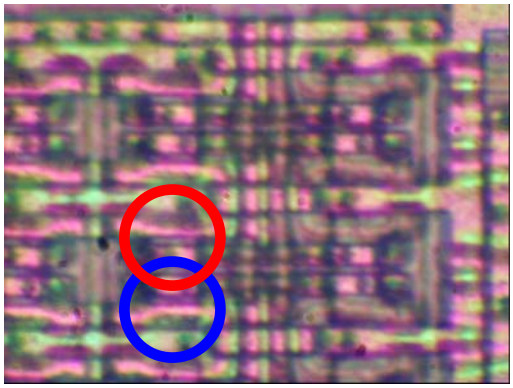
Mikrokontroler tersebut diprogram untuk meng-*upload* dan men-*download* memorinya sendiri. Dengan mengisi seluruh memori dengan nilai konstan, memaparkannya ke cahaya, dan men-*download* hasilnya, dapat diamati sel mana saja yang merubah *state* mereka.

.

State akhir dari sel tergantung kepada area yang terpapar oleh cahaya. Hal ini dapat membuktikan bahwa memungkinkan untuk mengganti isi dari SRAM dengan menggunakan serangan *semi invasive* yang berbiaya rendah.

Dari percobaan diatas ditemukan bahwa penyerang dapat mengubah bit-bit individual dari larik SRAM. Larik tersebut dalam perbesaran maksimal dapat dilihat pada gambar 12. Pemfokusan cahaya ditujukan ke area yang ditunjuk lingkaran merah membuat sel tersebut merubah *state*nya dari 1 menjadi 0, dan tidak berubah ketika *state*nya sudah 0. Dengan memfokuskan cahaya ke area yang ditunjuk lingkaran biru, sel tersebut merubah *state*nya dari 0 menjadi 1 dan tidak berubah ketika *state*nya sudah 1.

Dapat dilihat dari gambar 11 bahwa larik SRAM dibagi menjadi delapan blok yang sama. Dengan memaparkan sel di blok yang berbeda, ditemukan bahwa setiap blok berkorespondensi dengan satu bit dari informasi. Hasil dari operasi ini ditunjukkan oleh gambar 13.



Gambar 12: Larik memori SRAM dengan perbesaran 1500x

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Gambar 13: Alokasi bit data dalam larik memori SRAM

Peta dari *address* yang berkorespondensi langsung dengan lokasi fisik dari setiap sel memori pun dibangun dengan memaparkan setiap sel dengan cahaya. Hasilnya dipresentasikan pada gambar 14, dengan bagian kiri berkorespondensi dengan bagian bawah dari blok. Dapat dilihat bahwa *address-address* tersebut tidaklah sekuensial, namun dibagi menjadi tiga grup.

0x30	0x34	0x38	0x3C	0x40	0x44	0x48	0x4C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x0C
0x31	0x35	0x39	0x3D	0x41	0x45	0x49	0x4D	0x11	0x15	0x19	0x1D	0x21	0x25	0x29	0x2D	0x0D
0x32	0x36	0x3A	0x3E	0x42	0x46	0x4A	0x4E	0x12	0x16	0x1A	0x1E	0x22	0x26	0x2A	0x2E	0x0E
0x33	0x37	0x3B	0x3F	0x43	0x47	0x4B	0x4F	0x13	0x17	0x1B	0x1F	0x23	0x27	0x2B	0x2F	0x0F

Gambar 14: Alokasi *address* pada setiap blok bit dari larik memori SRAM

Hal ini menunjukkan bahwa betapa serangan *semi invasive* sederhana dapat digunakan untuk *reverse engineering* sebuah peta *address* memori.

3. Anti serangan

3.1 Anti serangan analisis daya

Pengurangan daya pada sirkuit CMOS static di dominasi oleh aktivitas *switching*. Sebagai hasilnya daya yang berkurang oleh sebuah sirkuit sangat tergantung dari aktivitas

switching yang dihasilkan dari perubahan input data. Dalam kasus sederhana dari bus, aktivitas diawasi seiiring dengan perubahan *state* dari berat Hamming.

Serangan analisis daya dapat dicegah dengan membuang atau menyembunyikan kebocoran informasi oleh aktivitas *switching*. Informasi dapat dihilangkan atau setidaknya dikurangi dengan usah untuk mengurangi daya konstan, dengan tidak memerhatikan input data. Pada level transistor perpindahan dari logika statik ke bentuk logika konstan dapat memberikan reduksi besar pada daya yang bergantung pada data, walaupun performansi dapat berkurang sangat drastis.

Informasi dapat disembunyikan dengan mengacak urutan dari eksekusi atau mengacak sumber daya perangkat keras tertentu yang digunakan dalam operasi. Operasi acak tambahan dapat pula dilakukan untuk menyembunyikan emisi daya, walaupun harus dilakukan secara hati-hati untuk menghindari kemungkinan penjadwalan ulang atau operasi *dummy* terdeteksi dan dibalik ketika *post-processing* sinyal.

Pada level lojik, transformasi dapat diaplikasikan ke komputasi untuk menghilangkan semua korelasi antara input data dengan pengurangan daya. Teknik-teknik ini menggunakan sumber angka acak untuk meng-*encode* setiap bit input dengan tujuan untuk menyembunyikan emisi yang bergantung pada data. Setiap set dari bit yang telah di-*encode* di rekombinasi setelah komputasi untuk menghasilkan output biner yang benar. Walaupun teknik seperti ini secara potensial menawarkan derajat keamanan yang tinggi, teknik-teknik seperti ini seringkali membutuhkan *overhead* area yang signifikan dan tidak memiliki properti untuk mengecek diri sendiri secara alami.

Teknik yang dipresentasikan disini bergantung pada penyeimbangan konsumsi daya pada level gerbang. Setiap bit atau grup dari bit di-*encode* menggunakan *encoding 1-of-n* atau *one-hot*. Walaupun secara umum *encoding* apapun yang bertipe *m-of-n* dapat digunakan, *1-of-n* seringkali merupakan yang paling pas unutk sebuah implementasi asinkron. *Encoding* ini menjamin bahwa nilai dari transisi sinyal akan selalu sama tidak peduli apapun input datanya, sehingga dapat menyeimbangkan konsumsi daya. Kasus *1-of-2* atau *dual-rail* digambarkan pada gambar 15. *Encoding* umum lainnya yang biasa digunakan adalah *1-of-4* yang merepresentasikan grup dari 2 bit. Sebagai tambahan kepada penyeimbangan konsumsi daya, encoding mengijinkan penyelesaian dari sebuah operasi untuk dideteksi, membuat implementasi *clock-free* yang *robust*. Desain dari lojik seperti ini biasa disederhanakan jika sebuah *return-to-zero* (RTZ) digunakan setelah setiap komputasi untuk menyediakan sebuah *spacer* diantara *item* data yang berbeda. Kombinasi dari fase RTZ ini dan desain yang cermat dari lojik itu sendiri berarti dapat menjamin operasi yang benar tanpa peduli dengan delai di gerbang.

Sebuah blok yang umum untuk sirkuit asinkron adalah Muller *C-element*. Gerbang beroperasi dengan menyetting outputnya sendiri ketika kedua input input pada *state High*, tetapi hanya melakukan reset jika kedua input berada pada *state Low*. Elemennya lalu menyediakan konjungsi baik untuk transisi naik maupun turun. Hal ini sangat krusial ketika menggunakan *encoding* yang tidak sensitif terhadap delai untuk mencegah RTZ dideteksi terlalu cepat. Simbol *C-element* ditunjukkan oleh gambar 16, dimana terlihat simbol gerbang-AND dengan tambahan karakter C ditengahnya.

Sayangnya, *encoding 1-of-n* tidak cukup untuk menjamin sebuah daya yang tidak bergantung pada data. Satu masalah potensial adalah isi dari gerbang mungkin berbeda karena ada perbedaan ketika *routing*. Desain dari setiap gerbang *dual-rail* juga harus memastikan isi pin input yang seimbang dan penggunaan daya yang seimbang. Dalam prakteknya ini mudah untuk dilakukan pada level gerbang dan dapat diperkuat dengan penggabungan sel yang cermat ketika penempatan.

Perhatian terakhir adalah ketika setiap unit fungsional individual tidak lagi menghasilkan tanda daya yang bergantung pada data, aktivitas pada level lebih tinggi masing mungkin bergantung pada data. Solusi paling mungkin adalah dengan menjaga level tinggi dari aktivitas sistem setiap saat, bahkan ketika operasi super harus dilakukan untuk menjaga agar lojik tetap sibuk. Dalam sistem yang sinkron hal ini trivial karena *clock* menyediakan menyediakan indikasi konstan kapan operasi baru harus dilakukan (tidak peduli dengan dependensi data di sirkuit). Kontras dengan itu, sebuah sistem yang asinkron adalah *data-driven* dimana setiap unit fungsional tidak mempunyai konsep dari putaran langsam. Dalam prakteknya, problem ini sering ditangani pada level arsitektural. Observasi lebih lanjut bahwa memaksakan

level tinggi dari aktivitas sirkuit dapat menuju ke sebuah mode *pseudo-synchronous* dari perilaku bahkan dalam implementasi asinkron yang tidak sensitif terhadap delai.

3.2 Anti serangan injeksi kesalahan

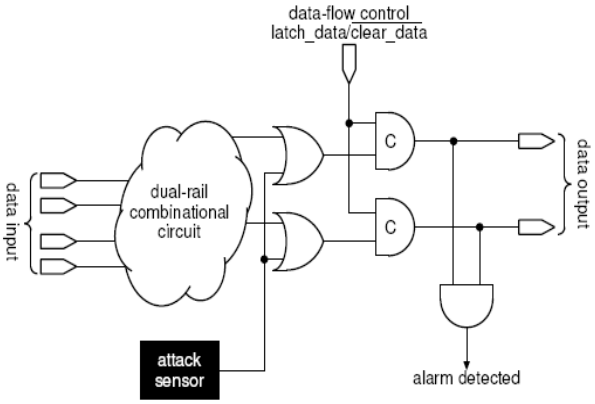
Kemungkinan untuk melakukan serangan *clock glitch* bisa dibuang jauh-jauh dalam kasus sirkuit asinkron. Dimana *clock* dibutuhkan, sebagai contoh dalam antar muka *clocked serial smart card*, sangat mudah untuk mengatur bahwa korupsi data tidak harus berakibat pada bocornya data yang sensitif.

Power glitch atau teknik induksi kesalahan terlokalisasi diselesaikan dengan mengeksploitasi property untuk pengecekan diri sendiri dari sirkuit ter-*encode m-of-n*. Skema *coding* seperti ini dikenal sebagai kode yang tidak dapat diubah, dan memegang properti bahwa setiap kata kode valid dengan panjang *n* memiliki tepat *m* 1's. Kesalahan apapun yang mengacu kepada perubahan dari berat Hamming dari kode pun nantinya akan dengan sangat mudah dideteksi. Kesalahan asimetrik atau bit tunggal apapun masuk kedalam kategori ini. Untuk itu, untuk sirkuit ter-*encode dual-rail* kode invalid (1,1) sekarang merepresentasikan kesalahan atau *state* alarm (lihat gambar 15). Sebuah kesalahan mengacu ke kode (0,0) akan mencegah penyelesaian dan mungkin dideteksi oleh sebuah *timeout*. Untuk menjamin bahwa kita mengamati kata kode ilegal apapun pada output dari sirkuit, kita harus hanya menjamin bahwa setiap gerbang menghasilkan *encoding* output ilegal sebagai respons dari *encoding* input ilegal. Walaupun banyak kode pengecekan diri sendiri dan arsitektur ditawarkan, belum jelas kalau salah satu dari mereka menawarkan kemungkinan untuk menyeimbangkan pengurangan daya untuk data input yang berbeda secara mudah.

.

A1	A0	meaning
0	0	clear
0	1	logical 0
1	0	logical 1
1	1	alarm

Gambar 15: *Encoding dual-rail* dengan definisi sinyal alarm



Gambar 16: Insersi dan deteksi alarm

Idealnya, semua kesalahan yang dapat dideteksi sekarang menghasilkan satu dari dua perilaku seperti dibawah ini pada kasus sirkuit asinkron:

1. Penurunan dalam jumlah 1's akan mencegah deteksi penyelesaian, memaksa sirkuit untuk *hang* dalam batas waktu yang tidak ditentukan. Sebagai tambahan, sebuah *timeout* dapat diasosiasikan dengan setiap sirkuit untuk menyebabkan kode alarm untuk disebarkan secara cepat.

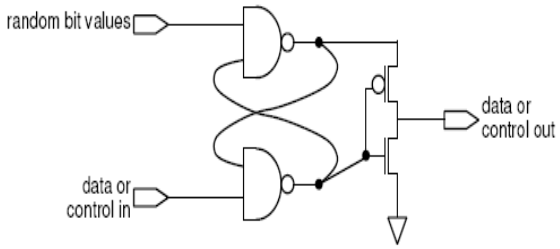
2. Kenaikan dalam jumlah 1's akan menyebabkan sebuah kesalahan dideteksi pada output. Sinyal alarm kemudian dapat didistribusikan ke sirkuit yang lain untuk memaksa sebuah *system-wide deadlock* secara cepat.

Banyak sensor kepalsuan yang berbeda yang ada yang mampu mendeteksi perubahan pada suplai tegangan atau usaha ketika pembongkaran (contoh temperatur, level cahaya, kapasitas/konduktivitas paket, dan sensor metal). Semuanya memancarkan sinyal ketika kondisi yang diinginkan dilanggar. *State* alarm dapat diaktifkan oleh ORing, sebuah sinyal kontrol alarm dari sensor kepalsuan dengan jalur data *dual-rail*. *State* alarm dideteksi dengan sebuah pohon AND-OR pada data, dan dapat diinjeksikan kedalam bagian lain dari sirkuit untuk mempercepat *deadlock*. Jumlah persirkuitan secara substansial dikurangi ketika diketahui bahwa menyetting alarm untuk satu bit *logical* akan berdampak pada penyebaran alarm ke bit-bit lainnya.

3.3 Menghamburkan *timing* yang bergantung pada data

Serangan *timing* pada prosesor dengan *clock* dilakukan dengan menghitung putaran antara *event* yang diketahui untuk menurunkan informasi rahasia. Sebuah prosesor asinkron lebih mudah terpengaruh untuk membocorkan data dalam domain waktu. Sebagai contoh, sebuah *adder* dapat didesain dengan cermat untuk mengkonsumsi energi secara konstan per operasi, tetapi durasi dari operasi tersebut tergantung pada penyebaran pembawanya. Dalam intansi ini, *adder* harus didesain untuk selalu mencapai performansi yang *worst case*. Hal ini tidak berarti bahwa *adder* harus punya performansi yang buruk. Sebagai contoh, dapat didesain sebuah *carry* yang independen dari sisi kecepatan yang memilih *adder* yang beroperasi cepat dan dengan properti *timing* yang konstan.

Variasi dalam jalur dan beban gerbang juga dapat mengacu kepada *timing* yang bergantung kepada data. Beban jalur dapat diminimalisasi dengan mengaplikasikan batasan yang pas di tempat dan rutenya. Ketika beban jalur dibuat insiginifikan maka desain harus dipartisi menjadi sejumlah blok kecil. Penyeimbangan beban jalur hanya dapat dipertimbangkan pada batasan antar muka.



Gambar 17: Insersi delai acak

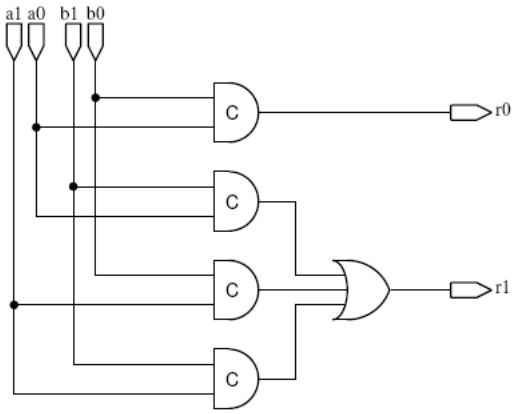
Ketika ada perhatian lebih untuk *timing* yang bergantung pada data, delai acak bisa dimasukkan dengan menggunakan sirkuit pada gambar 17 yang berbasis pada Seitz *arbiter*.

Timing perangkat lunak yang bergantung pada data juga merupakan sebuah isu tersendiri. Sudah kewajiban bahwa aplikasi perangkat lunak untuk *smart card* bekerja dalam kolaborasi dengan perangkat keras. Satu pendekatan adalah untuk memastikan bahwa pekerjaan yang sama dilakukan tanpa mempedulikan datanya. Akan tetapi hal ini menjadi sulit dan berakibat pada kode yang lambat. Sebuah pendekatan alternatif lain yaitu dengan memperkenalkan non-deterministik pada level perangkat lunak (contoh pada gambar 18). Sangat penting untuk dicatat bahwa bentuk non-deterministik ini hanya menambah kompleksitas dimana informasi *side-channel*nya susah untuk dianalisa pertama kali.

```
if(random_condition) {
            f(); g();
} else {
            g(); f();
}
```

where `f()` and `g()` are independent functions

Gambar 18: Menambahkan non-deterministik ke perangkat lunak

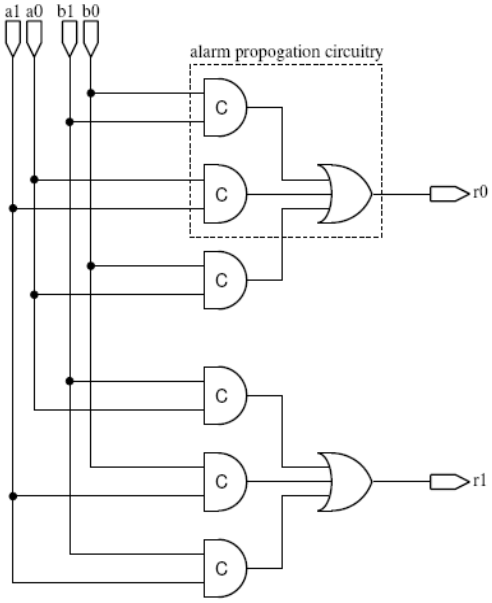


Gambar 19: Gerbang OR *dual-rail* SI

3.4 Contoh pekerjaan

Untuk mengilustrasikan teknik yang didiskusikan pada bagian-bagian sebelumnya, disini dijabarkan desain dari gerbang OR *dual-rail* dan sebuah *dual-rail full-adder*. Untuk memperbaiki keamanan, *timing* yang independen terhadap data dan konsumsi daya dibentuk. Sebuah gerbang OR *dual-rail* sederhana ditunjukkan pada gambar 19. Jika sebuah sinyal alarm ada pada input (a1; a0) atau (b1; b0) maka output r_1 akan selalu diset (ketika apapun selain sinyal *null* ada pada input yang lain). Akan tetapi, sinyal r_0 tidak akan diset jika hanya satu dari input *dual-rail* dalam posisi alarm. Gerbang OR juga mempunyai waktu yang bergantung pada data untuk menghasilkan output. Kedua isu ini

dipecahkan pada gambar 20 dengan menambahkan lojik tambahan untuk mendeteksi *state* alarm, memaksa r_0 pada *state High* dan pada saat yang bersamaan membuat delai data menjadi independen. Elemen C dalam sirkuit penyebaran alarm dapat diganti dengan gerbang AND jika pengganti percaya bahwa sinyal alarm yang datang stabil. Akan tetapi, dibawah serangan *glitch* kita tidak dapat percaya pada keadaan ini sehingga aspek penyimpanan dari elemen C diperlukan. Elemen ini juga berfungsi untuk menyeimbangkan beban pada setiap pin input.



Gambar 20: Gerbang OR *dual-rail* SI dengan penyebaran alarm

Contoh gerbang OR mengilustrasikan bagaimana delai dalam jalur dapat diseimbangkan. Akan tetapi, hal ini tidak selalu dibutuhkan. Sebagai contoh, coba perhatikan sebuah *dual-rail full-adder*. Input-input *dual-rail* adalah (a1; a0), (b1; b0), (cin1; cin0) dan output-outputnya adalah (sum1; sum0) dan (cout1; cout0). Karena kita membutuhkan beberapa fungsi dari input, maka cukup sepadan untuk melebarkan input menjadi *1-of-8 code vis*:

$i000 = C(a_0; b_0; cin_0)$
$i001 = C(a_0; b_0; cin_1)$
$i010 = C(a_0; b_1; cin_0)$
...
$i110 = C(a_1; b_1; cin_0)$
$i111 = C(a_1; b_1; cin_1)$

Dimana *C*() merepresentasikan sebuah fungsi elemen C.

Sekarang outputnya dapat didefinisikan sebagai OR-*plane* dari kode *1-of-8*:

$sum_0 = i000 + i011 + i101 + i110$
$sum_1 = i001 + i010 + i100 + i111$
$cou_{t0} = i000 + i001 + i010 + i100$
$cout_1 = i011 + i101 + i110 + i111$

Pendekatan ini berakibat pada *timing* yang independen terhadap data tetapi sinyal alarm tidak dijamin untuk menyebar. Sinyal alarm bisa diidentifikasi dan digabungkan dengan *term sum* dan *carry*:

$alarm = RS(reset; a_0{:}a_1 + b_0{:}b_1 + cin_0{:}cin_1)$
$sum_0 = i000 + i011 + i101 + i110 + alarm$
$sum_1 = i001 + i010 + i100 + i111 + alarm$
$cout_0 = i000 + i001 + i010 + i100 + alarm$
$cout_1 = i011 + i101 + i110 + i111 + alarm$

Dimana *RS*() adalah sebuah RS flip-flop.

RS flip-flop digunakan untuk membuat alarm dalam *state High* yang akan menyebabkan *deadlock*. Banyak fungsi *dual-rail* yang bisa diimplementasikan dalam bentuk SI ini: Pertama, perlebar input-inputnya menjadi sebuah kode *1-of-n* lalu bentuk sebuah OR *plane* untuk menentukan output-outputnya.

4. Simpulan

Ada dua tipe utama dari serangan umum terhadap *smart card,* yaitu serangan *side-channel* dan serangan injeksi kesalahan optikal. Serangan *side-channel* dapat dilakukan dengan berbagai cara baik dengan media perangkat keras maupun perangkat lunak. Sedangkan serangan injeksi kesalahan optikal adalah serangan yang menggunakan perangkat keras karena sifatnya yang *semi-invasive*. Kedua jenis serangan secara umum ini pada dasarnya sudah bisa ditangani secara umum pula oleh semua jenis *smart card* yang diproduksi saat ini.

Testing penyerangan telah membuka kekuatan dan kelemahan pada chip yang dites. Lojik *dual-rail* yang seimbang telah didemonstrasikan dan secara substansial lebih *robust* melawan analisis daya diferensial. Perkembangan dari serangan *optical probing* telah didemonstrasikan efektif untuk melawan chip tes yang tergolong masih modern, akan tetapi, disaat ada *redundancy*, justru serangan optikal dapat dipatahkan. Untuk memperbaiki keamanan lebih jauh harus dilihat anti serangan seperti: level tinggi dari kotak bertahan, sensor perubahan optikal, dan lain

sebagainya. Anti-anti serangan ini sudah tersedia pada *smart card* tipe *high-end* saat ini namun masih membutuhkan perbaikan seiiring dengan teknologi serangan yang juga semakin membaik.

Cost untuk menggunakan lojik *dual-rail* adalah sebuah area penalty yang signifikan. Prosesor standar membutuhkan hampir tiga kali area dari yang original bertipe sinkron, walaupun ini dapat dikurangi secara signifikan dengan menambahkan Muller C-*elements* ke *library* sel standar. Sistem secara keseluruhan juga harus dilihat dan bertanya bagaiamana anti serangan perangkat lunak dapat disederhanakan jika prosesor yang lebih aman digunakan. Anti serangan perangkat lunak dapat meningkatkan ukuran kode dengan faktor pengali 2 sampai 3, yang mempunyai dampak substansial terhadap ukuran memori ROM/Flash yang sebaliknya mempunyai dampak yang sudah diketahui terhadap area dari chip.

DAFTAR PUSTAKA

[1] Moore, Simon, Ross Anderson, Paul Cunningham, Robert Mullins, George Taylor. *Improving Smart Card Security using Self-timed Circuits.* University of Cambridge.

[2] Munir, Rinaldi. *Diktat Kuliah IF5054 Kriptografi*. Institut Teknologi Bandung. 2006

[3] Rantos, Konstantinos, Konstantinos Markantonakis. *An Asymmetric Cryptography Secure Channel Protocol For Smart Cards*. University of London. 2004

[4] Skorobogatov, Sergei, Ross Anderson. *Optical Fault Induction Attacks.* University of Cambridge. 2002

[5] Verschuren, Jan. *Smart Card Analysis*. TNO Evaluation Centre, Netherlands.

[6] Walter, Colin D. *Longer Keys May Facilitate Side Channel Attacks*. Comodo Research Lab. 2004

[7] http://www.cl.cam.uk/~rja14/ diakses tanggal 12 Desember 2006

[8] http://www.comodogroup.com/research/crypto/publications.html diakses tanggal 14 Desember 2006

[9] http://home.hkstar.com/~Ealanchan/papers/SmartCardSecurity/ diakses tanggal 14 Desember 2006

[10] http://www.ics.ee.ic.ac.uk/surp98/report/ylc3/report2.html diakses tanggal 14 Desember 2006