# SISTEM KRIPTOGRAFI PAILLIER

Andoko Gunawan – NIM : 13503083

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail : if13083@students.if.itb.ac.id

## 1. Abstrak

Sistem kriptografi Paillier adalah sebuah sistem yang berbasis algoritma asimetris probabilistik. Algoritma enkripsi yang digunakan adalah sebuah algoritma kriptografi kunci publik. Sistem ini ditemukan oleh Pascal Paillier pada tahun 1999.

Sistem kriptografi Paillier dibuat berdasarkan pemikiran bahwa untuk menghitung kelas residu yang ke –n, perhitungan yang dibutuhkan sangat sulit. Hal ini dikenal sebagai asumsi *Composite Residuosity* (CR).

Sistem kriptografi Paillier ini memiliki properti – properti seperti *random-self-reducibility, additive homomorphic property,* dan *self – blinding*. Properti – properti inilah yang menyebabkan kriptosistem Paillier ini dapat digunakan untuk berbagai keperluan, seperti pemungutan suara elektronik dan uang elektronik.

Yang akan dibahas pada makalah ini adalah dasar dari skema algoritma Paillier, cara enkripsi, cara dekripsi, optimisasi performansi, dan aplikasinya pada kehidupan dunia nyata.

Kata kunci : Sistem kriptografi, Paillier, enkripsi, dekripsi, Composite Residuosity

## 2. Pendahuluan

Sejak ditemukannya kriptografi kunci publik oleh Diffie dan Hellman, sangat sedikit skema kriptografi kunci asimetris yang telah ditemukan meskipun usaha untuk mencari algoritma yang bersangkutan sangat banyak dilakukan.

Sekarang, skema – skema kriptografi kunci asimetris yang ada dapat dibagi dalam dua aliran yang besar, menurut dasar aritmatik dari algoritma yang digunakan pada skema – skema tersebut.

Yang pertama adalah yang skema – skema yang berdasarkan algoritma dengan kekuatan pada kesukaran aritmatik dalam memfaktorkan integer. Skema –skema yang termasuk dalam aliran ini adalah RSA, skema Rabin – William, LUC, skema Dickson, dan versi ellipsis dari RSA, seperti KMOV.

Teknik ini menggabungkan antara ekstraksi faktor – faktor dari polinom dari *field* yang membutuhkan waktu komputasi polinomial, dengan sulitnya untuk memfaktorkan sebuah nomor yang sangat besar.

Dari skema – skema yang masuk dalam aliran ini, hanya skema Rabin – William yang memiliki tingkat kesukaran yang sebanding dengan persoalan pemfaktoran sejauh ini. Dan dari aliran ini hanya ada dua skema yang akhirnya digunakan untuk keperluan komersial, yakni RSA dan Rabin – William.

Yang kedua adalah skema – skema yang berdasarkan algoritma dengan kekuatan pada kesukaran aritmatik dalam memecahkan masalah logaritmik diskrit. Skema – skema yang termasuk dalam aliran ini adalah skema El – Gamal, DSA, McCurley, dll.

Teknik ini menggabungkan properti homomorfisme dari perpangkatan moduler dan sulitnya mengekstraks logaritma diskrit dari grup – grup yang terhingga banyaknya. Tingkat kesamaan dengan masalah

komputasi yang menjadi primitif dari algoritma yang digunakan untuk skema – skema diatas masih menjadi pertanyaan, kecuali sudah dinyatakan secara eksplisit.

Aliran – aliran selain dua aliran diatas umumnya masih memiliki masalah dengan efisiensi, kelemahan dalam keamanan, atau permasalahan dalam *public scrunity*. Beberapa skema ini adalah skema – skema kriptografi yang termasuk dalam aliran non mainstream adalah :

McEliece, *cryptosystem* yang berbasis pada kode pengoreksi kesalahan, skema Ajtai-Dwork yang berbasis *lattice problem* (skema ini sudah berhasil di kriptanalis oleh Nguyen dan Stern), sistem Knapsak yang bertipe penjumlahan dan perkalian yang meliputi skema Merkle-Hellman, Chor-Rivest (dipecahkan oleh Vaudenay dan Naccache – Stern), dan yang terakhir adalah kriptosistem Matsumoto-Imai dan Goubin-Patarin yang telah berhasil di kriptanalisis.

Namun, sekarang ada kemajuan yang sangat pesat dalam teknik *third class of trapdoor*. Sebelumnya dikenali sebagai trapdoors pada discrete log, teknik ini sebenarnya muncul dari kondisi umum dari kelas residu berderajat tinggi (*high degree residuosity classes*).

Selain dari *high degree residuosity classes* ini, masih ada beberapa teori lain yang mendasari kriptosistem ini, seperti *one-wayness*, *trapdoor permutations*, dan lain – lain. Hal ini akan dibahas pada bagian dasar teori dibawah.

## 3. Penekanan Keamanan

Dalam formalisasi kriteria keamanan yang lain yang mana sebuah skema enkripsi harus diverifikasi tidak hanya pada property *one – wayness*, Goldwasser dan Micali memperkenalkan sebutan atau istilah keamanan. Properti ini, disebut juga *indistinguishability* (ke-tidak-dapat dibedakan) *of encryptions* (atau disingkat pula sebagai IND), mewujudkan sebuah gagasan bahwa seorang penceroboh atau musuh seharusnya tidak bisa mendapatkan informasi apapun mengenai sebuah plainteks, kecuali panjangnya saja, dari enkripsi plainteks tersebut.

Properti dari *non-malleability* (NM), yang secara terpisah diajukan oleh Doley, Dwork, dan Naor, menduga bahwa, untuk setiap enkripsi plainteks x, si penyerang tidak dapat memproduksi enkripsi dari plainteks $x_0$ yang bersangkutan. Disini, daripada mencoba mendapat informasi yang penting mengenai x, si penyerang akan mencoba mengoutput enkripsi dari $x_0$. Dua property ini saling berhubungan dimana *non-malleability* menjamin adanya kemanan semantik untuk setiap model serangan.

Di sisi lain, ada beberapa tipe serangan, atau model serangan.

Dalam sebuah *chosen-plaintext attack* (CPA), si penyerang memiliki akses pada sebuah *encryption oracle*, yang berarti akses pada enkripsi seluruh plainteks yang dia miliki. Pada sebuah setting pada kriptografi kunci public, hal ini tidak dapat dihindarkan. Naor dan Yung mencoba *non-adaptive chosen-ciphertext attacks* (CCA1), atau yang dikenal sebagai serangan *lunch / midnigth*, dimana si penyerang memiliki akses ke *decryption oracle* sebelum diberikan cipherteks yang harus ia pecahkan.

Yang terakhir, Racko dan Simon mendefinisikan sebuah *adaptive chosen-ciphertext attacks* (CCA2) sebagai sebuah scenario dimana si penyerang menggunakan *decryption oracle* sebelum dan sesudah percobaan serangan. Satu – satunya batasan disini adalah bahwa si penyerang tidak boleh mencoba *oracle* dengan cipherteks yang ingin ia pecahkan.

Serangan tersebut adalah jenis serangan yang paling kuat yang diketahui sampai saat ini.

Beberapa level keamanan didefinisikan dengan memasangkan tiap objektif keamanan (IND atau NM) dengan model serangan (CPA, CCA1 atau CCA2), dua karakteristik ini biasanya dianggap terpisah. Yang menarik, telah dibuktikan bahwa IND-CCA2 dan NM-CCA2 adalah *notions* yang ekivalen secara ketat.

Lebih jauh lagi, Bellare dan Rogaway telah mengajukan sebuah konsep tentang kesadaran plainteks, dimana si penyerang

mencoba untuk memproduksi sebuah cipherteks yang valid tanpa mengetahui plainteks yang bersangkutan. Penekanan ini hanya didefinisikan pada Random Oracle Model.

Random Oracle Model diajukan oleh Bellare dan Rogaway untuk menyediakan bukti – bukti heuristic mengenai keamanan yang sangat menyakinkan. Pada model ini, fungsi hash dianggap ideal, karena sifatnya yang acak. Dari sudut pandang sekuriti, hal ini mempengaruhi ketiga model serangan dengan memberikan penyerang akses tambahan ke *random oracles* dari suatu skema.

## 4. Usaha – usaha Lainnya

Dasar dari skema enkripsi El Gamal, dimana *one-wayness* berhubungan dengan permasalahan Diffie – Hellman (DH) problem, terbukti aman (aman dari sudut pandang IND-CPA) oleh Tsiounis dan Yung dibawah asumsi *Decision Diffie-Hellman* (D-DH). Namun, seperti halnya RSA, skema original dari El Gamal tetap tidak aman apabila dihadapkan pada serangan aktif. Kemudian skemanya diubah sedikit sehingga terbukti aman dari sudut pandang IND-CCA2 pada *Random Oracle Model* dibawah asumsi D-DH dan dibawah asumsi yang standar.

Secara terpisah, Shoup dan Gennaro mengajukan perubahan skema yang lain untuk IND-CCA3 pada *Random Oracle Model* namun hanya dibawah asumsi D-DH saja. Pada tahun yang sama, Cramer dan Shoup juga menciptakan sebuah kriptosistem yang berbasis pada El Gamal, skema pertama yang terbukti praktis dan aman (aman dari sudut pandang IND-CCA2) pada model standard, dimana asumsi D-DH berlaku.

Selain itu, ada pula kajian mengenai asumsi tentang *intractability* yang lain. Point-Cheval mengajukan DRSA, sebuah skema enkripsi yang berdasarkan pada permasalahan *Dependent-RSA*, dan yang menyediakan varian lain yang efisien dan aman dari sudut pandan IND-CCA2, pada *Random Oracle Model* dibawah hipotesis bahwa versi D-RSA adalah *intractable.*

Naccache dan Stern, dan secara terpisah Okamoto dan Uchiyama, menginvestigasi beberapa pendekatan yang berbeda yang berdasar pada *high degree residues*. Properti *one-wayness* dai skema mereka dipastikan dengan asumsi *Prime-residuosity* (akibat dari sukarnya membedakan *prime-degree residues*). Akhirnya, yang akan dibahas disini, Paillier mengajukan sebuah skema enkripsi yang berbasis *composite-degree residues* dimana keamanan semantic juga bergantung pada asumsi yang sama.

Bellare dan Rogaway mengajukan OAEP, sebuah perlakuan spesifik berbasis hash yang dapat diaplikasikan untuk setiap *one-way trapdoor permutation* untuk membuatnya aman dari sudut pandang IND-CCA2. Memiliki dasar *Random Oracle Model*, keamanan sekuriti dari OAEP diakui secara luas, dan kemudian mendasari PKCS #1 V2.0 yang berbasis RSA. Akhir – akhir ini , Fujisaki dan Okamoto menemukan sebuah metode konversi generic yang mentransformasi tiap skema enkripsi yang secara semantic aman ke sebuah skema yang aman dari sudut pandang IND-CCA2 pada *Random Oracle Model.* Konversi yang diajukan oleh Pascal Paillier dan David Pointcheval sifatnya *low-cost* untuk enkripsi (hanya memerlukan sebuah tambahan hash, namun berat untuk proses dekripsinya.

## 5. Dasar Teori

Kriptosistem Paillier berbasis pada on *composite residuosity classes*, bukan pada *prime residuosity* yang selama ini sudah lebih banyak dikenal. Contohnya adalah penggunaan himpunan derajat pada sebuah bilangan n = pq yang sangat sulit untuk difaktorkan, dimana p dan q adalah dua buah bilangan prima. Dapat dilihat bahwa teknik *trapdoor* ini dapat menyediakan sebuah blok pembangun kriptografik untuk mendapat sebuah kriptosistem yang berbasis kunci publik.

Pada dasar teori ini akan dibahas pula kerangka kerja teori bilangan dan menginvestigasi pada konteks ini sebuah permasalahan komputasi dimana asumsi yang utama akan ada pada permasalahan *intractability*. Kemudian, akan dibahas pula tiga buah skema enkripsi yang bersifat

*homomorphic* yang berbasis pada permasalahan ini, termasuk pula sebuah teknik permutasi *trapdoor* yang baru. Skema probabilistic akan terbukti aman sejalan dengan asumsi mengenai *intractability* permasalahan yang bersangkutan. Seluruh reduksi polynomial akan dijaga agar tetap sederhana dan megikuti standard.

Notasi : pertama – tama akan diset n = pq dimana p dan q adalah dua buah bilangan prima yang sangat besar jumlahnya $\Phi$(n) = (p-1)(q-1) dan $\lambda$(n) = lcm(p-1, q-1) dalam kasus ini.

Bahwa $|\mathbb{Z}^*_{n^2}| = \phi(n^2) = n\phi(n)$ dan untuk setiap $w \in \mathbb{Z}^*_{n^2}$;

$$\begin{cases} w^{\lambda} = 1 \bmod n \\ w^{n\lambda} = 1 \bmod n^2 \end{cases}$$

yang sesuai dengan teorema Carmichael. Permasalahan mengekstrak akar dari modulus n yang ke e (dianggap *intractable* secara konvensi), dimana n = pq dituliskan dalam RSA[n, e]. dan n = pq tidak diketahui factor – faktornya. Relasi $P_1 \Leftarrow P_2$ (resp. $P_1 \equiv P_2$) akan menegaskan bahwa persoalan $P_1$ adalah dapat direduksi secara polynomial terhadap problem $P_2$.

### 5.1. Menentukan *Composite Residuosity*

Sebelumnya, akan diperkenalkan bahwa *composite degree residues* adalah sebuah contoh yang sangat alamiah dari *higher degree residues*, dan dapat memberikan beberapa fakta – fakta dasar yang saling berhubungan.

Inovasi yang terdapat disini terletak pada penggunaan bilangan kuadrat sebagai sebuah modulus. Seperti yang dijelaskan diatas bahwa n = pq adalah hasil perkalian dari dua buah bilangan prima yang sangat besar jumlahnya.

Definisi 1 : sebuah bilangan z disebut sebuah bilangan residu ke n dari modulo $n^2$ apalia terdapat sebuah bilangan $y \in \mathbb{Z}^*_{n^2}$ dimana $z = y^n \bmod n^2$

Himpunan bilangan residu yang ke n adalah sebuah sub-grup multiplikatif dari $\mathbb{Z}^*_{n^2}$ dengan pangkat $\phi(n)$. Tiap bilangan residu yang ke n – z memiliki tepat sejumlah n buah akar dengan derajat n. diantaranya hanya satu yang harus lebih kecil dari n (yakni $\sqrt[n]{z} \bmod n$ ). Akar ke n dari himpunan tersebut adalah bilangan yang berbentuk $(1 + n)^x = 1 + xn \bmod n^2$.

Permasalahan untuk menentukan bilangan residu ke n , contohnya memisahkan bilangan residu ke n dari bilangan residu yang bukan ke n, akan dituliskan dalam bentuk CR[n]. Seperti pada permasalahan untuk menentukan residu dari bilangan kuadratik atau yang lebih tinggi lagi, CR[n] adalah sebuah permasalahan acak yang sifatnya *self-reductible*, sehingga taip instansnya adalah ekivalen dalam tingkatan polynomial.
Tiap kejadian adalah kejadian yang umum, dan permasalahan tersebut sifatnya adalah salah satu dari *uniformly intractable* atau *uniformly polynomial*.

Sedangkan pada bilangan residu prima, menentukan residu yang ke n diyakini akan sangat sulit secara komputasi. Demikian juga, akan diasumsikan bahwa tidak ada pembeda waktu polinomial untuk residu ke n modulus $n^2$ yang *intractable*. Hipotesis mengenai *intractability* ini akan dirujuk sebagai *Decisional Composite Residuosity Assumption* (DCRA) untuk selanjutnya. Perhatikan pula bahwa karena adanya sifat *random-self-reducibility*, validitas dari DCRA hanya bergantung kepada pemilihan n.

### 5.2. Menghitung *Composite Residuosity*

Disini akan diterangkan kerangka kerja teori bilangan yang melandasi kriptosistem ini.

g adalah elemen dari $\mathbb{Z}^*_{n^2}$, dan dituliskan dengan $\mathcal{E}_g$, fungsi yang bernilai integer yang didefinisikan dengan $\mathbb{Z}_n \times \mathbb{Z}^*_n \longmapsto \mathbb{Z}^*_{n^2}$, kemudian $(x, y) \longmapsto g^x \cdot y^n \bmod n^2$.

Bergantung kepada g, $\mathcal{E}_g$ memiliki beberapa sifat unik, yaitu :

**Lemma 1** – Apabila orde g adalah kelipatan tidak nol dari n, maka $\mathcal{E}_g$bersifat bijektif.

Ditentukan bahwa $\mathcal{B}_\alpha \subset \mathbb{Z}^*_{n^2}$adalah himpunan elemen – elemen dengan orde $n\alpha$dan $\mathcal{B}$adalah himpunan disjoin untuk $\alpha = 1, \cdots, \lambda$.

Bukti : karena dua grup $\mathbb{Z}_n \times \mathbb{Z}^*_n$dan $\mathbb{Z}^*_{n^2}$ memiliki jumlah elemen yang sama, $n\phi(n)$, maka yang harus dibuktikan hanyalah bahwa $\mathcal{E}_g$adalah injektif. Anggap bahwa $g^{x_1} y_1^n = g^{x_2} y_2^n \bmod n^2$. Maka $\lambda(x_2 - x_1)$adalah kelipatan dari orde g, dan juga merupakan kelipatan dari n. karena $\gcd(\lambda, n) = 1$, $x_2 - x_1$ adalah kelipatan dari n. Karena itu, maka $x_2 - x_1 = 0 \bmod n$dan $(y_2/y_1)^n = 1 \bmod n^2$, yang menuju ke solusi yang unik $y_2/y_1 = 1$pada $\mathbb{Z}^*_n$. Hal ini berarti $x_2 = x_1$ dan $y_2 = y_1$. Oleh karena itu, maka $\mathcal{E}_g$ adalah bijektif.

Anggap bahwa $g \in \mathcal{B}$. Maka untuk $w \in \mathbb{Z}^*_{n^2}$, kelas residu ke n dari w, bergantung pada g, integer yang unik dari $x \in \mathbb{Z}_n$ dimana terdapat $y \in \mathbb{Z}^*_n$ sehingga $\mathcal{E}_g(x, y) = w$.

Mengadopsi notasi Benaloh, kelas dari w ditulis sebagai $[w]_g$ dan berikutnya lemma berikut :

**Lemma 2** - $[w]_g = 0$ jika dan hanya jika w adalah sebuah bilangan residu ke n dari modulus $n^2$. Lebih jauh lagi :
,

$$\forall w_1, w_2 \in \mathbb{Z}^*_{n^2} \quad [\![w_1 w_2]\!]_g = [\![w_1]\!]_g + [\![w_2]\!]_g \bmod n$$

Yakni sebuah kelas fungsi $w \mapsto [\![w]\!]_g$ adalah sebuah homomorfisme dari $(\mathbb{Z}^*_{n^2}, \times)\ to\ (\mathbb{Z}_n, +)$ untuk setiap $g \in \mathcal{B}$.

Permasalahan bilangan ke n dari kelas residu dengan basis g, dituliskan sebagai Class[n,g], dideginisikan sebagai permasalahan untuk menghitung fungsi kelas pada basis g, untuk sebuah $w \in \mathbb{Z}^*_{n^2}$.

Hitung $[w]_g$ dari w. Sebelum meneliti lebih jauh kompleksitas dari Clas[n,g], pengamatan berikut telah dilakukan :

**Lemma – 3** Class[n,g] adalah sebuah *random-self-reducible* pada $w \in \mathbb{Z}^*_{n^2}$.
Bukti tidak diberikan disini

**Lemma – 4** Class[n,g] adalah sebuah *random-self-reducible* pada $g \in \mathcal{B}$, misalnya :

$$\forall g_1,\ g_2 \in \mathcal{B} \quad Class[n, g_1] \quad \equiv \quad Class[n, g_2]$$

Bukti tidak diberikan disini

Lemma 4 pada dasarnya menyebutkan bahwa kompleksitas dari Class[n,g] adalah saling bebas dengan g. Hal ini memungkinkan untuk melihat bahwa Class[n.g] sebagai sebuah permasalahan komputasional yang bergantung pada n.

Permasalahan Kelas Residu Komposit dan permasalahan komputasi Class[n] didefinisikan sebagai berikut, apabila $w \in \mathbb{Z}^*_{n^2}$ dan $g \in \mathcal{B}$, maka hitung $[w]_g$. Kemudian, dapat ditemukan hubungan yang ada antara permasalahan Kelas Residu Komposit

dan permasalahan standard teori bilangan. Sebelumnya disebutkan bahwa :

**Teorema 1** –

$$Class\,[n] \quad \Leftarrow \quad Fact\,[n]$$

Amatilah bahwa himpunan $\mathcal{S}_n = \{u < n^2 \mid u = 1 \bmod n\}$ adalah subgrup multiplikatif dari integer modulus $n^2$ yang mana ada fungsi L dimana $\forall u \in \mathcal{S}_n \quad \mathrm{L}(u) = \dfrac{u-1}{n}$ dengan jelas didefinisikan.

**Lemma – 5** untuk setiap $w \in \mathbb{Z}^*_{n^2},\ L(w^\lambda \bmod n^2) = \lambda\,[\![w]\!]_{1+n} \bmod n$
Bukti tidak diberikan disini

**Teorema 2** –

$$Class\,[n] \quad \Leftarrow \quad RSA\,[n,n]$$

Bukti tidak diberikan disini.

**Teorema 3** – Jika D-Class[n] adalah permasalahan yang menentukan yang berhubungan dengan Class[n], contohnya bila diketahui $w \in \mathbb{Z}^*_{n^2},\ g \in \mathcal{B}$ dan $x \in \mathbb{Z}_n$, tentukan apakah x = $[w]_g$, maka :

$$CR\,[n] \quad \equiv \quad D\text{-}Class\,[n] \quad \Leftarrow \quad Class\,[n]$$

Bukti tidak diberikan disini.

Jadi kesimpulannya, hirarki komputasi yang ingin diturunkan adalah :

$$\mathrm{CR}\,[n] \equiv \mathrm{D\text{-}Class}\,[n] \Leftarrow \mathrm{Class}\,[n] \Leftarrow \mathrm{RSA}\,[n,n] \Leftarrow \mathrm{Fact}\,[n]$$

Dengan meninggalkan sebuah keraguan mengenai ekivalensi yang masih potensial, kecuali kemungkinan mengenai D-Class[n] dan Class[n]. Hipotesis mengenai *intractability* kedua adalah menganggap kesulitan dai permasalahan Kelas Residu Komposit dengan membuat *conjecture* berikut ini :

Tidak ada algoritma probabilistik dengan waktu polinomial untuk memecahkan permasalahan Kelas Residu Komposit. Contohnya, Class[n] adalah *intractable*.

Berlawanan dengan *Decisional Composite Residuosity Assumption*, *conjecture* ini akan diacu sebagai *Computational Composite Residuosity Assumption* (CCRA). Disini, sifat *random-self-reducibility*berarti validitas CCRA hanya bergantung pada pilihan n, Lebih jelasnya, jika DCRA benar, maka CCRA akan benar pula. Kebalikannya masih merupakan suatu pertanyaan terbuka.

### 5.3. Skema Enkripsi Probabilistik

Berdasarkan permasalahan Kelas Residu Komposit, dapat dibuat sebuah skema enkripsi kunci publik. Metode yang digunakan cukup natural, yakni menggunakan Eg untuk enkripsi dan reduksi polinomial dari Teorema 1 untuk dekripsi, menggunakan faktorisasi untuk *trapdoor*.

Isi n = pq dan pilih sebuah basis $g \in \mathcal{B}$ secara acak, seperti yang telah ditunjukkan sebelumnya. Hal ini dapat dilakukan secara efisien dengan mengecek apakah $\gcd\left(\mathrm{L}(g^\lambda \bmod n^2), n\right) = 1$.
Sekarang, anggap pasangan (n,g) sebagai parameter publik sementara pasangan (p, q) (atau yang ekivalen dengan ini) sebagai parameter privat. Kriptosistem akan digambarkan sebagai berikut :

**Encryption:**

plaintext $m < n$

select a random $r < n$

ciphertext $c = g^m \cdot r^n \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

plaintext $m = \dfrac{\mathrm{L}(c^\lambda \bmod n^2)}{\mathrm{L}(g^\lambda \bmod n^2)} \bmod n$

Kebenaran dari skema ini dengan mudah dapat diverifikasi dari Persamaan ke dua dan cukup mudah dilihat bahwa fungsi enkripsi ini adalah sebuah fungsi *trapdoor* dengan $\lambda$ (yang merupakan faktor – faktor yang mungkin dari n) sebagai rahasia *trapdoor*. Properti *one-wayness* berbasis pada permasalahan komputasi yang telah dibahas diatas.

**Teorema 4 -** Skema 1 adalah *one-way* jika dan hanya jika asumsi komputasi Residu Komposit (CCRA) berlaku.

Bukti – Kebalikan dari skema diatas secara definisi adalah permasalahan Kelas Residu Komposit yang sukses harus dapat menentukan *composite residuosity*, dan sebaliknya.

### 5.4. Teknik Permutasi *Trapdoor* Searah

Permutasi searah *trapdoor* adalah sebuah objek kriptografik yang sangat jarang. Pada bagian ini akan ditunjukkan bagaimana cara menggunakan teknik *trapdoor* yang sudah dibahas diatas untuk menurunkan permutasi pada $\mathbb{Z}^*_{n^2}$. Seperti sebelumnya, n adalah perkalian dari dua buah bilangan prima yang sangat besar dan g dipilih seperti pada persamaan dibawah ini :

**Encryption:**

plaintext $m < n^2$

split $m$ into $m_1$, $m_2$ such that $m = m_1 + nm_2$

ciphertext $c = g^{m_1} {m_2}^n \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

**Step 1.** $m_1 = \dfrac{\mathrm{L}(c^\lambda \bmod n^2)}{\mathrm{L}(g^\lambda \bmod n^2)} \bmod n$

**Step 2.** $c' = cg^{-m_1} \bmod n$

**Step 3.** $m_2 = {c'}^{n^{-1} \bmod \lambda} \bmod n$

plaintext $m = m_1 + nm_2$

**Teorema – 5** Skema 1 adalah aman secara semantik jika dan hanya jika *Decisional Composite Residuosity Assumption* (DCRA) berlaku

Bukti – Misalkan bahwa $m_0$ dan $m_1$ adalah dua buah pesan yang diketahui dan c adalah cipherteks dari salah satu dari $m_0$ atau $m_1$. Sesuai dengan lemma nomor 2, c adalah cipherteks dari $m_0$ jika dan hanya jika $cg^{-m_0} \bmod n^2$ adalah sebuah bilangan residu ke n. Oleh karena itu, seorang penyerang dengan teknik *chosen-plaintext-attack* Pertama – tama akan dibahas mengenai kebenaran dari skema.

Dengan jelas dapat dilihat bahwa langkah pertama secara benar mendapat $m_1 = m \bmod n$ seperti pada Skema 1 diatas. Langkah kedua adalah fase *unblinding*. Fase ini diperlukan untuk mendapatkan $m_2^n \bmod n$. Langkah ketiga adalah sebuah proses dekripsi RSA dengan kunci publik eksponen e = n. Langkah terakhir adalah mengkombinasikan ulang pesan awal m.

Kenyataan bahwa skema 2 adalah sebuah permutasi didapat dari sifat bijektif $\mathcal{E}_g$. Lagipula, *trapdoorness* sebenarnya berbasis pada faktorisasi n. Mempertimbangkan sifat searah (*one-wayness*), maka dapat dinyatakan bahwa :

**Teorema 6** – Skema 2 adalah searah jika dan hanya jika RSA[n,n] sulit dipecahkan.

Bukti :

a. Karena
$$\mathrm{Class}\,[n] \quad \Leftarrow \quad \mathrm{RSA}\,[n,n],$$
(lihat teorema 2), maka apabila diketahui akar ke n
dari modulus n, hal ini cukup untuk menghitung $m_1$ dari $\mathcal{E}_g(m_1, m_2)$. Untuk mendapat $m_2$, masih dibutuhkan satu ekstraksi lagi. Maka dari itu, untuk membalik proses skema 2 tidak lebih sulit dari mengekstrak akar ke n dari modulus n

b. Sebaliknya, sebuah *oracle* yang digunakan untuk membalik proses skema 2 dapat digunakan untuk mengekstraksi akar. Pertama – tama, dapatkan dua buah nomor dari *oracle,* yaitu a dan b, yang mana $1 + n = g^a b^n \bmod n^2$. Jika $w = y_0^n \bmod n$, maka cari lagi pada *oracle* untuk mendapat x dan y sehingga $w = g^x y^n \bmod n^2$. Karena $1+n \in \mathcal{B}$, maka diketahui bahwa ada nilai $x_0$ dimana $w = (1+n)^{x_0} y_0^n \bmod n^2$, dimana

$$w = (g^a b^n)^{x_0}\, y_0^n = g^{ax_0 \bmod n} \left(g^{ax_0 \operatorname{div} n} b^{x_0} y_0\right)^n \bmod n^2$$

Dengan identifikasi menggunakan $w = g^x y^n \bmod n^2$, nilai $x_0$ dapat dilihat sebagai $x_0 = xa^{-1} \bmod n$ dan juga

$$y_0 = y g^{-(ax_0 \operatorname{div} n)} b^{-x_0} \bmod n$$

yang merupakan nilai yang diinginkan.

Catatan 1 - Perhatikan bahwa dari definisi $E_g$, kriptosistem membutuhkan $m_2 \in \mathbb{Z}_n^*$, seperti pada setting RSA. Pada kejadian dimana $m_2 \notin \mathbb{Z}_n^*$ dapat berarti apakah dapat difaktorkan n atau mungkin menuju ke cipherteks nol untuk semua kemungkinan $m_1$. sebuah konsekuensi dari fakta ini adalah bahwa permutasi *trapdoor* tidak dapat digunakan untuk mengenkripsi pesan pendek yang panjangnya kurang dari n.

**Tanda – Tangan Digital**

Dengan menuliskan bahwa $h : \mathbb{N} \mapsto \{0,1\}^k \subset \mathbb{Z}_{n^2}^*$, sebuah fungsi hash dapat dilihat sebagai sebuah *random oracle*. Maka dapat dibangun sebuah skema untuk tanda tangan digital sebagai berikut ini : untuk setiap pesan m, si pembuat menghitung signature (s1, s2) dimana

$$\begin{cases} s_1 = \dfrac{\mathrm{L}(h(m)^\lambda \bmod n^2)}{\mathrm{L}(g^\lambda \bmod n^2)} \bmod n \\ s_2 = \left(h(m) g^{-s_1}\right)^{1/n \bmod \lambda} \bmod n \end{cases}$$

Dan si penerima mengecek apakah

$$h(m) \stackrel{?}{=} g^{s_1} s_2^n \bmod n^2$$

Pada *Random Oracle Model*, sebuah percobaan serangan pada skema digital signature diatas dengan menggunakan *adaptive chosen message attack* memiliki peluang sukses yang kecil karena RSA[n,n] adalah *intractable.* Meskipun permutasi trapdoor ini sebenarnya tidak terlalu dibutuhkan karena fungsinya mirip dengan RSA, untiknya objek seperti permutasi trapdoor ini menarik untuk dibahas. Selebihnya lagi, properti homomorfisme dari skema ini dapat digunakan untuk beberapa permasalah kriptografik yang ada.

### 5.5. Mencapai Kompleksitas Dekripsi Mendekati Kuadratik

Kebanyakan kriptosistem kunci publik yang ada memiliki kerumitan dekripsi yang berorde kubik (pangkat tiga). Hal ini juga berlaku untuk skema 1. Faktanya tidak ada desain yang lebih cepat dan aman yang sudah diajukan menyebabkan tingginya motivasi untuk mencari fungsi trapdoor yang berguan untuk meningkatkan performansi proses dekripsi.

Pada bagian ini akan dibahas versi modifikasi dari Skema 1 yang memiliki kompleksitas dekripsi $\mathcal{O}\left(|n|^{2+\epsilon}\right)$.

Gagasan ini berbasis pada pembatasan ruang cipherteks $\mathbb{Z}_{n^2}^*$ kepada subgrup $< g >$ yang berorde lebih kecil dengan menggunakan kelebihan dari ekstensi Equation 2. Anggap bahwa $g \in \mathcal{B}_\alpha$ untuk $1 \leq \alpha \leq \lambda$, maka untuk tiap $w \in < g >$,

$$[\![w]\!]_g = \frac{\mathrm{L}(w^\alpha \bmod n^2)}{\mathrm{L}(g^\alpha \bmod n^2)} \bmod n$$

.

Hal ini menyebabkan bahwa kriptosistem dapat diubah menjadi seperti dibawah ini :

Perhatikanlah bahwa kali ini properti *trapdoor* dari fungsi enkripsi bergantung kepada α (bukannya $\lambda$) sebagai kunci privat. Komputasi yang paling mahal dalam proses dekripsi adalah operasi eksponensiasi modular dari $c \rightarrow c^\alpha \bmod n^2$ dengan kompleksitas $\mathcal{O}\left(|n|^2|\alpha|\right)$ (lebih baik dibandingkan dengan kompleksitas kebanyakan kriptosistem kunci publik dan skema 1 yang kompleksitasnya $\mathcal{O}\left(|n|^3\right)$). Apabila nilai g diisi sehingga $|\alpha| = \Omega\left(|n|^\epsilon\right)$, untuk setiap $\epsilon > 0$, maka dekripsi hanya akan memiliki kompleksitas $\mathcal{O}\left(|n|^{2+\epsilon}\right)$ operasi bit. Sampai saat ini, hanya skema 3 yang memiliki fitur seperti itu.

Untuk membalik fungsi enkripsi, tidak ada ketergantungan pada permasalahan Kelas Residu Komposit, karena cipherteks dikenali sebagai elemen dari subgrup $< g >$.

**Teorema 7**, permasalahan komputasi *Partial Discrete Logarithm*, disebut sebagai PDL[n,g] didefinisikan sebagai berikut : apabila $w \in < g >$, hitung $[w]_g$ kemudian tentukan apakah $[w]_g$ = x. Skema 3 diatas akan aman jika dan hanya jika PDL[n,g] adalah sulit dipecahkan secara komputasional.

**Encryption:**

plaintext $m < n$

randomly select $r < n$

ciphertext $c = g^{m+nr} \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

plaintext $m = \dfrac{\mathrm{L}(c^\alpha \bmod n^2)}{\mathrm{L}(g^\alpha \bmod n^2)} \bmod n$

**Teorema 8** permasalahan komputasi *Decisional Partial Discrete Logarithm*, disebut sebagai permasalahan D-PDL[n,g] didefinisikan sebagai berikut : apabila $w \in <g>$ dan $x \in \mathbb{Z}_n$, tentukan apakah $[w]_g = x$. Skema 3 diatas akan aman jika dan hanya jika D-PDL[n,g] adalah sulit secara komputasional.

Bukti – bukti dari teorema diatas tidak disertakan.

Kebalikan dari permasalahan kelas asal, permasalahan – permasalahan diatas tidak bersifat *random-self-reducible* pada $g \in \mathcal{B}$ namun pada subgrup yang sifatnya siklik dari $\mathcal{B}$. Lebih lanjut lagi, dari dua teorema diatas dapat disimpulkan bahwa:

$$\text{PDL}\,[n,g] \quad \Leftarrow \quad \text{Class}\,[n]$$

Dan

$$\text{D-PDL}\,[n,g] \quad \Leftarrow \quad \text{CR}\,[n]$$

Namun ekivalensi dapat dicapai apabila g memiliki orde maksimum $n\lambda$dan $n$, hasil perkalian dari dua buah bilangan prima. Apabila $g \in \mathcal{B}$ pada rentang $\alpha < \lambda$ sehingga $|\alpha| = \Omega\left(|n|^{\epsilon}\right)$ untuk $\epsilon > 0$. Dengan aman dapat disimpulkan bahwa PDL[n,g] dan D-PDL[n,g] bersifat *intractable*.

Dalam menghadapi serangan yang bertipe *Baby-Step Giant-Step*, sangat disarankan penggunaan bilangan prima 160-bit untuk . Penggunaan bilangan prima 160-bit ini dapat dicapai apabila menggunakan algoritma pembangkitan kunci yang sesuai. Pada setting ini, kebutuhan komputasi untuk skema yang ketiga ini lebih kecil daripada dekripsi RSA yang sudah menggunakan algoritma *Chinese Remaindering* untuk optimisasi bila n lebih besar daripada 1280.

## 6. Metode Enkripsi dan Dekripsi

Seperti yang sudah dibahas pada dasar teori diatas, ada beberapa skema yang dapat digunakan untuk mengenkripsi, dan tentu saja untuk mendekripsi skema yang bersangkutan.

Dapat dilihat bahwa ada tiga skema yang dapat digunakan untuk melakukan enkripsi pada sebuah plainteks. Skema 1 adalah merupakan turunan dari teori *Composite Residuosity Classes*

**Encryption:**

plaintext $m < n$

select a random $r < n$

ciphertext $c = g^m \cdot r^n \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

$$\text{plaintext } m = \frac{\text{L}(c^{\lambda} \bmod n^2)}{\text{L}(g^{\lambda} \bmod n^2)} \bmod n$$

Skema 2 merupakan variasi dari skema satu yang menekankan pada teknik *one way trapdoor* yang diperoleh melalui faktorisasi m menjadi $m_1$ dan $n.m_2$. Cara enkripsi dan dekripsi sebenarnya sama, perbedaan hanya pada pemilihan bilangan yang menjadi derajat *g* dan *r* saja, yang pada skema yang kedua ini diubah menjadi $m_1$ dan $m_2$. Skema enkripsi dan dekripsi dapat dilihat dibawah ini :

**Encryption:**

plaintext $m < n^2$

split $m$ into $m_1$, $m_2$ such that $m = m_1 + nm_2$

ciphertext $c = g^{m_1} {m_2}^n \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

**Step 1.** $\displaystyle m_1 = \frac{\text{L}(c^{\lambda} \bmod n^2)}{\text{L}(g^{\lambda} \bmod n^2)} \bmod n$

**Step 2.** $c' = cg^{-m_1} \bmod n$

**Step 3.** $m_2 = {c'}^{n^{-1} \bmod \lambda} \bmod n$

plaintext $m = m_1 + nm_2$

Sedangkan skema ketiga adalah usaha untuk mengurangi waktu dekripsi yang tadinya kubik menjadi mendekati kuadratik. Yang perlu diperhatikan adalah bahwa proses enkripsi pada Paillier terhitung tidak berat (hal ini masih harus melihat bagaimanakah algoritma random generator untuk membangkitkan n = pq). Usaha ini dilakukan dengan membatasi ruang lingkup cipherteks menjadi subgroup yang memiliki orde lebih kecil. Karena itu, varian yang ketiga ini sering juga disebut sebagai *fast varians* dari kriptosistem paillier.Berikut adalah skema enkripsi dan dekripsi untuk skema 3

**Encryption:**

plaintext $m < n$

randomly select $r < n$

ciphertext $c = g^{m+nr} \bmod n^2$

**Decryption:**

ciphertext $c < n^2$

plaintext $m = \dfrac{\mathrm{L}(c^{\alpha} \bmod n^2)}{\mathrm{L}(g^{\alpha} \bmod n^2)} \bmod n$

## 7. Contoh Pengaplikasian Kriptosistem Paillier

Kriptosistem paillier memiliki beberapa aplikasi dunia nyata dimana kriptosistem ini dapat digunakan. Beberapa diantaranya adalah :

### 7.1. Pemilihan Suara Elektronik

Pada sistem pemilihan suara elektronik, kriptosistem paillier ini dapat merupakan pilihan yang cocok karena beberapa pertimbangan :

1. *Homomorphic property*. Kertas suara kriptografik tidak mendukung penulisan langsung calon pemilih. Umumnya, apabila Alice ingin menuliskan sebuah nama, ia akan memilih untuk menulis langsung pilihan kandidat – kandidat yang udah ada, maka dari pada itu dapat digunakan *homomorhic tallying*. Properti homomorfisme ini sejalan dengan properti homomorfisme dari kriposistem paillier, sehingga untuk properti homomorfisme dari Electronic Voting ini dapat digunakan kriptosistem paillier. Properti homomorfisme memungkinkan penjumlahan nilai yang dienkripsi dengan nilai lain, kemudian hasil penjumlahannya dapat didekripsi tanpa mengetahui nilai – nilai yang membentuknya.

2. Karena properti *semantic security* dari kriptosistem Paillier ini, maka suara yang masuk dapat dilindungi dari serangan yang bersifat Chosen Plaintext attack, namun karena sifatnya yang malleable, maka suara yang masuk masih dapat diserang melalui metode Chosen Ciphertext attack, kecuali apabila menggunakan skema ROM

3. Properti malleability yang dimiliki oleh kriptosistem Paillier ini.

### 7.2. Uang Elektronik

Fitur lain yang dapat digunakan adalah fitur *self-blinding*. Fitur ini adalah untuk mengganti sebuah cipherteks ke cipherteks yang lain tanpa mengganti isi pesan pada waktu didekripsi. Hal ini dapat digunakan untuk mengembangkan uang elektronik, sebuah usaha yang dipelopori oleh David Chaum

Hal ini berarti uang elektronik harus dapat digunakan dalam pembayaran online tanpa menggunakan nomor kartu kredit, dan tanpa menggunakan identitas pemakai (sama seperti penggunaan uang real yang lain)

Tujuan dari penggunaan kriptosistem Paillier ini ialah untuk memastikan bahwa suara atau uang yang dipergunakan adalah valid tanpa perlu mengetahui identitas pemilih atau pembayar yang bersangkutan.

## 8. Optimisasi dan Implementasi

Pada bagian ini akan dibahas cara – cara untuk meningkatkan performansi dari kriptosistem Paillier ini.

### 8.1. Pembangkitan Kunci

Faktor prima p dan q akan dibangkitkan sesuai dengan kaidah – kaidah yang umum berlaku supaya n menjadi sangat sulit difaktorkan. Untuk Skema 3 dibtuhkan juga bahwa $\lambda = \mathrm{lcm}(p-1, q-1)$ sebagai kelipatan dari bilangan prima 160 bit, dimana bilangan ini dapat didapatkan dari pembangkitan bilangan acak prima yang biasa digunakan pada DSA, RSA ataupun teknik – teknik yang bersesuaian yang lainnya.

Bilangan basis g dapat dipilih secara acak dari elemen – elemen yang ordenya dapat dibagi dengan n, namun perlu diperhatikan bahwa skema 3 ini akan memerlukan perlakuan yang khusus (biasanya berupa meningkatkan elemen dari orde yang paling tinggi ke pangkat $\lambda/\alpha$). Seluruh proses pembangkitan kunci ini dapat dipermudah dengan menggunakan komputasi mod $p_2$ dan mod $q_2$ dan *chinese – remaindering* g mod $p_2$ dan g mod $q_2$ pada akhirnya.

### 8.2. Enkripsi

Enkripsi membutuhkan eksponensiasi modular dengan menggunakan basis g. Komputasi yang dilakukan dapat dipercepat dengan menggunakan pilihan g yang tepat. Contohnya, mengisi g dengan bilangan yang nilainya kecil, seperti g = 2 misalnya, akan membawa percepatan yang signifikan dengan faktor 1/3, apabila nilai yang dipilih memenuhi kebutuhan persyaratan $g \in \mathcal{B}$.

Selain itu, g dapat saja dibuat tetap nilainya apabila proses pembangkitan kunci ini sudah melalui beberapa penyesuaian desain. Terlebih lagi, perhitungan perpangkatan untuk basis - basis yang sudah dibuat konstan tadi pastinya akan membawa percepatan dan mengurangi kebutuhan kekuatan komputasi. Komputasi berikut mengenai $r^n$ dan $g^{nr}$ dapat dihitung pada saat proses enkripsi berjalan.

### 8.3. Dekripsi

Untuk menghitung L(u) untuk $u \in \mathcal{S}_n$ dapat dihitung dengan kebutuhan komputasi yang sangat rendah, dengan menghitung $n^{-1} \bmod 2^{|n|}$. Parameter konstan

$$\mathrm{L}(g^{\lambda} \bmod n^2)^{-1} \bmod n \quad \text{or} \quad \mathrm{L}(g^{\alpha} \bmod n^2)^{-1}$$

Juga dapat dikomputasi di depan

### 8.4. Dekripsi Menggunakan Teknik Chinese Remaindering

Teorema *Chinese Remaindering* dapat digunakan untuk mengurangi beban komputasi pada tiga skema yang ada diatas secara signifikan. Untuk itu, fungsi $L_p$ dan $L_q$ yang didefinisikan dengan menggunakan :

$$\mathcal{S}_p = \{x < p^2 \mid x = 1 \bmod p\}$$

Dan

$$\mathcal{S}_q = \{x < q^2 \mid x = 1 \bmod q\}$$

Dengan

$$\mathrm{L}_p(x) = \frac{x-1}{p}$$

Dan

$$\mathrm{L}_q(x) = \frac{x-1}{q}$$

Proses dekripsi kemudian dapat dibuat lebih cepat dengan menghitung pesan mod p dan mod q secara terpisah, lalu kemudian menggabungkan kembali sisa modulus setelah itu :

$$m_p = \mathrm{L}_p(c^{p-1} \bmod p^2)\, h_p \bmod p$$
$$m_q = \mathrm{L}_q(c^{q-1} \bmod q^2)\, h_q \bmod q$$
$$m = \mathrm{CRT}(m_p, m_q) \bmod pq$$

Dengan menggunakan komputasi di depan yakni

$$h_p = \mathrm{L}_p(g^{p-1} \bmod p^2)^{-1} \bmod p$$

Dan

$$h_q = \mathrm{L}_q(g^{q-1} \bmod q^2)^{-1} \bmod q$$

Dimana p – 1 dan q – 1 akan digantikan menggunakan $\alpha$ pada Skema 3

### 8.5. Tinjauan Performansi

Untuk setiap $|n|$ = 512, … , 2048, perkalian modular dari ukuran bit $|n|$ dianggap sebagai operasi unitary, dan anggap waktu eksekusi perkalian modular adalah kuadratik yang bergantung pada ukuran operand dan perpangkatan dua modular tersebut akan dihitung menggunakan fungsi yang sama.

*Chinese remaindering*, dan juga algoritma pembangkitan bilangan acak pada skema probabilistik, dianggap tidak terlalu mempengaruhi perhitungan komputasi karena workload yang tidak begitu besar. Hal ini dapat dilihat pada bagian sebelumnya, dimana disebutkan bahwa teknik *chinese remaindering* ini dapat mengurani beban komputasi pada tiga skema yang ada secara signifikan, sehingga beban komputasinya dapat diabaikan.

Perpangkatan public RSA dianggap sama dengan $\mathrm{F}_4 = 2^{16} + 1$. Parameter g diisi dengan nilai 2 pada skema utama, dan juga pada skema yang menggunakan *trapdoor permutation* (skema 2).

Parameter – parameter lainnya, perpangkatan atau pesan rahasia dianggap memiliki jumlah bit 0 dan 1 dengan jumlah yang sama pada representasi biner bilangan tersebut.

Dibawah ini akan ditunjukkan estimasi dari performansi dari ketiga skema :

| Schemes | Main Scheme | Permutation | Fast Variant | RSA | ElGamal |
|---|---|---|---|---|---|
| One-wayness | Class $[n]$ | RSA $[n, n]$ | PDL $[n, g]$ | RSA $[n, \mathrm{F}_4]$ | DH $[p]$ |
| Semantic Sec. | CR $[n]$ | none | D-PDL $[n, g]$ | none | D-DH $[p]$ |
| Plaintext size | $\|n\|$ | $2\|n\|$ | $\|n\|$ | $\|n\|$ | $\|p\|$ |
| Ciphertext size | $2\|n\|$ | $2\|n\|$ | $2\|n\|$ | $\|n\|$ | $2\|p\|$ |

| Encryption | | | | | |
|---|---|---|---|---|---|
| $\|n\|, \|p\| = 512$ | 5120 | 5120 | 4032 | 17 | 1536 |
| $\|n\|, \|p\| = 768$ | 7680 | 7680 | 5568 | 17 | 2304 |
| $\|n\|, \|p\| = 1024$ | 10240 | 10240 | 7104 | 17 | 3072 |
| $\|n\|, \|p\| = 1536$ | 15360 | 1536 | 10176 | 17 | 4608 |
| $\|n\|, \|p\| = 2048$ | 20480 | 20480 | 13248 | 17 | 6144 |

| Decryption | | | | | |
|---|---|---|---|---|---|
| $\|n\|, \|p\| = 512$ | 768 | 1088 | 480 | **192** | 768 |
| $\|n\|, \|p\| = 768$ | 1152 | 1632 | 480 | **288** | 1152 |
| $\|n\|, \|p\| = 1024$ | 1536 | 2176 | 480 | **384** | 1536 |
| $\|n\|, \|p\| = 1536$ | 2304 | 3264 | **480** | 576 | 2304 |
| $\|n\|, \|p\| = 2048$ | 3072 | 4352 | **480** | 768 | 3072 |

## 9. Lampiran Source Code

Disini akan diberikan lampiran kode dalam bahasa Java, dan berbasis pada kode yang digunakan untuk proyek Kert Richardson.

Namun karena keterbatasan tempat maka hanya akan diberikan potongan kode yang relevan saja, seperti dekripsi dan enkripsi.

**Kelas PrivateKey**

```
static public class PrivateKey {
        /* bilangan p dan q */
        private BigInteger p, q;
        /* Konstruktor untuk menggenerate bilangan 512 - bit.
         */
        public PrivateKey () {
            this (512, 64);
        }
        /* Konstruktor untuk mengisi nilai p dan q
         *  bits adalah nomor bit dalam modulus
         *   certainty adalah kemungkinan bilangan p dan q bukan
prima harus lebih kecil dari 2 ^ (-certainty)
         */
        public PrivateKey (int bits, int certainty) {
            Random rand = new Random ();
            p = new BigInteger (bits / 2, certainty, rand);
            q = new BigInteger (bits / 2, certainty, rand);
            while (q.compareTo (p) == 0)
                q = new BigInteger (bits / 2, certainty, rand);
        }

        /** Metode untuk mengekstraksi modulus*/
        public BigInteger GetN () {
            return p.multiply (q);
        }

        /** metode dekripsi */
        public BigInteger Decrypt (BigInteger cip) {

            BigInteger n = GetN ();

            int s = (cip.bitLength () - 1)/n.bitLength();

            return Decrypt (cip, s);
        }

        /** dekripsi dengan menggunakan parameter s.
```

```
         *  dengan kemungkinan benar 1/n
         */
        public BigInteger Decrypt (BigInteger cip, int s) {

            BigInteger temp; // untuk menyimpan hasil sementara.
            BigInteger count;

            BigInteger[] n = new BigInteger[s+1]; /
            n[0] = GetN ();
            for (int i = 0; i < s; i++) n[i+1] = n[i].multiply
(n[0]);

            BigInteger ONE = BigInteger.ONE;

            // isi nilai lambda
            BigInteger   p1_q1   =   (p.subtract   (ONE)).multiply
(q.subtract (ONE)); // (p - 1)(q - 1)
            BigInteger lambda = p1_q1.divide ((p.subtract (ONE)).gcd
(q.subtract (ONE)));

            // isi nilai d.
            BigInteger d;
            d = lambda.multiply (lambda.modInverse (n[s-1]));

            // Dekripsi cipherteks.
            BigInteger cip_d;
            BigInteger[] L = new BigInteger[s];
            BigInteger[] mult = new BigInteger[s-1];
            BigInteger msg;

            cip_d = cip.modPow (d, n[s]);

            // L[i] = ((c^d mod n^{i+2}) - 1) / n
            for (int i = 0; i < s; i++)
                L[i] = ((cip_d.mod (n[i+1])).subtract (ONE)).divide
(n[0]);

            temp = ONE;  /
            count = ONE;
            for (int i = 1; i < s; i++) {

                count = count.add (ONE);
                temp = temp.multiply (count);
                mult[i-1] = (n[i-1].multiply (temp.modInverse (n[s-
1]))).mod (n[s-1]);
            }

            BigInteger t1, t2;
            msg = null;
            for (int j = 1; j <= s; j++) {
                t1 = L[j-1];
                t2 = msg;
                for (int k = 2; k <= j; k++) {
                    msg = msg.subtract (ONE);
                    t2 = (t2.multiply (msg)).mod (n[j-1]);
                    t1 = (t1.subtract (t2.multiply (mult[k-2].mod
(n[j-1])))).mod (n[j-1]);
```

```
                }
                msg = t1;
            }

            return msg;
        }
    }
```

**Kelas PublicKey**

```
    static public class PublicKey {

        /* nilai kunci publik */
        protected BigInteger n;

        /** Konstruktor untuk mengisi kunci publik */
        public PublicKey (BigInteger n) {
            this.n = n;
        }

        /** mengambil nilai kunci publik */
        public BigInteger GetN () {
            return n;
        }

        /** Mengambil nilai dari modulus cipherteks  */
        public BigInteger GetCiphertextModulus (int s) {
            return n.pow (s+1);
        }

        /** Mengambil nilai dari modulus cipherteks dengan nilai s
yang spesifik */
        public BigInteger GetPlaintextModulus (int s) {
            return n.pow (s);
        }

         /** enkripsi pesan menggunakan public key dan s  */
        public   BigInteger   Encrypt   (BigInteger   msg)   throws
MessageToBigException {
            /* cari s yang cukup besar */
            int s = (msg.bitLength ()/n.bitLength()) + 1;
            BigInteger n_s = n.pow (s);          // n^s

            while (n_s.compareTo (msg) <= 0) {
                n_s = n_s.multiply (n);
                s++;
            }
            try {
                return Encrypt (msg, s);
            }
            catch (MessageToBigException e) {
                System.out.println ("Assertion failure");
                System.exit (0);
            }
            return null;
```

```
        }

        /** enkripsi pesan menggunakan public key dan s  */
        public BigInteger Encrypt (BigInteger msg, int s) throws
MessageToBigException {
            BigInteger n_s = n.pow (s);          // n^s

            if (msg.compareTo (n_s) >= 0)
                throw new MessageToBigException ("Message to big for
encryption with " + s);

            BigInteger n_s1 = n.multiply (n_s);  // n^{s+1}

            // buat sebuah nilai random
            Random rand = new Random ();
            BigInteger r = new BigInteger (n_s1.bitLength (), rand);
            while ((r.compareTo (n_s1) >= 0) ||
                   (r.compareTo (BigInteger.ZERO) == 0) ||
                   (BigInteger.ONE.compareTo (r.gcd (n)) < 0))
                r = new BigInteger (n_s1.bitLength (), rand);

            // hitung hasil
            BigInteger res = BigInteger.ONE.add (msg.multiply (n));
            BigInteger binomial = msg;   // nilai dari binomial
            BigInteger msg_i    = msg;   // msg - i + 1
            BigInteger big_i     = BigInteger.ONE; // BigInteger yang
berisi i
            BigInteger n_i      = n;     // pangkat ke i dari n

            for (int i = 2; i <= s; i++) {
                n_i = n_i.multiply (n);
                msg_i = msg_i.subtract (BigInteger.ONE);
                big_i = big_i.add (BigInteger.ONE);
                binomial   =   ((binomial.multiply   (msg_i)).multiply
(big_i.modInverse (n_s1))).mod (n_s1);
                res   =   (res.add   (binomial.multiply   (n_i))).mod
(n_s1);
            }
            res = (res.multiply (r.modPow (n_s, n_s1))).mod (n_s1);

            return res;
        }
    }

    /**
     * menggabungkan dua buah cipherteks sehingga cipherteks hasil
merupakan
    * gabungan dari dua buah plainteks yang dienkripsi
    * merupakan manifestasi dari properti homomorfisme
     */
    public static BigInteger CombineCiphertexts(PublicKey pub,
            BigInteger cip1,
            BigInteger cip2) throws MitchmatchedSizeException {

        int       s1       =       (cip1.bitLength       ()       -
1)/pub.GetPlaintextModulus(1).bitLength();
```

```
        int       s2       =       (cip2.bitLength       ()       -
1)/pub.GetPlaintextModulus(1).bitLength();
        if (s1 != s2) {
            throw     new     MitchmatchedSizeException("Sizes     of
ciphertexts does not match (" + s1 + " != " + s2 + ").");
        }

        return CombineCiphertexts(pub, cip1, cip2, s1);
    }

     /**
     * menggabungkan dua buah cipherteks sehingga cipherteks hasil
merupakan
    * gabungan dari dua buah plainteks yang dienkripsi
    * merupakan manifestasi dari properti homomorfisme
     */
    public static BigInteger CombineCiphertexts(PublicKey pub,
            BigInteger cip1,
            BigInteger cip2,
            int s) {

        return cip1.multiply(cip2).mod(pub.GetCiphertextModulus(s));
```

## 10. Daftar Pustaka


1. P. Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, Gemplus Card International, 1999
2. M. Bellare dan P. Rogaway, Random Oracles are Practical: a Paradigm for Designing Efficient Protocols, In Proceedings of the First CCS, ACM Press, pp. 62{73,1993.
3. J. C. Benaloh, Verifiable Secret-Ballot Elections, PhD Thesis, Yale University, 1988.
4. D. Angluin dan D. Lichtenstein, Provable Security of Cryptosystems: A Survey, Computer Science Department, Yale University, TR-288, 1983.
5. R. Cramer, R. Gennaro dan B. Schoenmakers, A Secure And Optimally Effcient Multi-Authority Election Scheme, LNCS 1233, Proceedings of Eurocrypt'97, Springer-Verlag, pp. 103-118, 1997.
6. Wikipedia http://en.wikipedia.org/wiki/Paillier
7. Project Site http://www.cs.rit.edu/
8. W. Diffie dan M. Hellman, New Directions in Cryptography, IEEE Transaction on Information Theory, IT-22,6, pp. 644{654, 1995.
9. C. Ding, D. Pei dan A. Salomaa, Chinese Remainder Theorem - Applications in Computing, Coding, Cryptography, World Scienti_c Publishing, 1996.
10. T. ElGamal, A Public-Key Cryptosystem an a Signature Scheme Based on Discrete Logarithms, IEEE Trans. on Information Theory, IT-31, pp. 469{472, 1985.
11. J. Feigenbaum, Locally Random Reductions in Interactive Complexity Theory, in Advances in Computational Complexity Theory, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 13, American Mathematical Society, Providence, pp. 73{98, 1993.
12. S. Goldwasser dan S. Micali, Probabilistic Encryption, JCSS Vol. 28 No 2, pp. 270{299, 1984.