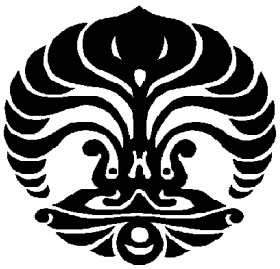
ANALISA PROTOKOL PEMBAYARAN ANTARA *CARDHOLDER* DAN *MERCHANT* PADA PROTOKOL *SECURE ELECTRONIC TRANSACTION* SERTA IMPLEMENTASI *OBJECT LIBRARY* PENDUKUNGNYA

Disusun sebagai Laporan Tugas Akhir

Oleh :
Dwinanda Prayudi



Fakultas Ilmu Komputer
Universitas Indonesia
Depok
1999

KATA PENGANTAR

Puji dan syukur dipanjatkan kehadirat Allah SWT, karena atas rahmat dan karunia-NYA, penulis dapat menyelesaikan tugas akhir ini.

Tugas akhir ini berjudul "Analisa Protokol Pembayaran Antara *Cardholder* dan *Merchant* pada Protokol *Secure Electronic Transaction* serta Implementasi *Object Library* Pendukungnya" dan merupakan salah satu syarat untuk memperoleh gelar Sarjana Ilmu Komputer pada Fakultas Ilmu Komputer Universitas Indonesia.

Selanjutnya penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Kedua orang tua penulis yang telah membesarkan penulis dengan penuh kasih sayang, yang selalu memberikan dukungan pada penulis, dan tak pernah bosannya selalu bertanya kepada penulis "Kamu kapan lulusnya ?"
2. Bapak Bob Hardian M. Kom sebagai pembimbing tugas akhir, yang telah bersedia memberikan waktu, perhatian, dan pemikirannya untuk membimbing penulis dalam mengerjakan tugas akhir ini.
3. Bapak FX. Nursalim Hadi Ph.D sebagai pembimbing awal tugas akhir dan telah bersedia memberikan pemikirannya dan fasilitas pada awal pengerjaan tugas akhir ini.
4. Arrianto Mukti Wibowo, S. Kom sebagai asisten pembimbing tugas akhir, yang tanpa lelah selalu mengarahkan penelitian dan memotivasi penulis untuk menyelesaikan tugas akhir, walaupun sudah pergi ke negeri seberang.
5. Bapak Prof. Toemin Masoem sebagai pembimbing akademis, yang telah membimbing penulis selama menjadi mahasiswa di Fakultas Ilmu Komputer UI.
6. Haris, Arif, Iman, Christine, dan teman-teman seruangan lainnya di 1222, yang ikut berjuang bersama menyelesaikan tugas akhir. Terima kasih atas kerja sama yang baik dan kebersamaan yang dialami selama 6 bulan terakhir.
7. Teman-teman angkatan 94 Fasilkom UI, terutama teman-teman dari Gank Jalan 94, yang terus memberikan motivasi dan sedikit ancaman agar penulis menyelesaikan tugas akhir. Akhirnya penulis bisa menyamai mereka sebagai sarjana.
8. Batti sebagai teman dan sahabat penulis yang menemani di kala susah maupun senang.
9. Idauli dan teman-teman dari Pondok Cempaka yang bersedia menemani dan menyemangati penulis sewaktu penulis kehilangan semangat untuk mengerjakan tugas

akhir.

10. Teman-teman dari angkatan lain yang tidak bisa disebutkan satu persatu di sini. Terima kasih atas keceriaan dan kegaringan yang dialami bersama.
11. Staf keamanan Fasilkom yang setiap malam diganggu oleh kedatangan penulis, staf sekretariat yang pusing mengatur jadwal sidang, dan segenap staf serta karyawan Fasilkom/Pusilkom UI lainnya yang tidak bisa disebutkan satu persatu di sini. Terima kasih atas kerjasamanya.

Penulis menyadari tugas akhir ini masih banyak kekurangannya, tetapi penulis sangat mengharapkan agar tugas akhir ini dapat memberikan manfaat bagi pihak-pihak yang berkepentingan.

Depok, Agustus 1999

Penulis

ABSTRAK

Dengan semakin berkembangnya perdagangan di Internet, banyak pula protokol-protokol yang dipergunakan untuk perdagangan di Internet. Salah satu protokol yang dianggap paling aman adalah *Secure Electronic Transaction* (SET) yang dikeluarkan oleh Visa dan Mastercard. Di Indonesia belum ada yang mengimplementasikan SET untuk perdagangan di Internet. Oleh karena itu dalam penelitian ini dicoba untuk menganalisa dan mengimplementasikan SET.

Tujuan penelitian adalah untuk menganalisa protokol SET dan mengimplementasi objek-objek yang dipakai dalam protokol SET. Karena protokol SET tersebut sangat kompleks, maka penelitian dilakukan hanya pada komunikasi antara *Cardholder* dan *Merchant.*

Penelitian dimulai dengan membaca ketiga buku referensi yang dikeluarkan Visa dan Mastercard. Kemudian mempelajari ASN.1, perangkat kriptografi yang dibutuhkan dalam SET, encoding data dalam format DER, dan merancang objek-objek yang akan diimplemetasikan. Lalu dimulailah pembuatan objek-objek tersebut, sambil merancang metode pengujiannya.

Hasil penelitian adalah *object library* yang dipakai dalam komunikasi antara *Cardholder* dan *Merchant.*

ix + 108 hlm; 12 gbr; 37 tbl;
Bibliografi: 12 (1993-1999)

DAFTAR ISI

KATA PENGANTAR i

ABSTRAK iii

DAFTAR ISI iv

DAFTAR TABEL vii

DAFTAR GAMBAR ix

BAB I PENDAHULUAN 1
- 1.1 Latar Belakang Masalah 1
- 1.2 Tujuan Penelitian 2
- 1.3 Pembatasan Masalah 2
- 1.4 Metodologi Penelitian 3
- 1.5 Sistematika Penulisan 3

BAB II LANDASAN TEORI 5
- 2.1 Enkripsi Simetris 6
- 2.2 Enkripsi Asimetris 7
- 2.3 Fungsi Hash Satu Arah 8
- 2.4 Tanda Tangan Digital 9
- 2.5 Tanda Tangan Ganda 10
- 2.6 Amplop Digital 11
- 2.7 Sertifikat Digital 12
- 2.8 ASN.1 dan DER 14
  - 2.8.1 ASN.1 14
  - 2.8.2 DER 17

BAB III SPESIFIKASI SET 20
- 3.1 Protokol 20
- 3.2 Kebutuhan Bisnis Transaksi Melalui Internet 21
  - 3.2.1 Confidentiality 21
  - 3.2.2 Authentication 21

3.2.3 Integrity 22
3.2.4 Interoperability 22
3.3 Pihak-Pihak Yang Terlibat 23
3.4 Alur Transaksi SET 24
3.5 Batasan SET 26
3.6 Komunikasi Antara Cardholder Dan Merchant 26
3.7 Pesan-Pesan Pembayaran Dalam SET 28
3.8 Notasi Penulisan 30
3.9 Operator Kriptografi 32
3.9.1 Fungsi *Hash* 32
3.9.2 Tanda Tangan Digital 33
3.9.3 Enkripsi 33
3.9.4 Enkapsulasi 34
3.10 Struktur Data 34
3.10.1 TransIDs 34
3.10.2 PI (*Payment Instruction*) 35
3.10.3 InstallRecurData 39
3.10.4 AcqCardMsg 40
3.10.5 PANData 41
3.10.6 PANToken 42
3.10.7 AmountFields 43
3.10.8 Blok OAEP 44
3.10.9 MessageWrapper 44
3.10.10 Error 46
3.11 Pasangan Pesan Yang Dipertukarkan Antara *Cardholder* dan *Merchant* 47
3.11.1 PInitReq/PInitRes 47
3.11.2 PReq/PRes 54
3.11.3 InqReq/InqRes 70

BAB IV ANALISA DAN PERANCANGAN 73
4.1 SET Menjawab Kebutuhan Bisnis 73
4.1.1 Kerahasiaan Data (*Confidentiality*) 73
4.1.2 Autentikasi Pihak-Pihak Yang Terlibat (*Authentication*) 73
4.1.3 Keutuhan Data (*Integrity*) 75
4.1.4 Interoperability 75
4.2 PKCS #7 76
4.2.1 SignedData 76
4.2.2 EnvelopedData 79
4.2.3 EncryptedData 82
4.2.4 DigestedData 82

4.2.5 ContentInfo 83
4.3 Pemilihan Bahasa Pemrograman 84
4.4 Perancangan Objek 84
4.4.1 Operator Kriptografi 85
4.4.2 Pembungkus Pesan (*message encapsulation*) 85
4.4.3 Komponen Pembentuk Pesan Pembayaran (*payment message component*) 85
4.4.4 Pesan Pembayaran (*payment messages*) 86
4.4.5 Pesan-pesan yang berhubungan dengan sertifikat. 86
4.5 Struktur Lapisan Objek Dalam SET 86

BAB V IMPLEMENTASI 88
5.1 Karakteristik Objek 88
5.1.1 Field Dalam Objek 88
5.1.2 Constructor 89
5.1.3 Method Yang Wajib Dimiliki Setiap *Class* 91
5.2 Karakteristik Objek Operator Kriptografi 93
5.3 Hirarki Objek 95
5.3.1 Paket digsec.set.core.crypto 95
5.3.2 Paket digsec.set.core.pkcs 95
5.3.3 Paket digsec.set.core.encapsulation 96
5.3.4 Paket digsec.set.payment.component 96
5.3.5 Paket digsec.set.payment.message 98
5.3.6 Paket digsec.set.payment.message.wrapper 98
5.3.7 Paket digsec.set.payment.message.error 99

BAB VI UJI COBA 100
6.1 Skenario Pengujian 100
6.2 Hasil Yang Diharapkan 100
6.3 Pengujian 101
6.3.1 Pengujian Objek Operator Kriptografi 101
6.3.2 Pengujian Objek Non Kriptografi 101
6.4 Hasil Pengujian 102
6.4.1 Objek Operator Kriptografi 102
6.4.2 Objek Non Kriptografi 103

BAB VII KESIMPULAN DAN SARAN 106
7.1 Kesimpulan 106
7.2 Saran 107
DAFTAR ACUAN 108

DAFTAR TABEL

Tabel I-1. Pihak yang terlibat dalam skenario 6
Tabel I-2. Contoh beberapa tipe dengan tag *universal*-nya 15
Tabel III-1. Notasi penulisan 31
Tabel III-2. Operator *Hash* 32
Tabel III-3. Operator tanda tangan digital 33
Tabel III-4. Operator enkripsi 33
Tabel III-5. Operator enkapsulasi 34
Tabel III-6. TransIDs 35
Tabel III-7. Payment Instruction 37
Tabel III-8. PIHead 38
Tabel III-8. PIHead (lanjutan) 39
Tabel III-9. InstallRecurData 40
Tabel III-10. AcqCardMsg 41
Tabel III-11. PANData 42
Tabel III-12. PANToken 42
Tabel III-13. Kode kesalahan 46
Tabel III-13. Kode kesalahan (lanjutan) 47
Tabel III-14. PInitReq 49
Tabel III-15. PInitRes 51
Tabel III-15. PInitRes (lanjutan) 52
Tabel III-16. PReq 54
Tabel III-17. PReqDualSigned 55
Tabel III-18. PReqUnsigned 56
Tabel III-18. OIData 57
Tabel III-18. OIData (lanjutan) 58
Tabel III-19. PRes 63
Tabel III-20. PresPayload 65
Tabel III-20. PResPayload (lanjutan) 66
Tabel III-21. Nilai untuk CompletionCode 66
Tabel III-21. InqReq 71
Tabel V-1. Objek-objek pembentuk PIHead 90
Tabel V-2. Paket digsec.set.core.crypto 95
Tabel V-3. Paket digsec.set.core.pkcs 96
Tabel V-4. Paket digsec.set.core.encapsulation 96
Tabel V-5. Paket digsec.set.payment.component 97
Tabel V-6. Paket digsec.set.payment.message 98
Tabel V-7. Paket digsec.set.payment.message.wrapper 99
Tabel V-8. Paket digsec.set.payment.message.error 99
Tabel VI-1. Hasil pengujian objek operator kriptografi 102

Tabel VI-2. Hasil pengujian objek komponen pesan..... 103
Tabel VI-3. Hasil pengujian objek pesan..... 104
Tabel VI-4. Hasil pengujian objek pembungkus pesan..... 105
Tabel VI-5. Hasil pengujian objek pesan kesalahan..... 105

DAFTAR GAMBAR

Gambar I-1. Enkripsi kunci simetris 7
Gambar I-2. Enkripsi kunci asimetris 8
Gambar I-3. Fungsi *hash* satu arah 9
Gambar I-4. Hirarki kepercayaan *Certificate Authority* 13
Gambar III-1. Alur transaksi SET 24
Gambarl III-2. Pesan-pesan pembayaran dalam SET 29
Gambar IV-1. SignedData 79
Gambar IV-2. EnvelopedData 81
Gambar IV-3. EncryptedData 82
Gambar IV-4. DigestedData 83
Gambar IV-6. Struktur lapisan objek 87

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG MASALAH

Pemakaian Internet semakin berkembang dari penggunannya sebagai sarana komunikasi global menjadi sarana perdagangan. Suatu perusahaan penjual barang atau jasa tidak cukup hanya menempatkan informasi mengenai profil perusahaan saja, tetapi juga harus bisa menyediakan layanan pembelian produknya melalui Internet [Hof 99].

Perkembangan perdagangan di Internet sudah mencapai angka jutaan dollar amerika untuk saat ini, dan akan berkembang lagi hingga mencapai milyaran dollar untuk tahun 2001. Menurut hasil peneltian yang dilakukan Forrester Research Inc., penjualan produk Hardware dan Software PC lewat Internet pada tahun 1997 adalah sebesar 863 juta dollar dan diperkirakan akan mecapai 3,8 milyar dollar pada tahun 2001. Penjualan produk buku dan musik pada tahun 1997 sebesar 156 juta dollar dan akan mencapai 1,1 milyar dollar pada tahun 2001 [HoMS 98].

Seiring dengan perkembangan perdagangan di Internet tersebut, diperlukan protokol perdagangan yang aman di Internet. Seperti sudah diketahui, Internet adalah jalur komunikasi publik yang terbuka dan sangat tidak aman. Ada dua protokol yang saat ini dianggap paling aman dan cukup luas pemakaiannya, yaitu *Secure Socket Layer (SSL)* dan *Secure Electronic Transaction (SET)*.

Kedua protokol tersebut sama-sama menggunakan konsep kriptografi, yaitu penyandian pesan sehingga hanya pihak penerima pesan yang berhak yang dapat membaca pesan yang dikirimkan. Kelebihan SET dibanding SSL adalah penggunaan sertifikat digital untuk mengidentifikasi pihak-pihak yang terlibat dalam proses perdagangan. Dengan adanya sertifikat digital, pihak pembeli yakin bahwa pihak penjual adalah benar-benar ada dan tidak dapat dipalsukan oleh pihak lain. Pihak penjual juga yakin bahwa pihak pembeli benar-benar ada dan akan membayar produk yang dibelinya. Kelebihan lainnya adalah, pada SET, nomor kartu kredit pembeli tidak diketahui oleh penjual, tetapi langsung dikirim ke bank penerbit

kartu kredit untuk diotorisasi. Hal ini dapat mencegah penyalahgunaan nomor kartu kredit oleh penjual.

Sejak April 1997 program SET telah mulai dilaksanakan di beberapa negara termasuk Amerika, Australia, Malaysia, Hong Kong. Di Jepang, program SET dilaksanakan sejak Oktober 1997, dan diberi nama *Smart Commerce Japan*. Di Singapura, *National Computer Board (NCB)* memprakarsai suatu program bernama *Electronic Commerce Hotbed (ECH)* yang bertujuan untuk mempercepat pelaksanaan transaksi secara online dan menjadikan Singapura sebagai salah satu pemimpin dalam bidang tersebut. Di Eropa, Bank Nasional Irlandia yang mempelopori pemakaian SET sejak musim semi 1997 untuk transaksi secara online. Kemudian diikuti oleh Spanyol dan Finlandia lalu Austria, Swedia, Norwegia, Denmark, Italia, Jerman, Portugal, dan Prancis. Total ada 47 lembaga keuangan Visa di 18 negara yang mendukung pemakaian SET di Eropa [Visa 99].

Sementara itu di Indonesia hingga saat ini belum ada satupun lembaga yang menerapkan SET sebagai sarana transaksi secara online.

1.2 TUJUAN PENELITIAN

Tujuan dilakukannya penelitian ini adalah analisa protokol komunikasi antara *Cardholder* dan *Merchant* pada protokol SET dan implementasi objek-objek pendukungnya.

1.3 PEMBATASAN MASALAH

- Karena protokol SET sangat kompleks, maka penelitian hanya dilakukan pada komunikasi antara *Cardholder* dengan *Merchant.*
- Penelitian dititikberatkan pada implementasi objek-objek yang dipakai dalam komunikasi antara *Cardholder* dan *Merchant.*
- Untuk perangkat-perangkat kriptografi yang diperlukan dalam SET, peneliti tidak membuatnya sendiri, melainkan memakai perangkat kriptografi yang sudah dibuat oleh pihak lain.
- Masalah pembuatan sertifikat digital tidak masuk dalam penelitian. Dalam laporan penelitian, hanya ditulis latar belakang sertifikat digital.
- Ekstensi setiap pesan dan komponen pembentuk pesan SET tidak diimplementasikan,

karena tergantung dari kebijakan penerbit kartu pembayaran.

1.4 METODOLOGI PENELITIAN

Penelitian dimulai dengan mempelajari protokol SET yang didokumentasikan dalam tiga buku dokumen SET yang dikeluarkan oleh Visa dan Mastercard. Buku pertama berisi kebutuhan bisnis SET, buku kedua berisi petunjuk untuk pemrogram, dan buku ketiga berisi spesifikasi formal SET dalam format *ASN.1*.

Kemudian penelitian diteruskan dengan mempelajari perangkat-perangkat kriptografi yang dibutuhkan pada SET. Karena spesifikasi formal SET ditulis dalam format *ASN.1* dan pengiriman data antar pihak yang terkait adalah dalam format *DER Encoding*, maka penelitian dilanjutkan dengan mempelajari format ASN.1 dan pengkodean data menggunakan format DER.

Langkah selanjutnya adalah merancang dan membuat operator kriptografi sesuai dengan spesifikasi yang terdapat dalam SET. Operator kriptografi tersebut sebagai dasar dari pembentukan pesan-pesan yang digunakan dalam SET. Kemudian dilanjutkan dengan membuat komponen-komponen pesan SET (*message component*), pembungkus pesan (*message wrapper*), dan pesan-pesan yang digunakan dalam komunikasi antara *Cardholder* dan *Merchant Server*.

Kemudian objek-objek yang dihasilkan diuji untuk memverifikasi kebenarannya menggunakan metoda yang dirancang oleh peneliti.

1.5 SISTEMATIKA PENULISAN

Bab pertama diawali dengan penjelasan mengenai latar belakang masalah, kemudian disambung dengan tujuan penelitian, ruang lingkup permasalahan dan metoda penelitian.

Bab 2 membahas landasan teori kriptografi, sertifikat digital, penulisan spesifikasi dalam format *ASN.1*, dan pengkodean data dalam format DER.

Selanjutnya bab 3 membahas mengenai spesifikasi SET dan analisa spesifikasi tersebut, sesuai dengan yang terdapat dalam dokumentasi SET.

Lalu bab 4 menjabarkan perancangan dan analisa komponen-komponen pembentuk

pesan-pesan SET, dan pesan-pesan SET.

Bab 5 berisi implementasi objek-objek yang dipakai dalam protokol komunikasi antara *Cardholder* dan *Merchant.*

Bab 6 berisi uji coba untuk objek-objek tersebut.

Bab terakhir akan diisi dengan kesimpulan dan saran dari penelitian yang penulis lakukan.

BAB II
LANDASAN TEORI

Dalam bab ini akan dibahas mengenai landasan teori yang perlu dikuasai untuk mengimplementasikan SET. Bagian pertama membahas mengenai kriptografi, dan bagian selanjutnya mengenai pendefinisian format pesan menggunakan *Abstract Syntax Notation (ASN.1)* dan pengkodeannya menggunakan aturan DER.

Kriptografi adalah suatu ilmu yang mempelajari bagaimana membuat suatu pesan menjadi aman selama pengiriman dari pengirim (*sender*) sampai ke penerima (*receiver*) [Schn96]. Pesan tersebut disebut *plaintext*, proses untuk mengubah *plaintext* menjadi suatu bentuk yang tidak dapat dibaca isinya disebut enkripsi. Pesan yang terenkrip disebut *ciphertext*. Proses untuk mengubah *ciphertext* ke pesan aslinya (*plaintext*) disebut dekripsi. Hubungan antara *plaintext*, *ciphertext*, enkripsi dan dekripsi dapat ditulis dalam bentuk sebagai berikut:

C = E (M) dimana:

C = *ciphertext*

E = proses enkripsi

M = *plaintext*

M = D (C) dimana:

C = *ciphertext*

D = proses dekripsi

M = *plaintext*

Seringkali fungsi-fungsi enkripsi dan dekripsi mempunyai satu parameter tambahan, yaitu kunci. Kunci diperlukan untuk mendapatkan *ciphertext* melalui proses enkripsi dan untuk mengembalikannya ke bentuk asli (*plaintext*) melalui proses dekripsi. Kunci yang dipakai untuk enkripsi bisa sama dengan kunci yang diperlukan untuk dekripsi, tetapi bisa juga berbeda.

Untuk memudahkan ilustrasi dari penjelasan selanjutnya mengenai kriptografi, maka digunakanlah istilah-istilah berikut sebagai pihak-pihak yang terlibat dalam skenario komunikasi yang menggunakan kriptografi.

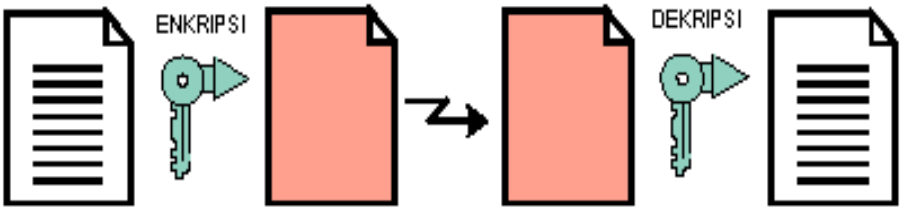
Kode & nama	**Keterangan**
A: Anna	Pihak pertama
B: Budi	Pihak kedua
C: Chris	Pihak ketiga
E: Elsa	Pihak penyadap informasi yang tidak diperuntukkan kepadanya
M: Mamad	Pihak yang tidak hanya menyadap informasi, namun juga mengubah informasi yang disadap

Tabel I-1. Pihak yang terlibat dalam skenario

Berikutnya akan dibahas mengenai beberapa teknik kriptografi yang diperlukan dalam implementasi SET.

2.1 ENKRIPSI SIMETRIS

Teknik kriptografi yang menggunakan kunci simetris adalah yang paling umum dipergunakan. Kunci untuk melakukan proses enkripsi sama dengan kunci untuk melakukan proses dekripsi. Jadi misalkan Anna ingin mengenkrip suatu pesan dan mengirimkannya ke Budi, maka baik Anna maupun Budi harus mempunyai kunci yang sama persis. Siapapun yang memiliki kunci tersebut – termasuk pihak-pihak yang tidak diinginkan – bisa mendapatkan pesan aslinya dari *ciphertext.* Problem yang paling signifikan disini adalah bukan pada masalah pengiriman *ciphertext*-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Teknik kriptografi menggunakan kunci simetri seringkali disebut juga sebagai *secret key cryptography.* Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*) dan RC-4.



Gambar I-1. Enkripsi kunci simetris

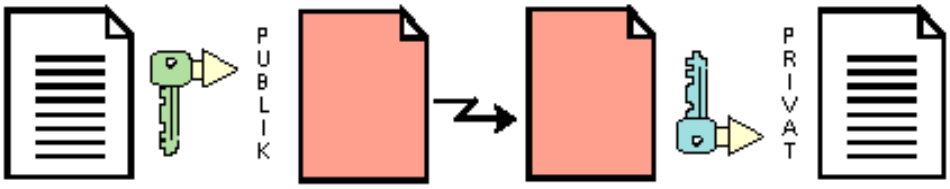
2.2 ENKRIPSI ASIMETRIS

Jika teknik kriptografi menggunakan kunci simetris, memakai kunci yang sama untuk melakukan proses enkripsi dan dekripsi, maka teknik kriptografi menggunakan kunci asimetris memerlukan sepasang kunci untuk enkripsi dan dekripsi. Pesan yang dienkrip menggunakan sebuah kunci hanya bisa dibuka menggunakan kunci pasangannya. Pesan tersebut tidak bisa dibuka menggunakan kunci yang sama.

Kunci yang pertama disebut kunci publik dan kunci pasangannya disebut kunci privat. Jadi sebuah pesan yang dienkrip menggunakan kunci publik hanya bisa dibuka menggunakan kunci privat, dan demikian pula sebaliknya. Proses enkripsi atau dekripsi tersebut hanya bisa dilakukan menggunakan pasangan kunci yang tepat, jika pasangan kuncinya salah, maka proses enkripsi atau dekripsi akan gagal. Kunci publik dapat diketahui oleh semua orang sedangkan kunci privat hanya boleh diketahui oleh satu orang saja, yaitu orang yang berhak memilikinya.

Dengan demikian maka Chris dapat yakin bahwa hanya Anna yang dapat membuka pesan yang dikirim oleh Chris kepada Anna, jika Chris mengenkrip pesan tersebut menggunakan kunci publik milik Anna. Demikian juga Budi dapat yakin bahwa pengirim pesan adalah benar-benar si Chris, jika Chris mengenkrip pesan tersebut menggunakan kunci privat miliknya sendiri, karena hanya kunci publik milik Chris yang dapat membuka pesan tersebut, dengan asumsi bahwa kunci publik yang dipakai adalah benar-benar milik Chris.

Masalah yang timbul adalah bagaimana caranya agar Anna bisa mendapatkan kunci publik Chris. Masalah tersebut timbul karena Anna harus benar-benar yakin bahwa kunci publik yang didapat adalah benar-benar milik Chris.



Gambar I-2. Enkripsi kunci asimetris

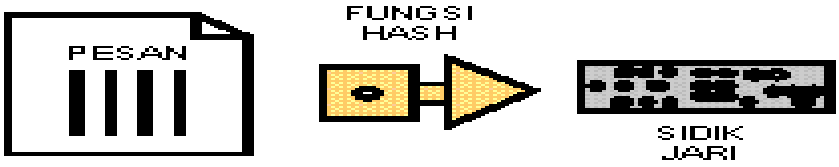
Contoh teknik enkripsi asimetris yang terkenal adalah RSA (merupakan singkatan penemunya yakni Rivest, Shamir dan Adleman).

2.3 FUNGSI HASH SATU ARAH

Sekarang coba perhatikan ilustrasi berikut ini. Misalkan Budi mengirimkan perintah pembayaran kepada bank untuk membayarkan sejumlah Rp. 200.000, 00 kepada Anna. Pesan tersebut di tengah jalan berhasil dibobol oleh Mamad yang entah bagaimana caranya berhasil mendapatkan pesan aslinya. Kemudian Mamad mengubah isi pesan pembayaran tersebut menjadi Rp. 2.000.000,00. Pihak bank tidak akan tahu bahwa pesan telah diubah selama perjalanan. Untuk itulah perlu ada suatu mekanisme agar pihak yang menerima pesan dapat yakin bahwa pesan tersebut tidak mengalami perubahan selama perjalanan, baik perubahan karena ada pihak tertentu yang mengubahnya maupun perubahan karena adanya kesalahan transmisi.

Masalah tersebut bisa diatasi menggunakan fungsi *hash* satu arah (*one way hash function*), yang terkadang disebut juga *message digest* atau penyandian searah. Sebuah pesan yang besarnya bervariasi yang akan menjadi input bagi fungsi hash satu arah, disebut *pre-image*, kemudian hasilnya adalah suatu nilai yang panjangnya tetap dan disebut nilai hash (*hash-value),* sidik jari (*fingerprint*), atau *hash*.

Bagaimanakah fungsi *hash* satu arah ini bisa memecahkan masalah tadi? Misalkan Budi ingin mengirimkan sebuah pesan kepada Anna. Pertama-tama Budi mendapatkan nilai hash dari pesan yang akan dikirimkannya tersebut. Kemudian pesan bersama nilai hash tersebut dikirimkan kepada Anna. Ketika Anna menerima pesan tersebut, dia membuat suatu nilai hash yang baru dari pesan yang dikirimkan Budi, kemudian Anna membandingkan nilai hash yang baru dia buat dengan nilai hash yang dikirimkan oleh Budi. Jika nilai hash tersebut sama, maka Anna dapat yakin bahwa pesan Budi tersebut tidak mengalami perubahan selama perjalanan.



Gambar I-3. Fungsi *hash* satu arah

Fungsi hash satu arah ini berangkat dari asumsi bahwa hampir tidak ada dua *pre-image* yang mempunyai nilai hash yang sama, atau bisa dibilang sangat kecil kemungkinannya. Dan asumsi berikutnya adalah sangat sulit atau hampir tidak mungkin untuk mendapatkan *pre-image* dari suatu nilai hash.

Fungsi hash untuk membuat sidik jari tersebut dapat diketahui oleh siapapun dan dapat dipakai oleh siapapun. Algoritmanya terbuka sehingga bisa diimplementasikan oleh siapapun.

Contoh fungsi hash yang terkenal adalah MD-5 dan SHA.

2.4 TANDA TANGAN DIGITAL

Dengan adanya fungsi *hash* satu arah, keutuhan selama pengiriman data dapat dijamin. Namun bagaimana jika terjadi hal berikut: Chris mengirimkan pesan kepada Anna, dan menjamin keutuhan isi pesan tersebut menggunakan fungsi *hash* satu arah. Anna dapat yakin bahwa pesan tersebut tidak mengalami perubahan sebagian isinya selama perjalanan. Tapi bagaimana jika Mamad mengganti keseluruhan pesan tersebut dan membuat nilai *hash* dari pesan palsu tersebut dan mengirimkannya ke Anna. Anna tetap yakin bahwa pesan tersebut tidak mengalami perubahan isi namun dia tidak tahu kalau ternyata pesan tersebut bukan pesan asli dari Chris, melainkan pesan yang dibuat oleh Mamad.

Untuk mengatasi hal tersebut, digunakanlah teknik tanda tangan digital. Sama halnya dengan fungsi tanda tangan yang sesungguhnya, tanda tangan digital digunakan untuk menjamin keaslian dan keutuhan data yang ditandatangani.

Hal-hal yang diharapkan dari tanda tangan digital adalah:

1. Tanda tangan digital dapat menjamin keaslian pesan dan tidak mudah dipalsukan sehingga pihak yang menandatangani tidak dapat menyangkal bahwa dia pernah

menandatangani pesan tersebut.

2. Tanda tangan tersebut mudah untuk diperiksa kebenarannya dan dapat diperiksa oleh pihak yang belum pernah bertemu dengan pihak yang menandatanganinya.
3. Tanda tangan tersebut hanya berlaku untuk pesan yang ditandatangani tersebut.

Cara membuat tanda tangan digital adalah sebagai berikut:

1. Suatu pesan dibuat nilai *hash*-nya menggunakan fungsi *hash* satu arah.
2. Kemudian nilai *hash* tadi dienkrip menggunakan kunci privat milik pihak yang menandatangani, hasilnya adalah tanda tangan digital.

Kemudian tanda tangan digital tersebut diverifikasi dengan cara:

1. Tanda tangan digital didekrip menggunakan kunci publik penandatangan, didapatkan suatu nilai *hash*, misalkan disebut h1.
2. Suatu nilai *hash* yang baru dibuat dari pesan tersebut, misalkan disebut h2.
3. h1 dan h2 dibandingkan, jika sama, maka dapat dijamin bahwa tanda tangan digital tersebut sah untuk pesan tersebut dan penandatangannya.

Dapat dilihat bahwa teknik tanda tangan digital adalah gabungan dari fungsi *hash* satu arah dan enkripsi kunci asimetris.

2.5 TANDA TANGAN GANDA

Misalkan Budi dan Chris mengadakan perjanjian jual beli. Budi membuat dua dokumen, yang pertama berisi isi perjanjian jual beli dan yang kedua berisi perintah pembayaran dari Bank kepada Chris. Budi tidak ingin agar Bank dapat mengetahui isi perjanjian jual beli, tapi harus dipastikan kalau perintah pembayaran tersebut sesuai dengan isi perjanjian jual beli.

Mekanisme untuk memecahkan masalah tersebut adalah teknik tanda tangan ganda. Caranya adalah:

1) Budi membuat *hash* dari kedua dokumen tersebut: h1 untuk dokumen perjanjjian jual beli dan h2 untuk dokumen perintah pembayaran
2) Kemudian Budi membuat *hash* dari gabungan h1 dan h2 menjadi h3
3) Lalu Budi mengenkrip h3 menggunakan kunci privat miliknya, jadilah tanda tangan ganda

4) Budi mengirim dokumen perjanjian jual beli kepada Chris bersama dengan tanda tangan ganda dan h2. Chris memeriksa tanda tangan ganda dengan cara :
   a) Mendekrip tanda tangan ganda menggunakan kunci publik milik Budi untuk mendapatkan h3
   b) Kemudian Chris membuat *hash* dari dokumen perjanjian jual beli untuk mendapatkan h1
   c) Lalu h1 digabung dengan h2 yang dikirim bersama dengan pesan, bandingkan dengan h3, jika sama, maka verifikasi tanda tangan ganda telah berhasil.
5) Budi mengirim dokumen perintah pembayaran kepada Bank bersama dengan tanda tangan ganda dan h1. Bank memeriksa tanda tangan ganda dengan cara yang mirip dengan yang dilakukan Chris. Perbedaannya adalah Bank membuat *hash* dari dokumen perintah pembayaran untuk mendapatkan h2, lalu menggabungkannya dengan h1 yang dikirim bersama pesan, dan membandingkannya dengan h3.

2.6 AMPLOP DIGITAL

Pemakaian teknik enkripsi asimetris dapat menjamin bahwa hanya orang yang berhak yang dapat membaca isi pesan yang dikirim. Namun pemakaian enkripsi asimetris tidak efisien untuk pesan yang panjang, karena waktu komputasi untuk melakukan enkripsi asimetris jauh lebih lambat daripada waktu komputasi untuk melakukan enkripsi simetris [Schn96].

Pemecahannya adalah menggunakan teknik amplop digital, yaitu pesan yang akan dikirim dienkrip dahulu menggunakan sebuah kunci simetris, kemudian kunci simetris tersebut dienkrip menggunakan kunci publik milik penerima pesan. Ukuran kunci simetris yang dipakai lebih kecil daripada ukuran pesan, oleh karena itu waktu komputasi untuk mengenkrip kunci tersebut secara asimetris lebih cepat daripada mengenkrip pesan itu sendiri.

Jika Anna ingin mengirim pesan kepada Budi, maka Anna membuat suatu kunci simetris dan mengnekrip kunci tersebut menggunakan kunci publik Budi. Kemudian Anna mengirimkan pesan bersama kunci yang terenkrip tadi kepada Budi. Ketika Budi menerima pesan terenkrip tersebut, dia mendapatkan kembali kunci simetris dengan cara mendekrip

kunci yang terenkrip menggunakan kunci privat miliknya. Kunci simetris yang didapat digunakan untuk mendekrip pesan, dan akhirnya pesan asli dapat dibaca oleh Budi.

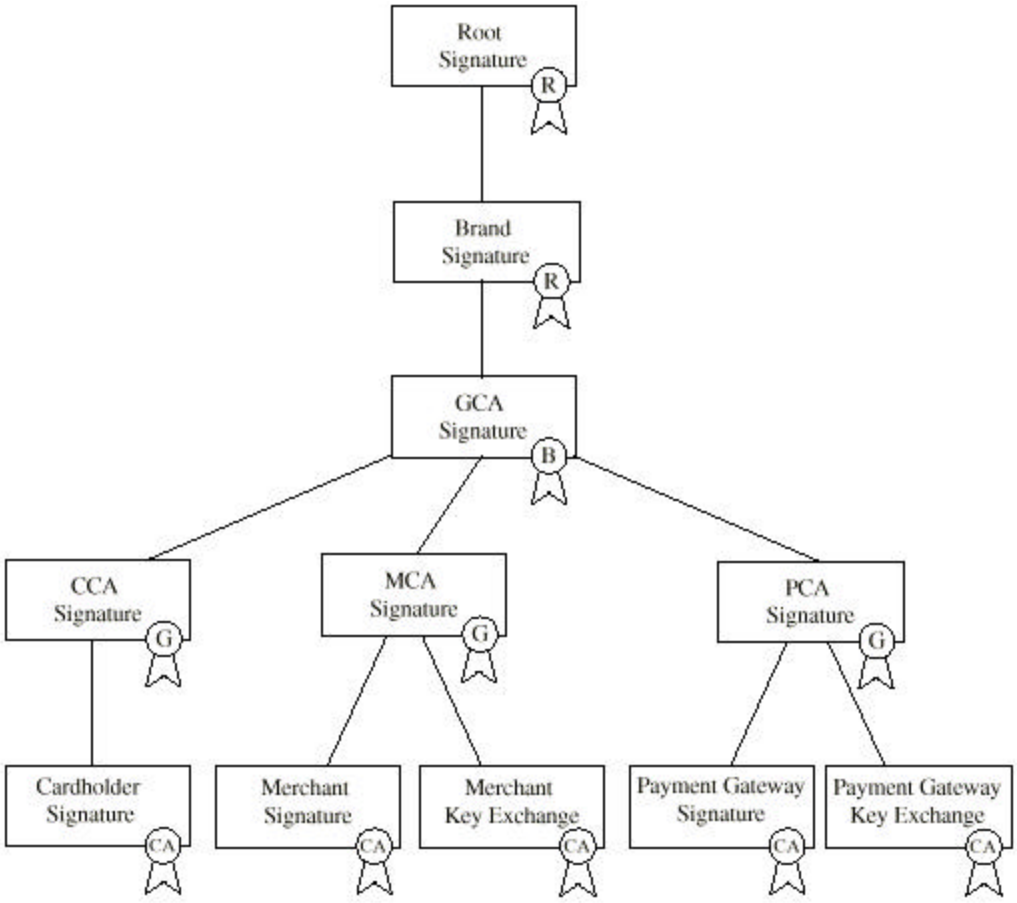
2.7 SERTIFIKAT DIGITAL

Sertifikat digital digunakan sebagai penanda jati diri suatu pihak yang melakukan komunikasi dengan pihak lain. Fungsinya sama dengan kartu identitas pada dunia nyata. Seperti halnya kartu identitas, maka pada sertifikat digital terdapat tanda tangan pemilik sertifikat, tanda tangan suatu badan yang menyatakan bahwa sertifikat tersebut masih berlaku, dan ada masa berlaku dari sertifikat tersebut.

Di dalam sertifikat digital disertakan kunci publik milik si pemegang sertifikat. Hal ini dapat menyelesaikan masalah bagaimana cara mendapatkan kunci publik milik seseorang seperti yang disebutkan sebelumnya dalam pembahasan mengenai enkripsi asimetrik.

Di dalam sertifikat digital juga diberikan suatu mekanisme yang mengasosiasikan informasi kartu pembayaran dengan sertifikat digital tersebut. Caranya adalah dengan menyertakan nilai *hash* dari informasi kartu pembayaran di dalam sertifikat digital.

Badan yang mengesahkan suatu sertifikat disebut *Certificate Authority (CA)*. Suatu CA bertugas untuk mengesahkan dan menerbitkan sertifikat digital dari suatu permohonan sertifikat digital. Sama juga halnya dengan badan yang mengesahkan kartu identitas, maka CA sebagai badan yang mengesahkan sertifikat mempunyai jenjang kedudukan. Masing-masing CA tersebut mempunyai sertifikat yang disahkan oleh CA di atasnya. Untuk lebih jelasnya dapat dilihat dari gambar berikut:



Gambar I-4. Hirarki kepercayaan *Certificate Authority*

CCA (*Cardholder Certificate Authority*) adalah CA yang menjamin sertifikat *Cardholder*, MCA (*Merchant Certificate Authority*) yang menjamin sertifikat milik *Merchant* dan PCA(*Payment Gateway Certificate Authority*) yang menjamin sertifikat milik *Payment Gateway*). Ketiga CA tadi (CCA, MCA, dan PCA) disahkan oleh GCA (*Geopolitical Certificate Authority*), dan demikian seterusnya sampai ke *Root.*

Jika suatu pihak ingin memvalidasi suatu sertifikat, maka dia melakukan penelusuran hirarki kepercayaan sertifikat sampai ke CA yang dia percaya. Penelusuran tersebut bisa saja sampai satu tingkat di atasnya atau bahkan harus sampai ke *Root.* Hal ini tergantung dari CA mana yang dia percaya.

Setiap CA mempunyai suatu daftar sertifikat-sertifikat yang sudah tidak berlaku (karena habis masa berlakunya atau karena pemilik sertifikat pernah berbuat curang, sehingga tidak dipercaya lagi). Daftar tersebut dinamakan CRL (*Certificate Revocation List*). CRL

berguna untuk memudahkan pengecekan sertifikat. Jika suatu sertifikat terdapat di dalam CRL, maka dapat dipastikan bahwa sertifikat tersebut tidak sah dan identitas pengirim sertifikat tidak bisa dipercaya lagi. Dengan demikian penelusuran hirarki kepercayaan tidak perlu diteruskan sampai ke tingkat yang lebih tinggi..

2.8 ASN.1 DAN DER

Abstraksi memungkinkan seorang perancang sistem menspesifikasikan bagian sistemnya tanpa memperdulikan bagaimana bagian tersebut diimplementasikan. Dia hanya menyebutkan spesifikasi dari bagian yang dia rancang dan implementasinya menjadi terbuka.

Open System Interconnection (OSI) memakai ASN.1 untuk menulis objek abstrak dan memakai BER untuk merepresentasikan objek tersebut ke dalam suatu rangkaian bit 0 dan 1. DER adalah metode pengkodean yang merupakan *subset* dari BER dan lebih spesifik lagi dalam pengkodean dibanding BER.

2.8.1 ASN.1

ASN.1 adalah sebuah bahasa formal untuk mendeskripsikan format suatu objek secara abstrak. ASN.1 dipakai untuk menuliskan tipe data abstrak dan nilainya. Dalam ASN.1 terdapat 4 tipe: tipe sederhana, yaitu tipe yang tidak mempunyai komponen, tipe terstrukur, yaitu tipe data yang mempunyai komponen, tipe turunan, yaitu tipe yang diturunkan dari tipe lainnya, serta yang terakhir adalah tipe lainnya, yaitu tipe CHOICE dan ANY. Tipe dan nilai bisa dinamakan dengan operator *assignment* (::=), dan nama tadi bisa digunakan untuk mendefinisikan tipe dan nilai lain.

Setiap tipe dalam ASN.1 selain CHOICE dan ANY, mempunyai sebuah tag yang terdiri dari sebuah *class* dan sebuah bilangan bulat non negatif. Dua tipe dalam ASN.1 dikatakan sama jika keduanya mempunyai tag yang sama, walaupun namanya berbeda. Ada 4 tag yang didefinisikan dalam ASN.1 :

1) Tipe *Universal,* untuk tipe yang mempunyai arti yang sama untuk semua aplikasi
2) Tipe *Application*, untuk tipe yang mempunyai arti yang spesifik hanya untuk aplikasi tertentu
3) Tipe *Private*, untuk tipe yang mempunyai arti yang spesifik hanya untuk perusahaan yang memberikannya

4) Tipe *Context-spesific*, untuk tipe yang artinya spesifik hanya untuk struktur tertentu. Tag context-specific digunakan untuk membedakan antara tipe komponen yang memiliki tag dasar yang sama di dalam konteks tipe terstruktur yang diberikan. Tipe komponen dalam dua tipe terstruktur yang berbeda bisa memiliki tag sama dengan arti berbeda.

Berikut ini diberikan contoh untuk beberapa tipe dan tag *universal*-nya:

Tipe	Nomor Tag (desimal)	Nomor Tag (heksadesimal)
BOOLEAN	1	1
INTEGER	2	2
BIT STRING	3	3
OCTET STRING	4	4
NULL	5	5
OBJECT IDENTIFIER	6	6
REAL	9	9
ENUMERATED	10	A
SEQUENCE (OF)	16	10
SET (OF)	17	11
NUMERIC STRING	18	12
PRINTABLE STRING	19	13
UTCTime	23	17
GENERALIZED TIME	24	18
VISIBLE STRING	26	1A
BMP STRING	30	1E

Tabel I-2. Contoh beberapa tipe dengan tag *universal*-nya

Tipe dan nilai ASN.1 ditulis dalam notasi yang fleksible dan mirip seperti bahasa pemrograman, dengan aturan khusus:

- Layout tidak signifikan, multi spasi dan baris baru bisa dianggap sebagai satu spasi.
- Komentar dibatasi dengan pasangan hipenasi (–), atau sepasang hipenasi dan baris baru.
- Identifier (untuk nama nilai dan field) dan referensi tipe (nama tipe) terdiri dari huruf besar dan huruf kecil, angka, hipenasi, dan spasi. Identifier diawali dengan huruf kecil, tipe referensi diawali dengan huruf besar.

a. Tipe Sederhana

Tipe sederhana adalah tipe yang tidak mempunyai komponen lagi di dalamnya. Tipe ini adalah tipe yang paling dasar dan tidak dibentuk dari tipe-tipe lainnya. Tipe yang

termasuk tipe ini adalah:

- BOOLEAN, nilainya hanya benar atau salah.
- INTEGER, bilangan bulat.
- BIT STRING, string yang terdiri dari bit (0 dan 1).
- OCTET STRING, string yang terdiri dari oktet (nilai 8-bit).
- NULL , nilai null.
- OBJECT IDENTIFIER, pengenal suatu obyek, terdiri dari kumpulan bilangan bulat berurutan yang mengidientifikasi suatu obyek seperti algoritma atau tipe atribut.
- PrintableString, string karakter yang bisa dicetak.
- T61String, string yang terdari dari karakter T.61 (8-bit).
- IA5String, string yang terdiri dari karakter IA5 (ASCII).
- NumericString, string yang terdiri dari karakter bilangan (0 sampai 9).
- UTCTime, waktu Greenwich Mean Time (GMT).
- GeneralizedTime, sama seperti UTCTime, tapi menyimpan format tahun dalam format empat digit.

b. Tipe Terstruktur

Tipe ini adalah tipe-tipe yang mempunyai komponen di dalamnya, yang termasuk dalam tipe ini:

- SEQUENCE, himpunan terurut dari satu atau lebih tipe.
- SEQUENCE OF, himpunan terurut dari nol atau lebih sebuah tipe tertentu.
- SET, himpunan tidak terurut dari satu atau lebih tipe.
- SET OF, himpunan tidak terurut dari nol atau lebih satu tipe tertentu.

c. Tipe Turunan

Tipe turunan adalah tipe yang diperoleh dari tipe tertentu dengan menambahkan atau mengganti *tag* dari tipe tersebut. Tujuannya adalah untuk mencegah kebingungan apabila suatu tipe terstruktur tersusun atas beberapa komponen setipe namun semuanya bersifat *optional.*

Ada dua macam tipe turunan, yaitu : IMPLICIT dan EXPLICIT. Tipe dengan *tag* EXPLICIT diambil dari tipe lain dengan menambahkan satu *tag* luar pada tipe yang dimaksud. Hasilnya, tipe dengan *tag* EXPLICIT adalah tipe terstruktur yang terdiri atas satu komponen, tipe di dalamnya. Notasi *tag* EXPLICIT adalah kata kunci ASN.1 [nomor

kelas] EXPLICIT.

Tipe dengan *tag* IMPLICIT dianggap sama dengan tipe yang disimpan, tapi *tag*-nya berbeda. Tag IMPLICIT didapatkan dengan mengganti *tag* tipe yang diturunkan dengan *tag* IMPLICIT.

d. Tipe Lainnya

Tipe lain dalam ASN.1 mencakup tipe CHOICE dan ANY. Tipe CHOICE menyatakan perpaduan satu atau lebih alternatif, tipe ANY menyatakan nilai tidak tentu dari tipe tidak tentu, dengan tipenya kemungkinan didefinisikan pada registrasi suatu *object identifier* atau nilai *integer*.

2.8.2 DER

DER digunakan untuk mengkodekan nilai yang ditulis dalam ASN.1 ke dalam suatu rangkaian *byte* secara spesifik. Pengkodean dalam format DER menghasilkan hanya satu macam rangkaian *byte*.

Dalam setiap rangkaian *byte* dari DER, terdapat tiga bagian:

- *Identifier* (I), menunjukkan kelas dan nomor *tag* dari nilai ASN.1 yang dikodekan, juga menunjukkan apakah tipe tersebut primitif atau terkonstruksi.
- *Length* (L), menunjukkan jumlah *byte* yang terdapat dalam rangkaian *byte*
- *Contents* (C), berisi representasi rangkaian *byte* dari nilai yang dikodekan. Untuk tipe primitif, bagian ini berisi berisi representasi rangkaian *byte* dari nilai tersebut. Untuk tipe terkonstruksi, bagian ini berisi gabungan dari representasi DER tipe-tipe primitif yang menjadi pembentuknya.

Cara pengkodean setiap bagian tersebut :

1) *Identifier*

   Ada dua macam pengkodean, yaitu untuk *tag* kecil (<31) dan *tag* besar (>31).

   Untuk *tag* kecil:

   Identifier untuk *tag* berukuran kecil hanya memerlukan 1 *byte*. Aturannya adalah bit ke-7 dan ke-8 mendefinisikan kelas, bit ke-6 mendefinisikan tipe sederhana atau terstruktur, dan bit ke-5 sampai ke-1 menunjukkan nomor *tag*. Kombinasi bit ke-8 dan ke-7 menunjukkan tipe kelas sebagai berikut: *Universal* : 00, *Application* : 01, *Context Specific* : 10, dan *Private* : 11. Aturan untuk bit ke-6 adalah: 0 untuk tipe

sederhana dan 1 untuk tipe terkonstruksi. Misalnya untuk tipe INTEGER mempunyai *identifier* 00000010 atau 02 dalam heksadesimal, karena INTEGER mempunyai tipe *Universal*, sederhana, dan nomor *tag* 2.

Untuk *tag* besar:

Aturan untuk *tag* besar adalah: pada bit ke-8 sampai bit ke-6 sama dengan aturan pada *tag* kecil. Sedangkan pada bit ke-5 sampai ke-1 berisi angka 1. Untuk menunjukkan nomor *tag* digunakan oktet tambahan yang jumlahnya tidak terbatas. Aturan pada oktet tambahan tersebut adalah: bit ke-8 berisi 1 jika bukan merupakan oktet terakhir yang menjadi penunjuk nomor *tag*, dan berisi 0 jika merupakan oktet terakhir. Bit ke-7 sampai dengan ke-1 menunjukkan nomor *tag*. Contohnya pada tipe yang mempunyai nomor *tag* 16383, maka *identifier* dari tipe tersebut adalah `01111111 11111111 01111111.`

2) *Length*

Ada dua macam aturan untuk menunjukkan panjang dari suatu tipe, yaitu bentuk pendek (<=127) dan bentuk panjang (>127).

Untuk bentuk pendek:

bit ke-8 berisi 0 dan bit sisanya menunjukkan panjang dari tipe tersebut.

Untuk bentuk panjang:

pada oktet pertama, bit ke-8 berisi 0, dan bit-bit sisanya menunjukkan jumlah oktet yang dipakai untuk menunjukkan panjang tipe.

3) *Content*

Untuk mengkodekan nilai dari suatu tipe dapat dilihat dari beberapa nilai berikut:

- Null selalu dikodekan sebagai `05 00` (dalam heksa desimal)

  Berarti I=05, L=00, dan C=kosong
- BOOLEAN, terdiri dari dua macam, yaitu *True* dan *False*. *True* dikodekan sebagai `01 01 FF` (I=01, L=01, C=FF) dan *Flase* dikodekan sebagai `01 01 00` (I=01, L=01, C=00).
- OCTET STRING, dikodekan sesuai dengan nilainya, dan ditambahkan I dan L didepannya. Misalnya untuk suatu OCTET STRING 03 AB 24, akan dikodekan sebagai 04 03 03 AB 24. I berisi 04 (*tag* untuk OCTET STRING), L=3, dan C=03 AB 24.

- STRING

  Beberapa tipe dapat dikategorikan sebagai tipe *string*.yaitu: IA5String, PrintableString dan BIT STRING.

  Pengkodean untuk tipe-tipe *string* ini adalah: I diisi sesuai dengan *tag* masing-masing *string*, L diisi sesuai dengan aturan yang berlaku mengenai panjang, dan C diisi oleh kumpulan nilai ASCII dari untaian *string* tersebut.

  Sebagai contoh suatu *string* "test1" bertipe IA5String dikodekan sebagai `16 05 74 65 73 74 31` (I = 16, L = 05, C = `74 65 73 74 31`).

- SEQUENCE dan SEQUENCE OF

  SEQUENCE dan SEQUENCE OF dikodekan hanya dengan menambahkan *identifier* sesuai dengan *tag* SEQUENCE dan SEQUENCE OF serta panjang dari total gabungan tipe-tipe penyusunnya di depan kumpulan oktet DER hasil pengkodean tipe-tipe penyusunnya.

- SET dan SET OF

  Sama aturannya dengan SEQUENCE, tapi berbeda dalam nomor *tag.*

BAB III
SPESIFIKASI SET

Bab ini membahas mengenai spesifikasi protokol SET dan analisa penulis terhadap spesifikasi tersebut. Pembahasan dimulai dari pengertian mengenai protokol, mengenai kebutuhan bisnis untuk transaksi melalui Internet dan bagaimana SET dapat memenuhi kebutuhan tersebut, kemudian masuk ke dalam pembahasan mengenai SET.

3.1 PROTOKOL

Protokol adalah kumpulan pesan-pesan (*messages*) yang didefinisikan secara jelas dan masing-masing pesan tersebut mempunyai arti dan fungsi yang telah didefiniskan, serta aturan-aturan mengenai kapan pesan tersebut dikirimkan [Larm99]. Protokol adalah kumpulan langkah-langkah yang jelas dan berurutan untuk menyelesaikan suatu pekerjaan antara dua pihak atau lebih [Schn96]. Dari kedua definisi tersebut, dapat dilihat bahwa protokol mempunyai dua hal yang diatur, yaitu:

1. Definisi yang jelas (format) dari pesan
2. Aturan mengenai pembuatan dan pengiriman pesan

SET adalah suatu protokol yang dirancang untuk melakukan transaksi yang aman lewat jalur komunikasi yang terbuka seperti Internet. SET dibuat oleh Visa dan Mastercard, dua badan penerbit kartu kredit yang paling besar. SET didukung pula oleh IBM, Microsoft, GTE, Netscape, Terisa, dan VeriSign. SET melakukan enkripsi data selama komunikasi antara pihak-pihak yang terkait berlangsung. Hal ini untuk menjamin kerahasiaan data dan integritas data selama pengiriman data melalui Internet. SET menggunakan sertifikat digital agar pihak pembeli dan penjual dapat saling mengidentifikasi satu sama lain. SET menyebabkan penjual tidak dapat mengetahui nomor kartu kredit pembeli, sehingga menghilangkan resiko akibat penyalahgunaan nomor kartu kredit oleh penjual.

Karena SET adalah suatu protokol, maka pembahasan SET dapat dilihat dari dua sudut pandang, yaitu proses dan format pesan.

1. *Proses* : di dalam SET didefiniskan secara jelas proses-proses pembentukan pesan dan langkah-langkah pemrosesan pesan tersebut.

2. *Format Pesan* : di dalam SET juga didefiniskan secara jelas dan spesifik format pesan-pesan yang dipakai

Dalam penelitian ini, titik beratnya adalah pada poin kedua, yaitu mengenai format dari pesan-pesan yang dipakai dalam SET dan bagaimana cara-cara pembentukannya.

3.2 KEBUTUHAN BISNIS TRANSAKSI MELALUI INTERNET

Sebelum masuk ke dalam pembahasan protokol SET, terlebih dahulu dibahas mengenai isu-isu yang diperhatikan untuk melakukan transaksi yang aman dan nyaman melalui Internet.

Internet adalah jalur komunikasi publik yang tidak aman. Data-data yang lewat pada jalur tersebut bisa disadap oleh siapa saja dengan keahlian dan peralatan yang memadai. Internet adalah suatu komunitas maya yang bisa diakses oleh hampir semua orang di dunia, tentu saja dengan beberapa infrastrukur yang memadai pula. Kemudian, para pelaku yang berkecimpung di Internet sangat banyak dan menggunakan beragam spesifikasi.

Melihat dari karakteristik Internet yang seperti disebut di atas, bisa terlihat empat kebutuhan bisnis untuk transaksi yang aman di Internet, yaitu:

3.2.1 Confidentiality

Pihak-pihak yang terlibat dalam transaksi tidak ingin data-data sensitif yang dikirim dapat diketahui oleh pihak lain yang tidak ikut terlibat. Data-data sensitif tersebut adalah informasi pembayaran dan nomor kartu pembayaran beserta tanggal kadaluarsanya. Lebih jauh lagi, bahkan diperlukan juga agar informasi kartu pembayaran tidak diketahui oleh pedagang, agar tidak ada penyalahgunaan kartu pembayaran di kemudian hari.

3.2.2 Authentication

Setiap pihak yang terlibat harus yakin akan identitas masing-masing. Pedagang harus yakin bahwa pembeli adalah pemegang yang sah dari kartu pembayaran yang digunakan. Pembeli juga harus yakin gahwa pedagang adalah benar-benar pedagang seperti yang diklaimnya, jadi harus benar-benar dapat menunaikan tugasnya sebagai pedagang, yaitu

memberikan produk yang dijualnya atau menjalankan jasa yang dijanjikannya.

3.2.3 Integrity

Setiap pihak yang terlibat dalam transaksi harus yakin bahwa data yang diterima adalah sama persis dengan data yang dikirm, tanpa ada perubahan sedikitpun selama perjalanan. Keutuhan data tersebut sangatlah vital dalam transaksi di Internet, karena perubahan sedikit saja isi dari informasi pembayaran akan merugikan salah satu atau kedua pihak yang terlibat (pedagang dan pembeli). Misalnya pembeli jadi harus membayar lebih dari yang sudah disepakati, atau pedagang mendapat kurang dari yang sudah disepakati.

3.2.4 Interoperability

Kebutuhan yang satu ini adalah mengenai dapat atau tidaknya sistem pembayaran diterima oleh semua orang sevara global. Ada beberapa isu untuk kebutuhan ini:

- Harus dipastikan bahwa sistem pembayaran yang dibuat oleh satu *vendor* dapat bekerjasama dengan baik dengan sistem pembayaran buatan *vendor* lain.
- Menggunakan sistem kartu pembayaran yang sudah ada.
- Menggunakan teknologi yang boleh dipakai oleh semua negara untuk menjamin bahwa sistem pembayaran ini dapat diterima di seluruh dunia.
- Dibuat di atas standar-standar yang sudah ada dan sudah diterima secara luas.
- Dapat dijalankan di semua *platform*, yaitu semua kombinasi perangkat keras dan sistem operasi yang sudah ada, seperti *PowerPC, Intel, Sparc, UNIX, MS-DOS, OS/2, Windows,* dan *Macintosh.*

SET dapat memenuhi 4 kebutuhan tersebut :

- *Confidentiality*, untuk menjaga kerahasiaan data pembayaran selama komunikasi, SET menggunakan teknik enkripsi simetrik maupun asimetrik.
- *Authentication*, untuk autentikasi pedagang maupun pembeli, SET menggunakan sertifikat digital dan teknik tanda tangan digital. SET menggunakan suatu mekanisme yang menghubungkan antara sertifikat digital dengan nomor kartu pembayaran, sehingga penyalahgunaan kartu pembayaran oleh orang yang tidak berhak dapat dihilangkan.

- *Integrity*, untuk menjamin data tidak berubah selama perjalanan dari pengirim ke penerima data, SET menggunakan teknik tanda tangan digital.
- *Interoperability*, SET menggunakan protokol dan format pesan yang spesifik untuk menjamin kepastian komunikasi antara produk-produk sistem pembayaran yang sesuai dengan standard SET.

Untuk pembahasan lebih detil mengenai bagaimana cara SET menjawab empat kebutuhan bisnis tersebut, dapat dilihat pada bab selanjutnya.

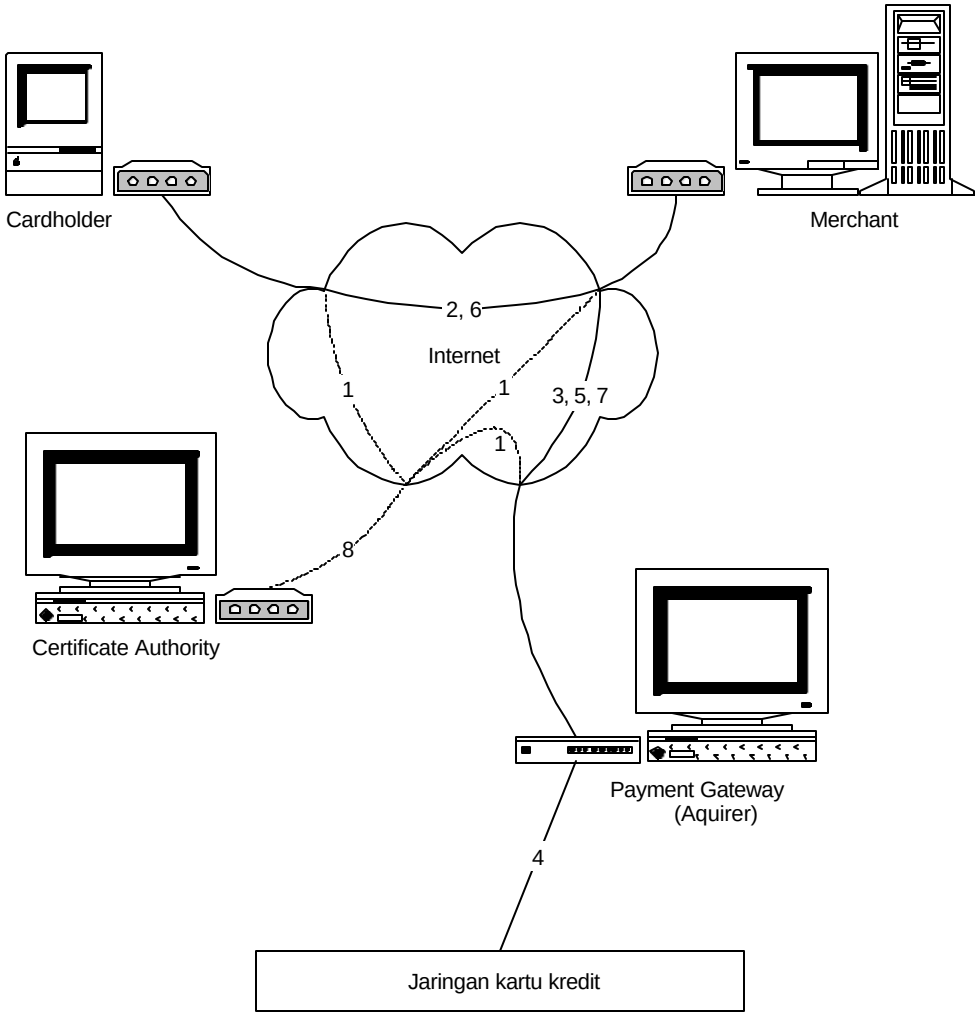
3.3 PIHAK-PIHAK YANG TERLIBAT

Pihak-pihak yang terlibat dalam protokol SET adalah:

- *Cardholder*, yaitu konsumen pemegang kartu pembayaran yang dijamin oleh suatu *Issuer* tertentu untuk transaksi pembelian melalui Internet
- *Issuer*, yaitu suatu institusi keuangan atau bank yang mengeluarkan kartu pembayaran yang dipunyai oleh *Cardholder. Issuer* pula yang melakukan otorisasi terhadap kartu pembayaran yang digunakan.
- *Merchant,* yaitu penjual yang menjual dagangannya melalui Internet dan dijamin oleh *Acquirer* dalam melakukan transaksi melalui Internet.
- *Acquirer,* yaitu suatu institusi keuangan atau bank yang menjamin pedagang (*Merchant*) yang melakukan transaksi melalui Internet. *Acquirer* melakukan otorisasi terhadap pembayaran dalam setiap transaksi melalui Internet.
- *Payment Gateway*, yaitu suatu alat yang dioperasikan oleh *Acquirer* untuk memberikan layanan pemrosesan instruksi pembayaran dalam transaksi melalui Internet. *Payment Gateway* menghubungkan antara *Issuer* dan *Acquirer*.
- *Brand*, yaitu suatu institusi di atas *Issuer* dan *Acquirer*, merupakan pemilik merek dari produk dan sistem pembayaran yang digunakan dalam transaksi melalui Internet.

Komunikasi yang dibahas di dalam SET hanyalah antara *Cardholder, Merchant, Payment Gateway,* dan *Certificate Authority*.

3.4 ALUR TRANSAKSI SET



Gambar III-1. Alur transaksi SET

Secara singkat, alur transaksi pada skenario SET dapat dijelaskan sebagai berikut:

1) Untuk melakukan transaksi SET, *Cardholder* dan *Merchant* harus mendapatkan sertifikat terlebih dahulu dari *Certificate Authority* (CA). *Cardholder* dalam langkah ini harus mengetikkan *primary account number* (PAN) dan informasi jati dirinya. *Merchant* dalam langkah ini juga harus memberikan informasi jati dirinya kepada CA.
2) *Cardholder* kemudian dapat mulai berbelanja. Jika sudah memilih barang apa yang hendak dibeli, *Cardholder* membuat *order instruction* (OI) dan *payment instruction* (PI). *Cardholder* menyerahkan OI dan PI kepada *Merchant* untuk diproses. PI tidak bisa

dibaca oleh *Merchant* karena dienkripsi dengan kunci publik *Payment Gateway*.

3) Setelah *Merchant* memproses OI, maka *Merchant* melakukan otorisasi PI melalui *Payment Gateway*.
4) *Payment Gateway* melakukan otorisasi kartu pembayaran dengan *Issuer* melalui jaringan privat kartu kredit.
5) Jika otorisasi disetujui, maka *Payment Gateway* menginstruksikan *Merchant* untuk menyerahkan barang dagangannya kepada *Cardholder*.
6) *Cardholder* menerima produk yang dibelinya.
7) *Merchant* kemudian dapat memperoleh pembayarannya dengan melakukan proses *capture* melalui *Payment Gateway*. Langkah ini sering di-*batch*, sehingga akan ada tenggang waktu antara permintaan pembayaran (*payment capture*) dengan proses otorisasi.
8) Setiap melakukan komunikasi, setiap pihak yang terlibat dalam transaksi dapat melakukan otentikasi sertifikat digital pihak yang lain dengan menghubungi CA.

Dalam dokumentasi SET, alur transaksi SET dibagi dalam beberapa fase, yaitu:

1. *Cardholder Registration.* Pada fase ini *Cardholder* mendaftar pada CA sebelum dapat melakukan transaksi. Setelah proses ini selesai, maka *Cardholder* akan memiliki sebuah sertifikat digital yang dapat dipakai untuk melakukan transaksi selanjutnya.
2. *Merchant Registration*. *Merchant* harus mendaftar pada CA sebelum dapat menerima PI dari *Cardholder* atau memproses transaksi SET melalui *Payment Gateway.* Sama seperti *Cardholder*, maka *Merchant* akan memiliki sertifikat digital setelah proses ini selesai.
3. *Purchase Request*. Pada fase inilah transaksi yang sebenarnya antara pembeli dan penjual terjadi. Setelah *Cardholder* selesai memilih produk yang akan dibelinya, menyepakati harga yang ditawarkan, dan memilih metode pembayaran, maka proses ini dimulai. Pada proses inilah OI dan PI dibuat oleh *Cardholder* yang kemudian dikirimkan ke *Merchant* untuk diproses. Dari sisi pembeli, setelah proses ini selesai, maka selesai pula partisipasinya pada transaksi, dan dia tinggal menunggu produk yang dibelinya diantarkan atau menerima jasa yang dibutuhkan.
4. *Payment Authorization.* Pada tahap ini kartu pembayaran yang dimiliki *Cardholder* diotorisasi. Otorisasi dilakukan melalui *Payment Gateway.* Setelah kartu pembayaran diotorisasi, maka *Merchant* menjalankan kewajibannya kepada *Cardholder*.
5. *Payment Capture*. Setelah selesai memproses pesanan yang diminta oleh *Cardholder*,

maka pada tahap ini *Merchant* dapat meminta pembayarannya kepada *Acquirer* melalui *Payment Gateway*.

3.5 BATASAN SET

Hal-hal yang didefinisikan di dalam SET adalah:

- Protokol komunikasi antara 4 entitas, yaitu *Cardholder*, *Merchant*, *Payment Gateway* ,dan *Certificate Authority*.
- Pemakaian aplikasi enkripsi (seperti RSA dan DES) dan aplikasi penyandian searah (seperti SHA-1).
- Format dari sertifikat digital, pesan pembelian, otorisasi, dan pembayaran untuk *Merchant.*

Yang tidak didefinisikan dalam SET:

- Protokol pesan untuk penawaran, pemilihan, dan pengantaran produk kepada pembeli.
- Protokol komunikasi antara Cardholder sebagai pemegang kartu pembayaran dan Issuer sebagai penerbit kartu pembayaran.
- Protokol komunikasi antara *Acquirer* dan *Issuer* untuk otorisasi kartu pembayaran.
- Keamanan data yang disimpan oleh Cardholder, Merchant, Certificate Authority, dan Payment Gateway. Termasuk keamanan data dari serangan virus, hackers, dan program kuda trojan.
- Protokol jaringan komunikasi antara entitas-entitas pada SET

3.6 KOMUNIKASI ANTARA CARDHOLDER DAN MERCHANT

Sesuai dengan ruang lingkup penelitian, berikut ini akan dijabarkan komunikasi antara *Cardholder* dan *Merchant* yang terjadi pada tahap *Purchase Request*. Tahap *Purchase Request* dimulai setelah pembeli selesai memilih produk, dan menentukan metode pembayaran.

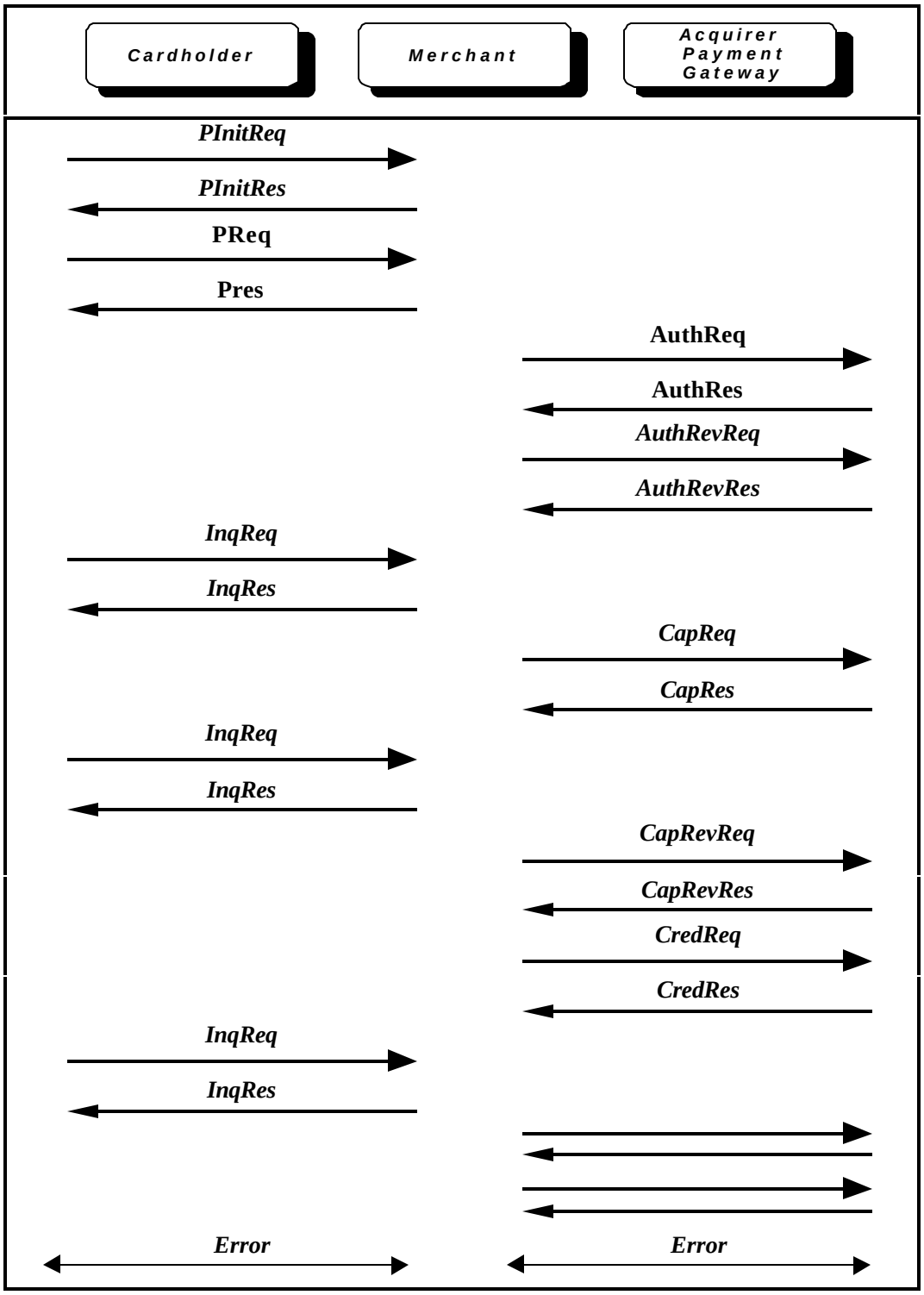
1. *Cardholder* mengirimkan *Purchase Initiation Request* kepada *Merchant.*
2. Program pada *Merchant* menerima *Purchase Initiation Request* dari *Cardholder*.

3. *Merchant* membuat *Purchase Initiation Response* dan menandatanganinya secara digital dengan cara membuat *hash* dari respon tersebut dan mengenkripnya dengan kunci privat milik *Merchant.*
4. *Merchant* mengirimkan respon tersebut bersama sertifikat digital miliknya dan milik *Payment Gateway* kepada *Cardholder.*
5. *Cardholder* menerima respon yang dikirim oleh *Merchant* dan memeriksa keabsahan sertifikat digital milik *Merchant* dengan cara menelusuri hirarki sertifikat digital sampai ke otoritas sertifikat utama.
6. *Cardholder* memeriksa kebenaran tanda tangan digital *Merchant* dengan cara mendekrip tanda tangan digital *Merchant* dengan kunci publik *Merchant* dan membandingkannya dengan *hash* dari respon yang dikirim *Merchant.*
7. *Cardholder* membuat OI berdasarkan data yang didapatnya selama tahap belanja.
8. *Cardholder* membuat PI
9. *Cardholder* membuat tanda tangan ganda dengan cara menyandikan searah gabungan *hash* dari OI dan PI, dan kemudian mengenkripnya menggunakan kunci privat *Cardholder*.
10. *Cardholder* membuat suatu kunci simetri yang baru dan menggunakannya untuk mengenkrip PI. Kemudian simetri kunci tersebut bersama dengan informasi *account* dari *Cardholder* dienkrip menggunakan kunci publik *Payment Gateway*.
11. *Cardholder* mengirimkan PI yang sudah terenkrip tadi bersama dengan OI kepada *Merchant.*
12. *Merchant* menerima *Purchase Request* yang dikirim oleh *Cardholder* dan memeriksa keabsahan sertifikat digital *Cardholder* dengan cara menelusuri hirarki sertifikat digital sampai ke otoritas sertifikat utama.
13. *Merchant* memeriksa kebenaran tanda tangan ganda *Cardholder* dengan cara mendekrip tanda tangan ganda tersebut menggunakan kunci publik *Cardholder* dan membandingkannya dengan hasil penyandian searah gabungan *hash* dari OI dan PI yang diterima.
14. *Merchant* memproses permohonan dari *Cardholder* tersebut, termasuk meneruskan PI kepada *Payment Gateway* untuk diotorisasi.
15. *Merchant* membuat *Purchase Response* yang berisi juga sertifikat digital milik *Merchant.* Respon tersebut ditandatangani secara digital dengan cara membuat *hash*-nya yang kemudian dienkrip menggunakan kunci privat *Merchant.*

16. *Merchant* mengirimkan respon tersebut kepada *Cardholder*.
17. Jika kartu pembayaran telah diotorisasi oleh *Payment Gateway*, maka *Merchant* menjalankan kewajibannya, yaitu mengirimkan produk yang dibeli atau menjalankan jasa yang ditawarkan.
18. *Cardholder* menerima respon dari *Merchant* dan memeriksa keabsahan sertifikat *Merchant* dengan cara menelusuri hirarki sertifikat digital sampai ke otoritas sertifikat utama.
19. *Cardholder* memeriksa kebenaran tanda tangan digital *Merchant* dengan cara mendekripnya menggunakan kunci publik *Merchant* dan membandingkan hasilnya dengan *hash* dari respon tersebut.
20. *Cardholder* menyimpan *Purchase Response* tersebut.

3.7 PESAN-PESAN PEMBAYARAN DALAM SET

Pasa dasarnya komunikasi antara entitas pada SET adalah menggunakan pesan-pesan yang dibuat dan dipertukarkan oleh masing-masing entitas. Pesan-pesan tersebut merupakan pasangan permintaan (*request*) dan respon (*response*), dan dipertukarkan antara *Cardholder* dan *Merchant* serta antara *Merchant* dan *Payment Gateway*. Berikut ini adalah gambar alur pertukaran pasangan pesan-pesan tersebut pada fase *Purchase Request*, *Payment Authorization*, dan *Payment Capture*.



Gambarl III-2. Pesan-pesan pembayaran dalam SET

1. PInitReq/PInitRes

Merupakan inisialisasi dari proses pembayaran, untuk penentuan katu pembayaran yang

digunakan, dan agar *Cardholder* mendapatkan sertifikat dan CRL milik *Merchant* dan *Payment Gateway*.

2. **PReq/PRes**

   Merupakan pasangan pesan proses pembayaran. *PReq* berisi OI dan PI yang dibuat oleh *Cardholder*. Sedangkan *PRes* adalah jawaban dari *Merchant* yang berisi status hasil otorisasi atau tidak.

3. **AuthReq/AuthRes**

   Berisi permintaan (*request*) otorisasi kartu pembayaran oleh *Merchant* kepada *Payment Gateway* dan jawabannya (*response*).

4. **AuthRevReq/AuthRevRes**

   Pasangan pesan untuk pembatalan dari permintaan otorisasi sebelumnya.

5. **InqReq/InqRes**

   Pasangan pesan yang digunakan *Cardholder* untuk memeriksa status dari transaksi.

6. **CapReq/CapRes**

   Pasangan pesan yang digunakan oleh *Merchant* untuk mendapatkan pembayaran dari *Acquirer*, setelah kartu pembayaran diotorisasi oleh *Payment Gateway*.

7. **CapRevReq/CapReqRev**

   Untuk pembatalan atau pengurangan jumlah pembayaran kepada *Merchant*

8. **CredReq/CredRes**

   Untuk proses *credit* terhadap transaksi yang sudah disahkan, yaitu pengembalian uang yang telah dibayarkan kepada *Merchant.*

9. **Error**

   Pesan-pesan yang dikirim jika terjadi kesalahan pada proses transaksi, misalnya jika terjadi perubahan isi pesan selama pengiriman. Pesan ini tidak memberitahukan kalau terjadi kegagalan otorisasi kartu pembayaran.

3.8 NOTASI PENULISAN

Sebelum melangkah lebih jauh ke pembahasan spesifikasi SET, berikut ini diberikan notasi-notasi penulisan yang akan dipakai seterusnya dalam penulisan.

Konsep	Notasi	Definisi	
Tuple	**{A, B, C}**	Sebuah pengelompokkan dari nol atau lebih elemen data. Maksud notasi disamping adalah suatu tuple yang terdiri dari A,B, dan C yang masing-masing bisa saja berupa tuple.	
Komponen	**T = {A, B, C}**	Sebuah tuple yang mempunyai nama, diwakili oleh T, maka komponen dari tuple tersebut dapat disebut sebagai T.A, T.B, dan T.C.	
Penggabungan berurutan	**A | B | C**	Menunjukkan penggabungan secara berurutan dari A,B, dan C.	
Opsional	**[A]**	Maksudnya elemen A boleh ada boleh tidak.	Untuk notasi-notasi ini dapat dilakukan penulisan bersarang.
Pilihan	**<A, B, C>**	Maksudnya hanya ada salah satu dari A, B, and C boleh ada.	
Pilihan bersifat opsional	**[<A, B, C>]**	Maksudnya pilihan tersebut bersifat opsional, atau hanya salah satu dari A, B, and C atau tidak satupun yang boleh ada.	
Multiple instances	**{A +}**	Notasi ini berarti sebuah tuple yang mengandung satu atau lebih instances dari A.	
	{A *}	Notasi ini berarti sebuah tuple yang mengandung nol atau lebih instances dari A.	
	{[A] +}	Notasi ini berarti sebuah tuple yang mengandung nol atau lebih instances dari A, dimana setiap instances dari A bersifat opsional.	
Exclusive-or	**A Å B**	Menunjukkan operasi exclusive-or (XOR).	

Tabel III-1. Notasi penulisan

Selain notasi tersebut di atas, untuk mewakili entitas-entitas yang terlibat pada SET, notasi berikut dipergunakan:

- **C** untuk mewakili *Cardholder*.
- **M** untuk mewakili *Merchant.*
- **CA** untuk mewakili *Certificate Authority*.

- **P** untuk mewakili *Payment Gateway*.

Terkadang bisa terdapat lebih dari satu *Payment Gateway* yang diperlukan untuk suatu transaksi. Oleh karena itu notasi **P1** dipergunakan untuk mewakili *Payment Gateway* pertama dan **P2** untuk mewakili *Payment Gateway* kedua.

Untuk penulisan spesifikasi dalam ASN.1, di depan setiap baris dari spesifikasi tersebut diberikan suatu bilangan. Bilangan tersebut menunjukkan nomor baris pemunculan kode ASN.1 tersebut pada buku 3 spesifikasi SET. Nomor baris tersebut ditulis agar memudahkan pencarian spesifikasi dalam ASN.1 pada dokumen SET sebagai referensi.

3.9 OPERATOR KRIPTOGRAFI

Pembahasan berikutnya adalah mengenai operator-operator kriptografi yang dipakai dalam protokol SET.

3.9.1 Fungsi *Hash*

Notasi	Operator	Deskripsi
H(*t*)	*Hash*	*Hash* 160-bit SHA-1 dari *tuple* **t**.
HMAC(*t, k*)	*Hash* menggunakan kunci	*Hash* 160-bit SHA-1 dari *tuple* **t** menggunakan kunci **k** berdasarkan HMAC-SHA-1. HMAC(t,k) = H((k ⊕ opad) | H((k ⊕ ipad) | t)) Dimana: **ipad** adalah *byte* 0x36 yang diulang sebanyak 64 kali, **opad** adalah *byte* 0x5C yang diulang sebanyak 64 kali; dan ⊕ adalah fungsi XOR.
L(*t1, t2*)	*Linkage*	Sebuah referensi, penunjuk, atau *link* ke **t2** yang diberikan bersama **t1**, sama dengan *tuple* **{t1, H(t2)}**.

Tabel III-2. Operator *Hash*

3.9.2 Tanda Tangan Digital

Notasi	Operator	Deskripsi
S(s, *t*)	Pesan yang ditandatangani	Tanda tangan dari entitas **s** pada *tuple* **t**, termasuk juga *plaintext* dari **t**. Algoritma tanda tangan digital yang dipakai dalam SET adalah RSA dengan SHA-1. Berhubungan dengan **SignedData** pada PKCS #7.
SO(s, *t*)	Tanda tangan saja	Tanda tangan dari entitas **s** pada *tuple* **t**, tidak termasuk *plaintext* dari **t**. Algoritma tanda tangan digital yang dipakai dalam SET adalah RSA dengan SHA-1.

Tabel III-3. Operator tanda tangan digital

3.9.3 Enkripsi

Notasi	Operator	Deskripsi
E(*r, t*)	Enkripsi asimetris	Mula-mula, enkrip **t** dengan sebuah kunci simetrik **k**, kemudian bungkus **k** di dalam **OAEP(k)**. Setelah itu enkrip **OAEP(k)** tadi mengunakan kunci publik entitas **r**, yang didapat dari sertifikat dari dalam *tuple* **r**. Berhubungan dengan **EnvelopedData** pada PKCS #7.
EX(*r, t, p*)	Enkripsi ekstra	Operator ini mirip dengan **E,** tapi dengan tambahan parameter **p**. Parameter tersebut diletakkan pada blok **OAEP**. Kemudian yang dienkrip menggunakan sebuah kunci **k** adalah **L(t,p)**.
EXH(*r, t, p*)	Enkripsi ekstra dengan integritas	Operator ini mirip dengan **EX,** tapi dengan tambahan **H(t)** dmasukkan juga ke dalam blok **OAEP**.
EK(*k, t*)	Enkripsi menggunakan kunci simetris	Enkripsi *tuple* **t** menggunakan kunci simetris **k**. Berhubungan dengan **EncryptedData** pada PKCS #7.

Tabel III-4. Operator enkripsi

3.9.4 Enkapsulasi

Notasi	Operator	Deskripsi
Enc(*s, r, t*)	Enkapsulasi sederhana dengan tanda tangan	*Tuple* **t** ditandatangani oleh entitas **s** kemudian dienkrip menggunakan kunci publik entitas **r**. Berhubungan dengan **SignedData** pada PKCS #7 yang dienkapsulasi di dalam **EnvelopedData.**
EncK(*k, s, t*)	Enkapsulasi sederhana dengan tanda tangan dan enkripsi simetris	*Tuple* **t** ditandatangani oleh entitas **s** kemudian dienkrip menggunakan kunci simetris **k**. Berhubungan dengan **EncryptedData** pada PKCS #7.

Tabel III-5. Operator enkapsulasi

3.10 STRUKTUR DATA

Sebelum membahas mengenai pasangan pesan yang dibentuk dan dipertukarkan oleh *Cardholder* dan *Merchant,* dibahas lebih dulu mengenai struktur data yang memuat data-data yang diperlukan dalam pesan-pesan SET. Pembahasan struktur data berikut dibatasi hanya yang diperlukan dalam komunikasi antara *Cardholder* dan *Merchant* saja.

3.10.1 TransIDs

Merupakan identitas dari suatu pesan. Untuk menandakan suatu pesan tertentu termasuk bagian dari transaksi yang mana dan menandakan juga suatu pasangan pesan. Misalnya menandakan suatu **PInitReq** termasuk dari transaksi yang mana, dan menandakan **PInitRes** mana yang merupakan pasangannya.

Berikut ini adalah spesifikasi *TransIDs* dalam format *ASN.1*:

```
337 TransIDs ::= SEQUENCE {
338     lid-C           LocalID,
339     lid-M           [0] LocalID  OPTIONAL,
340     xid             XID,
341     pReqDate        Date,
342     paySysID        [1] PaySysID  OPTIONAL,
343     language        Language        -- Cardholder requested session language
344 }

348 XID ::= OCTET STRING (SIZE(20))
```

320 PaySysID ::= VisibleString (SIZE(1..ub-paySysID))

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

Keterangan mengenai *field* yang terdapat pada **TransIDs** :

TransIDs	**{LID-C, [LID-M], XID, PreqDate, [PaySysID], Language }**
LID-C	Identitas lokal; suatu label yang dibuat oleh *Cardholder*.
LID-M	Identitas lokal; suatu label yang dibuat oleh *Merchant.*
XID	Identitas yang unik secara global.
PreqDate	Tanggal suatu transaksi dimulai, dibuat oleh *Merchant* di dalam **PInitRes** atau oleh *Cardholder* di dalam **PReq**.
PaySysID	Digunakan oleh beberapa merek kartu pembayaran untuk menandakan transaksi dari waktu otorisasi.
Language	Bahasa yang dipilih oleh *Cardholder* untuk digunakan selama transaksi.

Tabel III-6. TransIDs

XID adalah suatu identitas dari transaksi yang dibuat oleh *Merchant,* kecuali pada kasus tidak adanya **PInitRes**, dibuat oleh *Cardholder*. **XID** berisi 20 *byte* angka yang dibuat secara acak.

Pembuatan **TransIDs** :

1. Jika suatu pesan yang mendahului sudah pernah diterima, maka salin semua *field* yang terdapat pada **TransIDs**.
2. Jika pesan ini merupakan transaksi yang baru (belum ada pesan yang mendahuluinya), maka buat *field-field* yang diperlukan.
3. Isi setiap *field* yang bersifat opsional oleh entitas yang bersangkutan.

3.10.2 PI (*Payment Instruction*)

PI merupakan data yang paling vital dan sensitif dalam SET, karena berisi informasi kartu pembayaran dan jumlah yang harus dibayarkan untuk diotorisasi oleh *Payment Gateway*. PI dibuat dan dikirim oleh *Cardholder* kepada *Payment Gateway* melalui *Merchant.* PI dienkrip oleh *Cardholder* menggunakan kunci publik *Payment Gateway* sehingga *Merchant* tidak bisa melihat isinya.

PI ada 3 macam, dua yang pertama dibuat oleh *Cardholder* untuk diproses oleh

Payment Gateway, sedangkan yang ketiga dibuat oleh *Payment Gateway* untuk mendukung pengiriman secara terpisah dan penagihan yang berulang. Tiga macam PI tersebut adalah:

1. **PIUnsigned**, dibuat oleh *Cardholder* yang tidak memiliki sertifikat digital, digunakan di dalam pesan **PReqUnsigned** Keutuhan data dijamin dengan cara menyertakan nilai *hash* dari PI di dalam blok OAEP. Autentikasi sumber data tidak dijamin dengan cara ini.
2. **PIDualSigned**, dibuat oleh *Cardholder* yang memiliki sertifikat digital, digunakan di dalam pesan **PreqDualSigned**. Keutuhan data dan autentikasi sumber dijamin dalam cara ini.
3. **AuthToken**, dibuat oleh *Payment Gateway* untuk dikirimkan ke *Merchant.* Selanjutnya tipe PI ini tidak dibahas, karena sudah merupakan bagian dari komunikasi antara *Merchant* dengan *Payment Gateway*.

Spesifikasi **PI** dalam ASN.1 :

```
822 PI ::= CHOICE {
823     piUnsigned      [0] EXPLICIT PIUnsigned,
824     piDualSigned    [1] EXPLICIT PIDualSigned,
825     authToken       [2] EXPLICIT AuthToken
826 }

898 PIUnsigned ::= EXH { P, PI-OILink, PANToken }

799 PIDualSigned ::= SEQUENCE {
800     piSignature     PISignature,
801     exPIData        EX { P, PI-OILink, PANData }
802 }

807 PI-OILink ::= L { PIHead, OIData }

811 PISignature ::= SO { C, PI-TBS }

813 PI-TBS ::= SEQUENCE {
814     hPIData         HPIData,
815     hOIData         HOIData
816 }

818 HPIData ::= DD { PIData }                  -- PKCS#7 DigestedData

820 HOIData ::= DD { OIData }                  -- PKCS#7 DigestedData

828 PIData ::= SEQUENCE {
829     piHead          PIHead,
830     panData         PANData
831 }
```

Keterangan mengenai *field* yang terdapat pada **PI** :

PI	**< PIUnsigned, PIDualSigned, AuthToken >** Pilihan dari salah satu di atas.
PIUnsigned	**EXH(P, PI-OILink, PANToken)** Keutuhan data pada **PIUnsigned** dijamin dengan menyertakan *hash* dari **PI-OILink** di dalam blok OAEP secara terenkrip.
PIDualSigned	**{PISignature, EX(P, PI-OILink, PANData)}** Tanda tangan *Cardholder* terdapat pada **PISignature**, dan kerahasiaan PI didapat dari enkapsulasi menggunakan operator **EX** dengan pihak penerimanya adalah *Payment Gateway*.
AuthToken	Tidak dibahas lebih lanjut
PI-OILink	**L(PIHead, OIData)** Suatu link yang menghubungkan antara PI dengan OI. Gunanya agar dapat diketahui bahwa suatu PI (berisi informasi kartu pembayaran) berhubungan dengan suatu pembelian tertentu (informasinya terdapat di dalam OI).
PISignature	**SO(C, PI-TBS)** Tanda tangan digital dari PI, dengan *Cardholder* sebagai pihak yang menandatanganinya. Teknik tanda tangan ganda terpakai dalam *field* ini.
PI-TBS	**{HPIData, HOIData}** Nilai *hash* dari PI dan OI, dipakai untuk tanda tangan digital *Cardholder* terhadap PI
HPIData	**DD(PIData)** Nilai *hash* dari PI
HOIData	**DD(OIData)** Nilai *hash* dari OI
PIData	**{PIHead, PANData}**

Tabel III-7. Payment Instruction

Spesifikasi **PIData** dalam ASN.1 :

```
833 PIHead ::= SEQUENCE {
834     transIDs                TransIDs,
835     inputs                  Inputs,
836     merchantID              MerchantID,
837     installRecurData        [0] InstallRecurData  OPTIONAL,
838     transStain              TransStain,
839     swIdent                 SWIdent,
840     acqBackKeyData          [1] EXPLICIT BackKeyData  OPTIONAL,
841     piExtensions            [2] MsgExtensions {{PIExtensionsIOS}} OPTIONAL
842 }

846 Inputs ::= SEQUENCE {
```

```
847     hod             HOD,
848     purchAmt        CurrencyAmount
849 }
```

```
294 MerchantID ::= SETString { ub-MerchantID }
```

```
851 TransStain ::= HMAC { XID, Secret }
```

```
328 SWIdent ::= VisibleString (SIZE(1..ub-SWIdent))    -- Software identification
```

```
870 HOD ::= DD { HODInput }
```

```
348 XID ::= OCTET STRING (SIZE(20))
```

```
1102 AcqBackKey ::= BackKeyData
```

Keterangan mengenai *field* yang terdapat pada **PIHead** :

PIHead	**{TransIDs, Inputs, MerchantID, [InstallRecurData], TransStain, SWIdent, [AcqBackKeyData], [PIExtensions]}**
TransIDs	Lihat keterangan pada hal 34
Inputs	**{HOD, PurchAmt}**
MerchantID	Identitas *Merchant,* disalin dari sertifikat milik *Merchant.*
InstallRecurData	Lihat halaman 39.
TransStain	**HMAC(XID, CardSecret)**
SWIdent	Suatu *string* yang menandakan program yang dipakai oleh *Cardholder*.
AcqBackKeyData	**{AcqBackAlg, AcqBackKey}** *Field* ini berisi data yang diperlukan oleh *Payment Gateway* untuk mengirimkan data kepada *Cardholder.* Lihat keterangan mengenai **AcqCardMsg** pada halaman 40.
PIExtensions	Berisi data-data tambahan yang berhubungan dengan pemrosesan otorisasi oleh *Payment Gateway*.

Tabel III-8. PIHead

HOD	Nilai *hash* dari data-data *order description*, isinya sama dengan **HOD** yang terdapat dalam **OIData**.
PurchAmt	Jumlah pembayaran transaksi yang disepakati oleh *Cardholder* dan *Merchant.*
XID	Disalin dari **TransIDs**.
CardSecret	Diambil dari **PANData0** yang terdapat pada pesan **CertReq**.
AcqBackAlg	Algoritma enkripsi yang terdapat dalam sertifikat *Payment Gateway*.
AcqBackKey	Kunci untuk **AcqCardMsg**.

Tabel III-8. PIHead (lanjutan)

3.10.3 InstallRecurData

Struktur data ini memuat informasi mengenai cara pembayaran selain pembayaran penuh oleh *Cardholder*, yaitu secara mengangsur (*installment*) atau berlangganan (*recurring*).

Spesifikasi **InstallRecurData** dalam ASN.1 :

```
1945 InstallRecurData ::= SEQUENCE {
1946     installRecurInd          InstallRecurInd,
1947     irExtensions             [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }

1952 InstallRecurInd ::= CHOICE {
1953     installTotalTrans        [0] INTEGER (2..MAX),
1954     recurring                [1] Recurring
1955 }

1957 Recurring ::= SEQUENCE {
1958     recurringFrequency       INTEGER (1..ub-recurringFrequency),
1959     recurringExpiry          Date
1960 }
```

Keterangan mengenai *field* yang terdapat pada **InstallRecurData** :

InstallRecurData	**{InstallRecurInd, [IRExtensions]}**
InstallRecurInd	**< InstallTotalTrans, Recurring >**
IRExtensions	Data yang terdapat dalam *field* ini berhubungan untuk pemrosesan otorisasi transaksi oleh *Merchant* dan *Payment Gateway*. Isinya tergantung dari kebijakan kedua entitas tersebut.
InstallTotalTrans	Jumlah maksimum banyaknya angsuran yang diperbolehkan oleh *Cardholder*.
Recurring	**{RecurringFrequency, RecurringExpiry}**
RecurringFrequency	Menyatakan frekuensi berlangganan. Isinya adalah jumlah minimum hari antara dua otorisasi yang berurutan. Untuk menyatakan waktu satu bulan, diberikan angka 28.
RecurringExpiry	Tanggal batas akhir otorisasi. Setelah lewat tanggal tersebut, penagihan tidak diperbolehkan lagi..

Tabel III-9. InstallRecurData

3.10.4 AcqCardMsg

Struktur data ini berguna bila *Acquirer* ingin mengirimkan data ke *Cardholder*. Pengiriman pesan data tersebut melalui *Merchant* tanpa *Merchant* mengetahui isinya. Hal ini dicapai dengan cara:

1. *Cardholder* membuat suatu kunci simetrik untuk mengenkrip/mendekrip pesan yang dikirim oleh *Acquirer*.
2. Kunci tersebut diletakkan di dalam **PI,** dan karena **PI** hanya bisa dibaca oleh *Payment Gateway*, maka *Merchant* tidak bisa mengetahui kunci simetri tersebut

Spesifikasi **AcqCardMsgb** dalam ASN.1 :

```
1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }

1109 AcqCardCodeMsg ::= SEQUENCE {
1110     acqCardCode             AcqCardCode,
1111     acqCardMsgData          AcqCardMsgData
1112 }

1114 AcqCardCode ::= ENUMERATED {
1115     messageOfDay            (0),
1116     accountInfo             (1),
1117     callCustomerService     (2)
1118 }
```

```
1120 AcqCardMsgData ::= SEQUENCE {
1121    acqCardText     [0] EXPLICIT SETString { ub-acqCardText }  OPTIONAL,
1122    acqCardURL      [1] URL  OPTIONAL,
1123    acqCardPhone  [2] EXPLICIT SETString { ub-acqCardPhone }  OPTIONAL
1124 }
```

Keterangan mengenai *field* yang terdapat pada **AcqCardMsg** :

AcqCardMsg	**EncK(AcqBackKeyData, P, AcqCardCodeMsg)** **AcqBackKeyData** didapat dari *Cardholder* yang memberikannya di dalam **PI**.
AcqBackKeyData	Disalin dari **PIHead.AcqBackKeyData** yang dikirimkan oleh *Cardholder* melalui *Merchant.* Lihat halaman 35 mengenai **PI**.
AcqCardCodeMsg	**{AcqCardCode, AcqCardMsgData}**
AcqCardCode	Kode mengenai tipe pesan yang disampaikan oleh *Payment Gateway*.
AcqCardMsgData	**{[AcqCardText], [AcqCardURL], [AcqCardPhone]}** Pesan yang disampaikan oleh *Payment Gateway*.
AcqCardText	Pesan dalam bentuk teks
AcqCardURL	*URL* yang menunjuk ke isi pesan pada Internet.
AcqCardPhone	Nomor telepon yang diberikan pada *Cardholder* agar dapat mengetahui isi pesan.

Tabel III-10. AcqCardMsg

3.10.5 PANData

Struktur data ini menyimpan informasi mengenai kartu pembayaran yang dipakai oleh *Cardholder*. Dipakai dalam **PI** yang ditandatangani (**PIDualSigned**).

Spesifikasi **PANData** dalam ASN.1 :

```
300 PANData ::= SEQUENCE {
301     pan             PAN,
302     cardExpiry      CardExpiry,
303     panSecret       Secret,
304     exNonce         Nonce
305 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) – YYYYMM expiration date of card
```

```
296 Nonce ::= OCTET STRING (SIZE(20))
```

Keterangan mengenai *field* yang terdapat di dalam **PANData** :

PANData	**{PAN, CardExpiry, PANSecret, EXNonce}**
PAN	*Primary Account Number;* biasanya adalah nomor kartu pembayaran
CardExpiry	Tanggal berakhirnya masa berlaku kartu.
PANSecret	Sebuah nilai rahasia yang diketahui oleh *Cardholder, Payment Gateway,* dan *Cardholder CA*. Gunanya untuk mencegah penebakan **PAN** yang terdapat dalam sertifikat *Cardholder.* Masalah penyimpanan **PAN** di dalam sertifikat tidak dibahas, karena berada di luar ruang lingkup penelitian.
EXNonce	Suatu nilai acak untuk yang digunakan mencegah *dictionary attacks* pada **PANData.**

Tabel III-11. PANData

Langkah-langkah untuk membuat **PANData**:

1. Isi **PAN** dengan nomor kartu pembayaran milik *Cardholder*
2. Isi **CardExpiry** dengan tanggal kadaluarsa kartu pembayaran
3. Isi **PANSecret** dengan **PANSecret** yang didapat dari *Cardholder CA.*
4. Buat **EXNonce** secara acak.

3.10.6 PANToken

PANToken menyimpan informasi mengenai kartu pembayaran yang dipakai oleh *Cardholder*, dipakai oleh **PI** dalam bentuk yang tidak ditandatangani (**PIUnsigned**).

Spesifikasi **PANToken** dalam ASN.1 :

```
314 PANToken ::= SEQUENCE {
315     pan             PAN,
316     cardExpiry      CardExpiry,
317     exNonce         Nonce
318 }
```

Keterangan mengenai *field* yang terdapat di dalam **PANToken**:

PANToken	**{PAN, CardExpiry, EXNonce}**
PAN	*Primary Account Number;* biasanya adalah nomor kartu pembayaran
CardExpiry	Tanggal berakhirnya masa berlaku kartu.
EXNonce	Suatu nilai acak untuk yang digunakan mencegah *dictionary attacks* pada **PANData.**

Tabel III-12. PANToken

Langkah-langkah untuk membuat **PANToken**:

1. Isi **PAN** dengan nomor kartu pembayaran milik *Cardholder*
2. Isi **CardExpiry** dengan tanggal kadaluarsa kartu pembayaran
3. Buat **EXNonce** secara acak.

Perbedaanya dengan **PANData** adalah pada **PANToken** tidak terdapat **PANSecret**, hal ini disebabkan pada perbedaan penggunaan antara keduanya. **PANToken** dipakai pada **PI** oleh *Cardholder* yang tidak memiliki sertifikat digital. Oleh karena *Cardholder* tidak memiliki sertifikat, maka dia tidak ada hubungannya dengan *Certificate Authority*, sehingga tidak mempunyai suatu nilai rahasia (**PANSecret**) yang diketahui oleh *Cardholder*, *Payment Gateway*, dan *Certificate Authority*.

3.10.7 AmountFields

Jumlah pembayaran untuk transaksi pada SET direpresentasikan dalam tiga komponen, yaitu: *currency, amount,* dan *amtExp10.* Ketiga komponen tersebut bertipe integer.

- *currency*, menunjukkan kode mata uang seperti yang terdapat dalam standar ISO 4217. Misalnya, kode mata uang untuk Rupiah adalah 360 dan US$ adalah 840.
- *amount,* menunjukkan jumlah pembayaran dan nilainya tidak boleh negatif
- *amtExp10*, menunjukkan suatu nilai dimana :

  jumlah pembayaran = *amount* x (*10^amtExp*)

  Nilai dari komponen ini adalah bilangan bulat positif atau negatif.

Misalnya untuk menunjukkan pembayaran sejumlah Rp. 1.400,00. Maka *currency* = 360, *amount* = 14, dan *amtExp* = 2. Representasi dari US$ 3,4 adalah *currency* = 840, *amount* = 34, dan *amtExp* = -1.

Struktur data **CurrencyAmount** dipakai untuk menyimpan jumlah pembayaran sesuai ketentuan di atas.

Spesifikasi **CurrencyAmount** dalam ASN.1 :

```
1843 CurrencyAmount ::= SEQUENCE {
1844    currency        Currency, -- Currency code as defined in ISO-4217
1845    amount          INTEGER (0..MAX),
1846    amtExp10        INTEGER (MIN..MAX)
335                     -- Base ten exponent, such that the value in local
335                     -- currency is “amount * (10 ** amtExp10)”
335                     -- The exponent shall be the same value as defined
```

```
335                     -- for the minor unit of currency in ISO-4217.
1851 }

263 Currency ::= INTEGER (1..999)       -- ISO-4217 currency code
```

3.10.8 Blok OAEP

OAEP (*Optimal Asymmetric Encryption Padding*) adalah struktur data yang dipakai untuk menyimpan kunci simetri yang dipakai oleh operator **E, EX,** dan **EXH**. Dalam blok OAEP, setiap bit informasi yang tersimpan tersebar secara merata. Tujuannya untuk mempersulit penebakan kunci tersebut, karena ada suatu teknik kriptoanalisis yang memungkinkan beberapa bit tertentu dari suatu pesan lebih mudah dibedakan dari bit lainnya.

Selain untuk menyimpan kunci simetri pada operator-operator tersebut di atas, pada blok OAEP dapat disimpan juga *hash* dari pesan dan suatu parameter tambahan. Pada SET, parameter tambahan tersebut berupa **PANData** atau **PANToken**.

3.10.9 MessageWrapper

Struktur data ini dipakai untuk membungkus semua pesan-pesan yang dihasilkan di dalam SET. **MessageWrapper** terdiri dari tiga bagian, yaitu **MessageHeader** yang berguna sebagai identifikasi pesan yang dibungkus, **Message** yang dapat terdiri dari salah satu pesan, dan **MWExtension** yang merupakan ekstensi dari pembungkus pesan.

Spesifikasi **MessageWrapper** dalam ASN.1 :

```
43 MessageWrapper ::= SEQUENCE {
44    messageHeader  MessageHeader,
45    message        [0] EXPLICIT MESSAGE.&Type (Message),
46    mwExtensions   [1] MsgExtensions {{MWExtensionsIOS}} OPTIONAL
47 }
```

Spesifikasi **MessageHeader** dalam ASN.1

```
58 MessageHeader ::= SEQUENCE {
59       version          INTEGER { setVer1(1) } (setVer1),
60       revision         INTEGER (0) DEFAULT 0,   -- This is version 1.0
61       date             Date,
62       messageIDs       [0] MessageIDs  OPTIONAL,
63       rrpid            [1] RRPID  OPTIONAL,
64       swIdent          SWIdent
65 }
```

Date menunjukkan waktu dibuatnya pesan yang terbungkus dalam **MessageWrapper**.

Spesifikasi **Message** dalam ASN.1 :

```
75 Message ::= CHOICE {
76
77    purchaseInitRequest                [ 0] EXPLICIT PInitReq,
78    purchaseInitResponse               [ 1] EXPLICIT PInitRes,
79
80    purchaseRequest                    [ 2] EXPLICIT PReq,
81    purchaseResponse                   [ 3] EXPLICIT PRes,
82
83    inquiryRequest                     [ 4] EXPLICIT InqReq,
84    inquiryResponse                    [ 5] EXPLICIT InqRes,
85
86    authorizationRequest               [ 6] EXPLICIT AuthReq,
87    authorizationResponse              [ 7] EXPLICIT AuthRes,
88
89    authReversalRequest                [ 8] EXPLICIT AuthRevReq,
90    authReversalResponse               [ 9] EXPLICIT AuthRevRes,
91
92    captureRequest                     [10] EXPLICIT CapReq,
93    captureResponse                    [11] EXPLICIT CapRes,
94
95    captureReversalRequest             [12] EXPLICIT CapRevReq,
96    captureReversalResponse            [13] EXPLICIT CapRevRes,
97
98    creditRequest                      [14] EXPLICIT CredReq,
99    creditResponse                     [15] EXPLICIT CredRes,
100
101    creditReversalRequest             [16] EXPLICIT CredRevReq,
102    creditReversalResponse            [17] EXPLICIT CredRevRes,
103
104    pCertificateRequest               [18] EXPLICIT PCertReq,
105    pCertificateResponse              [19] EXPLICIT PCertRes,
106
107    batchAdministrationRequest        [20] EXPLICIT BatchAdminReq,
108    batchAdministrationResponse       [21] EXPLICIT BatchAdminRes,
109
110    cardholderCInitRequest            [22] EXPLICIT CardCInitReq,
111    cardholderCInitResponse           [23] EXPLICIT CardCInitRes,
112
113    meAqCInitRequest                  [24] EXPLICIT Me-AqCInitReq,
114    meAqCInitResponse                 [25] EXPLICIT Me-AqCInitRes,
115
116    registrationFormRequest           [26] EXPLICIT RegFormReq,
117    registrationFormResponse          [27] EXPLICIT RegFormRes,
118
119    certificateRequest                [28] EXPLICIT CertReq,
120    certificateResponse               [29] EXPLICIT CertRes,
121
122    certificateInquiryRequest         [30] EXPLICIT CertInqReq,
123    certificateInquiryResponse        [31] EXPLICIT CertInqRes,
124
125    error                             [999] EXPLICIT Error
126 }
```

3.10.10 Error

Struktur data ini digunakan untuk menyampaikan pesan kesalahan yang terjadi selama pemrosesan transaksi di dalam SET. **Error** ada dua macam, yaitu yang ditandatangani dan yang tidak ditandatangani.

Spesifikasi **Error** dalam ASN.1 :

```
144 Error ::= CHOICE {
145     signedError     [0] EXPLICIT SignedError,
146     unsignedError   [1] EXPLICIT ErrorTBS
147 }

149 SignedError ::= S {EE, ErrorTBS}

151 ErrorTBS ::= SEQUENCE {
152     errorCode       ErrorCode,
153     errorNonce      Nonce,
154     errorOID        [0] OBJECT IDENTIFIER  OPTIONAL,
155     errorThumb      [1] EXPLICIT CertThumb  OPTIONAL,
156     errorMsg        [2] EXPLICIT ErrorMsg
157  }
```

Untuk **Error** yang ditandantangani, maka **ErrorTBS** dikirim dengan terlebih dahulu menandatanganinya menggunakan operator **S**.

ErrorCode berisi kode yang menunjukkan kesalahan yang terjadi. **ErrorThumb** menunjukkan sertifikat yang berhubungan dengan kesalahan.**ErrorMsg** berisi salah satu dari **MesageHeader** dari pesan yang menyebabkan kesalahan, atau pesan penyebab kesalahan dalam bentuk DER.

Kode Kesalahan	Arti
unspecifiedFailure	Penyebab kesalahantidak diketahui.
messageNotSupported	Pesan yang diterima tidak didukung oleh penerima
decodingFailure	Kesalahan terjadi ketika mendekode representasi DER.
invalidCertificate	Sertifikat yang diterima tidak sah.
expiredCertificate	Sertifikat sudah tidak berlaku lagi
revokedCertificate	Sertifikat masuk dalam CRL, sehingga tidak berlaku lagi.
missingCertificate	Sertifikat yang diperlukan untuk memroses pesan tidak dapat ditemukan.
signatureFailure	Tanda tangan digital tidak valid.

Tabel III-13. Kode kesalahan

badMessageHeader	Terjadi kesalahan pada kepala pesan..
wrapperMsgMismatch	Identitas pesan yang dibungkus **MessageWrapper** tidak sesuai dengan identitas pesan pada **MessageHeader**.
versionTooOld	Nomor versi pesan terlalu lama .
versionTooNew	Nomor versi pesan terlalu baru..
unrecognizedExtension	Ekstensi pesan tidak dikenali..
messageTooBig	Pesan terlalu besar untuk diproses.
signatureRequired	Pesan yang ditandatangani diperlukan untuk dapat diproses.
messageTooOld	Tanggal pesan terlalu lama.
messageTooNew	Tanggal pesan terlalu baru..
thumbsMismatch	Thumbs yang dikirim tidak valid..
unknownRRPID	**RRPID** yang diterima tidak dikenali.
unknownXID	**XID** yang diterima tidak dikenali
unknownLID	**LID** yang diterima tidak dikenali.
challengeMismatch	Variabel tantangan yang dikirim tidak sesuai dengan variabel tantangan pada pesan jawabannya.

Tabel III-13. Kode kesalahan (lanjutan)

3.11 PASANGAN PESAN YANG DIPERTUKARKAN ANTARA *CARDHOLDER* DAN *MERCHANT*

Pasangan-pasangan pesan berikut ini dipertukarkan antara *Cardholder* dan *Merchant* dalam tahap *purchase request.*

3.11.1 PInitReq/PInitRes

Pasangan pesan ini berfungsi sebagai inisialisasi transaksi menggunakan protokol SET. Pasangan pesan ini bersifat *optional,* jadi boleh ada ataupun tidak. Jika tidak diikutkan dalam transaksi, maka data-data yang dipertukarkan dalam pasangan pesan ini bisa didapatkan dari media lain seperti CD-ROM, tapi dengan ketentuan tidak dapat dijamin kesegaran data yang diterima. Setelah pasangan pesan ini dipertukarkan, maka *Cardholder* mendapatkan sertifikat digital dari *Merchant* dan *Payment Gateway*. Demikian pula sebaliknya, *Merchant* mendapatkan sertifikat digital dari *Cardholder. Payment Gateway*

tidak perlu mendapatkan sertifikat digital dari *Cardholder*, karena *Payment Gateway* tidak pernah berkomunikasi secara langsung dengan *Cardholder*.

PInitReq berisi merek dari kartu pembayaran yang dipakai oleh *Cardholder*, suatu nomor identifikasi yang unik dari *Cardholder* untuk transaksi tersebut, suatu *challenge variable* untuk menjamin kesegaran data yang dikirim, dan *thumbs* (nilai *hash* dari sertifikat dan CRL yang sudah dipunyai *Cardholder*, sehingga tidak perlu dikirim lagi).

PInitRes berisi data-data yang diminta, termasuk sertifikat dan CRL. Dalam *PInitRes*, *Merchant* memberikan tanggal terjadinya transaksi, suatu XID dan menambahkan suatu *challenge variable* untuk menjaga kesegaran data.

Spesifikasi **PInitReq** dalam *ASN.1*:

```
756 PInitReq ::= SEQUENCE {             -- Purchase Initialization Request
757     rrpid           RRPID,
758     language        Language,
759     localID-C       LocalID,
760     localID-M       [0] LocalID  OPTIONAL,
761     chall-C         Challenge,
762     brandID         BrandID,
763     bin             BIN,
764     thumbs          [1] EXPLICIT Thumbs  OPTIONAL,
765     piRqExtensions  [2] MsgExtensions {{PIRqExtensionsIOS}}  OPTIONAL
766 }

324 RRPID ::= OCTET STRING(SIZE(20)) – Request response pair identification

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

232 BrandID ::= SETString { ub-BrandID }

250 BIN ::= NumericString (SIZE(6))          -- Bank identification number

330 Thumbs ::= SEQUENCE {
331     digestAlgorithm         AlgorithmIdentifier {{DigestAlgorithms}},
332     certThumbs              [0] EXPLICIT Digests  OPTIONAL,
333     crlThumbs               [1] EXPLICIT Digests  OPTIONAL,
334     brandCRLIdThumbs        [2] EXPLICIT Digests  OPTIONAL
335 }
```

Keterangan mengenai *field* yang terdapat di dalam **PInitReq** :

PInitReq	**{RRPID, Language, LID-C, [LID-M], Chall-C, BrandID, BIN, [Thumbs], [PIRqExtensions]}**
RRPID	Identitas pasangan *request/response*.
Language	Bahasa yang dipakai oleh *Cardholder.*
LID-C	Identitas local; suatu label yang dibuat oleh *Cardholder*
LID-M	Disalin dari proses inisiasi SET (kalau ada).
Chall-C	Variabel tantangan dari *Cardholder* untuk kesegaran tanda tangan *Merchant.* Dbuat secara acak.
BrandID	Merek kartu pembayaran yang dipilih oleh *Cardholder.*
BIN	*Bank Identification Number*, yaitu 6 digit pertama dari nomor kartu pembayaran milik *Cardholder.* Sebagai identitas dari bank yang mengeluarkan kartu pembayaran.
Thumbs	Nilai *hash* dari sertifikat, CRL, dan *BrandCRLIdentifier* yang sudah dimiliki oleh *Cardholder.*
PIRqExtensions	Keterangan tambahan untuk pesan ini. Data-data yang tersimpan dalam *field* ini tidak dienkrip, sehingga tidak boleh mengandung data-data yang sensitif.

Tabel III-14. PInitReq

Proses pembuatan **PInitReq** oleh *Cardholder*:

1) Buat **RRPID** secara acak untuk mengidentifikasi pesan dan pesan pasangannya.
2) Isi **Language** sesuai dengan bahasa yang dipakai *Cardholder.*
3) Buat **LID-C** sebagai identifikasi untuk setiap pasangan pesan, karena **XID** belum dibuat. *Field* ini boleh dibuat secara acak atau berurutan, tetapi tidak boleh diulang terlalu sering.
4) Kalau *Merchant* memberikan **LID-M** dalam inisialiasi SET (tahap sebelum protokol SET dimulai), maka salin *field* tersebut ke dalam pesan.
5) Buat suatu **Chall-C** yang baru secara acak.
6) Isi **BrandID** berdasarkan kartu pembayaran yang dimiliki *Cardholder.*
7) Isi **BIN** sesuai dengan kartu pembayaran yang dipakai oleh *Cardholder*.
8) Opsional: isi **Thumbs** dari sertifikat, CRL dan **BrandCRLIdentifier** yang sudah dimiliki oleh *Cardholder.* Jika *field* ini diisi, maka makin sedikit jumlah sertifikat, CRL,

dan **BrandCRLIdentifier** yang dikirim dalam **PInitRes**.

9) Simpan **RRPID, LID-C, LID-M** (jika ada)**, Chall-C, and Thumbs** (jika ada) dalam catatan transaksi.
10) Opsional: Isi semua ekstensi dari **PInitReq**
11) Bungkus pesan dalam *Message Wrapper* dan kirimkan ke *Merchant.*

Pesan **PInitReq** tidak dienkrip terlebih dahulu sebelum dikirim ke *Merchant.* Hal ini menunjukkan bahwa tidak ada data-data yang sensitif dalam pesan ini. Bila data ini disadap dalam perjalanan, tidak akan mempengaruhi proses pembayaran, karena penyadap tidak mendapatkan informasi yang berharga dari pesan ini. Kalau pesan ini mengalami perubahan isi di perjalanan, juga tidak apa-apa, karena kemungkinan terburuk adalah *Merchant* meminta untuk mengirim ulang **PInitReq**.

Merchant memproses **PInitReq** :

1) Terima **PInitReq** dari *Cardholder.*
2) Jika ada **LID-M**, cari transaksi dari catatan transaksi berdasarkan **LID-M**. Jika tidak ditemukan:
   a) Kembalikan sebuah pesan kesalahan dengan **ErrorCode** diset ke **unknownLID**.
   b) Hentikan memroses **PInitReq**
3) Jika **LID-M** tidak ada, cari transaksi berdasarkan kriteria lain yang tidak ada dalam ruang lingkup SET, atau jika *Merchant* belum membuat **LID-M** untuk transaksi ini, buat **LID-M** baru dan simpan di catatan transaksi.
4) Buat suatu **XID** baru.
5) Simpan **XID, RRPID, Language, LID-C, Chall-C, BrandID,** dan **BIN** di dalam catatan transaksi.
6) Jika ada **Thumbs**, simpan dalam catatan transaksi.
7) Jika ada ekstensi pesan, maka diproses. Jika ekstensi tidak dikenal, dan tanda kritis diset *True*, hasilkan pesan kesalahan, jika tidak, abaikan ekstensi. Jika ekstensi dikenal, maka diproses.

Spesifikasi **PInitRes** dalam *ASN.1*:

```
770 PInitRes ::= S { M, PInitResData }

772 PInitResData ::= SEQUENCE {
773     transIDs              TransIDs,
774     rrpid                 RRPID,
775     chall-C               Challenge,
776     chall-M               Challenge,
777     brandCRLIdentifier    [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
778     peThumb               [1] EXPLICIT CertThumb,
779     thumbs                [2] EXPLICIT Thumbs  OPTIONAL,
780     piRsExtensions        [3] MsgExtensions {{PIRsExtensionsIOS}}  OPTIONAL
781 }

337 TransIDs ::= SEQUENCE {
338     lid-C           LocalID,
339     lid-M           [0] LocalID  OPTIONAL,
340     xid             XID,
341     pReqDate        Date,
342     paySysID        [1] PaySysID  OPTIONAL,
343     language        Language        -- Cardholder requested session language
344 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

191 BrandCRLIdentifier ::= SIGNED {
192     EncodedBrandCRLID
193 } ( CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )

330 Thumbs ::= SEQUENCE {
331     digestAlgorithm         AlgorithmIdentifier {{DigestAlgorithms}},
332     certThumbs              [0] EXPLICIT Digests  OPTIONAL,
333     crlThumbs               [1] EXPLICIT Digests  OPTIONAL,
334     brandCRLIdThumbs        [2] EXPLICIT Digests  OPTIONAL
335 }
```

Keterangan mengenai *field* yang terdapat di dalam **PInitRes** *:*

PInitRes	**S(M, PinitResData)** Pesan ini ditandatangani oleh *Merchant* menggunakan operator **S**.
PInitResData	**{TransIDs, RRPID, Chall-C, Chall-M, [BrandCRLIdentifier], PEThumb, [Thumbs], [PIRsExtensions]}**
TransIDs	Identitas pesan
RRPID	Identitas pasangan *request/response*.
Chall-C	Identitas *Cardholder*, disalin dari **PInitReq**.

Tabel III-15. PInitRes

Chall-M	Variabel tantangan dari *Merchant* untuk kesegaran tandatangan *Cardholder*.
BrandCRLIdentifier	Daftar CRL yang dimiliki *Brand CA*.
PEThumb	*Thumbprint* dari sertifikat *Payment Gateway*.
Thumbs	Nilai *hash* dari sertifikat, CRL, dan *BrandCRLIdentifier* yang sudah dimiliki oleh *Cardholder*. Disalin dari **PInitReq**.
PIRsExtensions	Keterangan tambahan untuk pesan ini.

Tabel III-15. PInitRes (lanjutan)

Merchant membuat **PInitRes**:

1) Buat **PInitResData** sebagai berikut:
    a) Buat **TransIDs** sebagai berikut:
        1) Salin **LID-C**, **XID**, dan **Language** dari catatan transaksi.
        2) Salin **LID-M** jika ada pada catatan transaksi.
        3) Isi **TransIDs.PReqDate** dengan tanggal transaksi.
    b) Salin **RRPID** dari catatan transaksi.
    c) Salin **Chall-C** dari catatan transaksi.
    d) Buat **Chall-M** baru.
    e) Jika **Thumb** untuk **BrandCRLIdentifier** tidak diterima atau tidak ada, isi **BrandCRLIdentifier** dengan yang baru.
    f) Pilih *Payment Gateway* berdasarkan **BrandID**, **BIN**, dan sertifikat *Cardholder* dari **PInitReq**. Isi **PEThumb** dengan *thumbprint* sertifikat *Payment Gateway* yang dipilih.
    g) Salin **Thumbs** dari **PInitReq** jika ada. Berguna agar *Cardholder* bisa memeriksa apakah *Merchant* telah menerima semua **Thumbs** yang dikirimkan.
    h) Opsional: tambahkan **PIRsExtensions**.
2) **PInitResData** ditandatangani secara digital menggunakan operator **S**.
3) Bungkus pesan dalam *Message Wrapper* dan kirimkan ke *Cardholder.*

Cardholder memroses *PInitRes* :

1) Ambil data dari *MessageWrapper*.
2) Periksa tanda tangan *Merchant* yang terdapat pada operator **S** dan ambil **PInitResData**.
3) Periksa **TransIDs** dengan cara sebagai berikut:
    a) Cari transaksi berdasarkan **LID-C**. Kalau tidak ditemukan:
        1) Kirim pesan **Error** dengan **ErrorCode** diset ke **unknownLID**.
        2) Hentikan memroses **PInitRes**.
    b) Jika **LID-M** telah dikirim pada tahap inisiasi SET, periksa **LID-M** dengan **LID-M** pada catatan transaksi. Jika tidak sesuai:
        1) Kirim pesan **Error** dengan **ErrorCode** diset ke **unknownLID**.
        2) Hentikan memroses **PInitRes**.
    c) Jika **LID-M** tidak dikirim sebelumnya, padahal ada **LID-M**:
        1) Kirim pesan **Error** dengan **ErrorCode** diset ke **unknownLID**.
        2) Hentikan memroses **PInitRes**.
4) Periksa **RRPID** dengan **RRPID** yang ada pada catatan transaksi. Jika **RRPID** berbeda:
    a) Kirim pesan **Error** dengan **ErrorCode** diset ke **unknownRRPID**.
    b) Hentikan memroses **PInitRes**. Hal ini menunjukkan kesalahan pengiriman pesan, balasan pesan bukanlah pasangan permintaannya.
5) Periksa **Chall-C** dengan **Chall-C** yang ada pada catatan transaksi. Jika **Chall-C** berbeda:
    a) Kirim pesan **Error** dengan **ErrorCode** diset ke **challengeMismatch**.
    b) Hentikan memroses **PInitRes**.
6) Sebagai opsional, ambil nama *Merchant* dari sertifikat milik *Merchant* dan tampilkan ke pengguna. Lanjutkan proses jika pengguna menyetujui nama yang ditampilkan. Jika tidak, hentikan memroses **PInitRes.**
7) Simpan data, termasuk **TransIDs**, dan **Chall-M** pada catatan transaksi.

8) Proses **BrandCRLIdentifier** jika ada.

9) Gunakan **PEThumb** untuk mengidentifikasi sertifikat *Payment Gateway* (**Cert-PE**) untuk digunakan pada **PReq** untuk mengenkrip data yang ditujukan pada *Payment Gateway*.

3.11.2 PReq/PRes

Pasangan pesan ini merupakan inti dari sistem pembayaran SET. Pada pasangan pesan inilah data-data yang paling sensitif dipertukarkan, oleh karena itu mekanisme pengamanannya diperhatikan sekali oleh SET. Data-data sensitif tersebut adalah informasi kartu pembayaran yang dipergunakan oleh *Cardholder*.

Pasangan **PReq/PRes** boleh didahului atau tidak oleh **PInitReq/PInitRes**. **PRes** boleh dikirimkan sebelum proses otorisasi dan *capture*, implementasinya tergantung aturan yang diterapkan oleh *Brand*. Jika *Cardholder* memiliki sertifikat digital, maka teknik tanda tangan ganda dipakai untuk autentikasi dan pengecekan integritas data.

PReq terdiri dari dua bagian, yaitu *Payment Instruction (PI)* dan *Order Instruction (OI)*. PI dikirimkan ke *Merchant* untuk disampaikan ke *Payment Gateway* tanpa *Merchant* bisa mengetahui isinya.

Spesifikasi **PReq** dalam ASN.1 :

```
787 PReq ::= CHOICE {
788       pReqDualSigned          [0] EXPLICIT PReqDualSigned,
789       pReqUnsigned  [1] EXPLICIT PReqUnsigned
790 }
```

Keterangan mengenai *field* yang terdapat di dalam **PReq :**

PReq	**< PReqDualSigned, PReqUnsigned >** Ada dua macam, yaitu yang ditandatangani dan yang tidak.
PReqDualSigned	**PReq** dalam bentuk tertandatangani
PReqUnsigned	**PReq** yang tidak ditandatangani

Tabel III-16. PReq

Spesifikasi **PReqDualSigned** dalam ASN.1 :

```
794 PReqDualSigned ::= SEQUENCE {
795       piDualSigned     PIDualSigned,
796       oiDualSigned     OIDualSigned
```

797 }

809 OIDualSigned ::= L { OIData, PIData }

Keterangan mengenai *field* yang terdapat di dalam **PReqDualSigned** :

PReqDualSigned	**{PIDualSigned, OIDualSigned}** Bentuk **PReq** yang ditandatangani
PIDualSigned	Lihat halaman 36 mengenai **PI**
OIDualSigned	**L(OIData, PIData)** Referensi yang menunjukkan hubungan antara OI dengan PI.
OIData	Lihat tabel mengenai **OIData**.
PIData	**{PIHead, PANData}** **PANData** berisi informasi kartu pembayaran. Hubungan antara PI dan kartu pembayaran terdapat pada *field* ini.

Tabel III-17. PReqDualSigned

Tanda tangan ganda dari *Cardholder* terdapat pada **PISignature** yang terdapat di dalam **PIDualSigned** Tanda tangan ganda tersebut didapat dengan cara menandatangani *sequence* **{DD(PIData), DD(OIData)}**. *Merchant* memeriksa tanda tangan tersebut menggunakan **DD(PIData)** yang terdapat dalam **OIDualSigned** dan membuat **DD(OIData)** baru.

Spesifikasi **PReqUnsigned** dalam ASN.1 :

```
886 PReqUnsigned ::= SEQUENCE {  -- Sent by cardholders without certificates
887    piUnsigned  PIUnsigned,
888    oiUnsigned  OIUnsigned
889 }
```

898 PIUnsigned ::= EXH { P, PI-OILink, PANToken }

891 OIUnsigned ::= L { OIData, PIDataUnsigned }

Keterangan mengenai *field* yang terdapat di dalam **PReqUnsigned** :

PReqUnsigned	**{PIUnsigned, OIUnsigned}** Bentuk **PReq** yang tidak ditandatangani.
PIUnsigned	Lihat tabel mengenai **PIUnsigned**
OIUnsigned	**L(OIData, PIDataUnsigned)** Referensi yang menunjukkan hubungan antara PI dan OI.
OIData	Lihat tabel berikutnya mengenai **OIData**
PIDataUnsigned	**{PIHead, PANToken}** **PANToken** berisi informasi kartu pembayaran. Hubungan antara PI dengan kartu pembayaran terdapat pada *field* ini.

Tabel III-18. PReqUnsigned

Spesifikasi **OIData** dalam ASN.1 :

```
853 OIData ::= SEQUENCE {                    -- Order Information Data
854     transIDs        TransIDs,
855     rrpid           RRPID,
856     chall-C         Challenge,
857     hod             HOD,
858     odSalt          Nonce,
859     chall-M         Challenge  OPTIONAL,
860     brandID         BrandID,
861     bin             BIN,
862     odExtOIDs       [0] OIDList  OPTIONAL,
863     oiExtensions    [1] MsgExtensions {{OIExtensionsIOS}} OPTIONAL
864 }

337 TransIDs ::= SEQUENCE {
338     lid-C           LocalID,
339     lid-M           [0] LocalID  OPTIONAL,
340     xid             XID,
341     pReqDate        Date,
342     paySysID        [1] PaySysID  OPTIONAL,
343     language        Language        -- Cardholder requested session language
344 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

870 HOD ::= DD { HODInput }

232 BrandID ::= SETString { ub-BrandID }

250 BIN ::= NumericString (SIZE(6))          -- Bank identification number
872 HODInput ::= SEQUENCE {
873     od                      OD,
874     purchAmt                CurrencyAmount,
```

```
875     odSalt                  Nonce,
876     installRecurData        [0] InstallRecurData  OPTIONAL,
877     odExtensions            [1] MsgExtensions {{ODExtensionsIOS}} OPTIONAL
878 }

882 OD ::= OCTET STRING                       -- Order description

1945 InstallRecurData ::= SEQUENCE {
1946    installRecurInd         InstallRecurInd,
1947    irExtensions            [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }
```

Struktur data pada tabel berikut terdapat pada **PReqDualSigned**maupun **PReqUnsigned**

OIData	**{TransIDs, RRPID, Chall-C, HOD, ODSalt, [Chall-M], BrandID, BIN, [ODExtOIDs], [OIExtensions]}**
TransIDs	Jika **PInitRes** ada, maka **TransIDs** disalin dari **PInitRes**.
RRPID	Identifikasi pasangan pesan.
Chall-C	Disalin dari **PInitReq** yang berhubungan.
HOD	**DD(HODInput)** Nilai *hash* dari **HODInput**. Gunanya agar bisa menghubungkan suatu **OD** dengan **PurchAmt** tertentu**.** Lihat tabel berikutnya mengenai **HODInput.**
ODSalt	Disalin dari **HODInput**.
Chall-M	Variabel tantangan dari *Merchant* untuk menjamin kesegaran tandatangan *Cardholder*.
BrandID	Merek kartu pembayaran yang dipilih oleh *Cardholder.*
BIN	*Bank Identification Number*, yaitu 6 digit pertama dari nomor kartu pembayaran milik *Cardholder*.
ODExtOIDs	Daftar *object identifier* dari **ODExtensions** dengan urutan yang sama dengan kemunculan objek-objek tersebut pada **ODExtensions**.
OIExtensions	Berisi ekstensi dari OI; tergantung dari kebijakan *Merchant.* Data yang tersimpan dalam *field* ini tidak dienkrip, sehingga tidak boleh mengandung data-data yang sensitif.

Tabel III-18. OIData

HODInput	**{OD, PurchAmt, ODSalt, [InstallRecurData], [ODExtensions]}**
OD	*Order Description.* Informasi ini diberikan oleh *Cardholder* kepada *Merchant* sebelum proses SET dimulai (sebelum pesan **PInitReq**). Berisi data-data produk yang dibeli (jumlah, ukuran, harga,dsb.), alamat pengiriman, dan alamat penagihan. Jadi, ketika proses SET dimulai, *Merchant* sudah memiliki **OD** tersebut.
PurchAmt	Jumlah pembayaran untuk transaksi sekarang, sebagaimana yang telah disepakati oleh *Cardholder* dan *Merchant. Field* ini harus cocok dengan **PurchAmt** yang terdapat pada **PIHead.**
ODSalt	*Nonce* baru yang dibuat oleh *Cardholder* untuk mencegah *dictionary attack* pada **HOD**.
InstallRecurData	Lihat keterangan pada hal 39.
ODExtensions	Data-data yang terdapat pada ekstensi ini berhubungan dengan pemrosesan pesanan oleh *Merchant.* Isinya tergantung dari kebijakan *Brand*.

Tabel III-18. OIData (lanjutan)

Cardholder membuat **PReq**:

1) Ambil **PurchAmt** dan **OD** dari hasil tahap belanja (sebelum protokol SET dimulai).
2) Buat **OIData** sebagai berikut:
   a) Jika **PInitReq/PInitRes** telah dipertukarkan sebelumnya salin **TransIDs** dari **PInitRes**.

      Jika tidak: Untuk membuat **TransIDs**, *Cardholder* membuat **PReqDate** (waktu atau tanggal transaksi), **LID-C**, dan **XID**.

   b) Buat **RRPID** baru; simpan untuk mencocokkan jawaban dari *Merchant.*

      Jika **PInitReq/PInitRes** telah dipertukarkan sebelumnya: salin **Chall-C** dari **PInitRes**.

      Jika tidak: Buat **Chall-C** baru.

   c) Buat **ODSalt** baru.
   d) Buat **HODInput** sebagai berikut:
      1) Salin **OD** dari tahap inisiasi SET (tahap belanja).

2) Isi **PurchAmt** dengan jumlah yang telah disepakati oleh *Cardholder* selama tahap inisiasi SET.

3) Salin **ODSalt** dari langkah 2.c.

4) Jika *Cardholder* memilih untuk melakukan angsuran pembayaran atau langganan, isi **InstallRecurData** (lihat hal 39).

5) Opsional: tambahkan **ODExtenstions** .

e) Buat **HOD** menggunakan **HODInput** dari langkah 2.d.

f) Jika **PInitReq/PInitRes** telah dipertukarkan sebelumnya: salin **Chall-M** dari **PInitRes**. Jangan isi **BrandID**, karena *Merchant* sudah mendapatkan informasi tersebut dari **PInitReq**

Jika tidak: jangan isi **Chall-M**. Isi **BrandID** untuk menunjukkan kartu pembayaran yang dipakai.

g) Isi **BIN** dengan 6 digit pertama dari PAN milik *Cardholder*.

h) Jika **HODInput** dari langkah 2.d mempunyai **ODExtensions** , isi **ODExtOIDs** dengan semua **OIDs** dari **ODExtensions** . Urutan **ODExtOIDs** harus sama dengan urutan yang terdapat pada **ODExtensions** supaya *Merchant* dapat membuat *hash* yang sama.

i) Opsional: tambahkan **OIExtenstions** .

3) Buat **PIHead** sebagai berikut:

a) Salin **TransIDs** yang dihasilkan dari langkah 2.a..

b) Buat **Inputs** sebagai berikut:

1) Salin **HOD** dari langkah 2.e.

2) Salin **PurchAmt** dari langkah 2.d.2.

c) Salin **MerchantID** dari sertifikat milik *Merchant* yang terdapat pada **PInitRes**, atau dari media lain.

d) Salin **InstallRecurData** dari langkah 2.d.4 jika tersedia.

e) Buat **TransStain** menggunakan operator **HMAC** menggunakan **XID** sebagai datanya dan **CardSecret** sebagai kuncinya. Jika *Cardholder* tidak mempunyai

sertifikat,isi **CardSecret** dengan angka 0.

f) Isi **SWIdent**, yang didapat dari program yang dipakai *Cardholder*. Nilai ini harus sama dengan yang terdapat pada **MessageWrapper**.

g) Jika ekstensi *tunneling* dari **Cert_PE** ada, maka buat **AcqBackInfo** sebagai berikut:

1) Cari sebuah algoritma enkripsi yang dapat diterima oleh *Cardholder* pada ekstensi *tunneling*. Jika ditemukan, isi **AcqBackAlg**; jika tidak, **AcqBackInfo** tidak dibuat dan langsung ke langkah 4.

2) Buat sebuah **AcqBackKey** yang baru (sesuai dengan **AcqBackAlg**).

h) Opsional: tambahkan **PIExtentions** .

4) Buat **PANData** seperti dijelaskan pada halaman 41.

5) Buat **PI-OILink** menggunakan operator **L** dengan **PIHead** dari langkah 3 sebagai parameter t1 dan **OIData** dari langkah 2 sebagai parameter t2.

6) Menggunakan data-data yang didapat dari langkah 2, 3, dan 4, buat **PReqDualSigned** jika *Cardholder* memiliki sertifikat atau **PReqUnsigned** jika *Cardholder* tidak memiliki sertifikat.Jika sertifikat *Payment Gateway* mensyaratkan perlunya pesan-pesan yang ditandatangani dan *Cardholder* tidak memiliki sertifikat, maka program pada komputer *Cardholder* harus memberitahukan bahwa transaksi tidak dapat diteruskan. Pemberitahuan tersebut dikeluarkan sebelum menyusun **PReq**.

Jika *Cardholder* memiliki sertifikat digital, maka selesaikan **PReqDualSigned** sebagai berikut :

1) Buat **PISignature** sebagai berikut:

A. Buat PI-TBS sebagai berikut:

a) Buat **HPIData** sebagai nilai *hash* dari **PIData** menggunakan operator **H**.

b) Buat **HOIData** sebagai nilai *hash* dari **OIData** menggunakan operator **H**.

B. Selesaikan **PISignature** menggunakan operator **SO** dengan sertifikat *Cardholder* sebagai parameter s, dan **PI-TBS** sebagai parameter t.

2) Gunakan operator **EX** dengan kunci publik *Payment Gateway* sebagai parameter r, **PI-**

OILink dari proses pembuatan **PReq** langkah 5 sebagai parameter t, dan **PANData** dari proses pembuatan **PReq** langkah 4 sebagai parameter p.

3) Buat **PIDualSigned** sebagai gabungan dari **PISignature** yang didapat dari langkah 1 dan data yang terenkrip dari langkah 2.

4) Buat **PIData** sebagai gabungan dari **PIHead** yang didapat dari proses pembuatan **PReq** langkah 3 dan **PANData** yang didapat dari proses pembuatan **PReq** langkah 4.

5) Buat **OIDualSigned** menggunakan operator **L** dengan **OIData** yang didapat dari proses pembuatan **PReq** langkah 2 sebagai parameter t1 dan **PIData** dari langkah 4 sebagai parameter t2.

6) Buat **PReqDualSigned** sebagai gabungan dari **PIDualSigned** dari langkah 3 dan **OIDualSigned** dari langkah 5.

Jika *Cardholder* tidak memiliki sertifikat digital, maka selesaikan **PReqUnSigned** sebagai berikut :

1) Buat **PIUnsigned** menggunakan operator **EXH** dengan kunci publik *Payment Gateway* sebagai parameter r, **PI-OILink** yang didapat dari proses pembuatan **PReq** langkah 5 sebagai parameter t, dan **PANToken** yang didapat dari proses pembuatan **PReq** langkah 4.

2) Buat **PIDataUnsigned** sebagai *sequence* dari **PIHead** dan **PANToken**.

3) Buat **OIUnsigned** menggunakan operator **L** dengan **OIData** yang didapat dari proses pembuatan **PReq** langkah 2 sebagai parameter t1 dan **PIDataUnsigned** dari langkah 2 sebagai parameter t2.

4) Buat **PReqUnsigned** sebagai *sequence* dari **PIUnsigned** dari langkah 1 dan **OIUnsigned** dari langkah 3.

Merchant memroses **PReq** :

1) Terima **PReq** dari *Cardholder*

2) Jika yang diterima adalah **PReqDualSigned**, cek **PISignature** sebagai berikut:

   A. Ambil **OIData** dan **HPIData** dari **OIDualsigned**

B. Buat **HOIData** sebagai nilai *hash* dari **OIData**.

C. Buat **PI-TBS** sebagai *sequence* dari **HPIData** dari langkah A dan **HOIData** dari langkah B.

D. Buat tanda tangan menggunakan operator **SO** dengan sertifikat *Cardholder* sebagai parameter s, dan **PI-TBS** dari langkah C sebagai parameter t..

E. Bandingkan tanda tangan yang dihasilkan dari langkah D dengan **PISignature**. Jika tidak sama, maka kembalikan pesan **Error** dengan **ErrorCode** diset ke **signatureFailure** . Hentikan memroses **PReq**.

F. Lakukan langkah 4.

3) Jika yang diterima **PReqUnsigned**, cek apakah Cert-PE memperbolehkan **PReqUnsigned**. Jika tidak :

A. Kembalikan **PRes** dengan **CompletionCode** diset ke **signatureRequired**

B. Hentikan memroses **PReq**.

4) Proses **TransIDs** sebagai berikut:

Cari transakasi dari catatan transaksi berdasarkan **XID**.

Jika ditemukan, cek **LID-C** dan **LID-M** dengan transaksi yang ditemukan tersebut.

Jika tidak cocok:

a) Kembalikan pesan **Error** dengan **ErrorCode** diset ke **unknownLID**.

b) Hentikan memroses **PReq**.

Jika cocok, cek **Chall-M** dengan transaksi yang ditemukan tersebut. Jika tidak cocok:

a) Kembalikan pesan **Error** dengan **ErrorCode** diset ke **challengeMismatch**.

b) Hentikan memroses **PReq**.

Jika tidak :

a) Buat catatan transaksi baru.

b) Simpan **LID-C**, **PReqDate** dan **Language**.

c) Opsional: buat **LID-M**.

5) Cek apakah **BrandID** didalam sertifikat *Cardholder* sama dengan **BrandID** di dalam

PInitReq atau **OIData**. Jika tidak sama:

a) Kembalikan **PRes** dengan **completionCode** diset ke **orderRejected**.

b) Hentikan memroses **PReq**.

6) Simpan **Chall-C** untuk nantinya dikembalikan di dalam **PRes**.

7) Simpan *field* lainnya di dalam catatan transaksi.

8) Cek **HOD** dengan *hash* dari **OD**, **PurchAmt**, **ODSalt**, **InstallRecurData** (jika ada) dan **ODExtensions** (jika ada).

9) Setelah sampai langkah ini, *Merchant* bisa saja mengirim **PRes** ke *Cardholder*, atau menunggu sampai setelah **AuthRes** diterima dari *Payment Gateway.*

Spesifikasi **PRes** dalam ASN.1 :

```
903 PRes ::= S { M, PResData }

905 PResData ::= SEQUENCE {
906     transIDs                TransIDs,
907     rrpid                   RRPID,
908     chall-C                 Challenge,
909     brandCRLIdentifier      [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
910     pResPayloadSeq          PResPayloadSeq
911 }
```

Keterangan mengenai *field* yang terdapat di dalam **PRes** :

PRes	**S(M, PResData)**
PResData	**{TransIDs, RRPID, Chall-C, [BrandCRLIdentifier], PResPayloadSeq}**
TransIDs	Disalin dari **TransIDs** yang terdapat pada **PReq**
RRPID	Identitas pasangan pesan.
Chall-C	Disalin dari **PInitReq** yang berhubungan.
BrandCRLIdentifier	Daftar CRL yang dimiliki *Brand CA.*
PResPayloadSeq	**{PResPayload +}** Satu **PResPayload** untuk setiap otorisasi yang telah dilakukan. Jika otorisasi belum dilakukan, maka diisi dengan sebuah **PResPayload** dan statusnya (**CompletionCode**) diset sesuai kondisi.
PResPayload	Lihat tabel berikutnya

Tabel III-19. PRes

Spesifikasi **PResPayload** dalam ASN.1 :

```
915 PResPayload ::= SEQUENCE {
916     completionCode          CompletionCode,
917     results                 Results  OPTIONAL,
918     pRsExtensions           [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
919 }


923 CompletionCode ::= ENUMERATED {
924     meaninglessRatio        (0),  -- PurchAmt = 0; ratio cannot be computed
925     orderRejected           (1),  -- Merchant cannot process order
926     orderReceived           (2),  -- No processing to report
927     orderNotReceived        (3),  -- InqReq received without PReq
928     authorizationPerformed (4),  -- See AuthStatus for details
929     capturePerformed        (5),  -- See CapStatus for details
930     creditPerformed         (6)   -- See CreditStatus for details
931 }

933 Results ::= SEQUENCE {
934     acqCardMsg    [0] EXPLICIT AcqCardMsg  OPTIONAL,
935     authStatus    [1] AuthStatus  OPTIONAL,
936     capStatus     [2] CapStatus  OPTIONAL,
937     credStatusSeq  [3] CreditStatusSeq  OPTIONAL
938 }

1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }

940 AuthStatus ::= SEQUENCE {
941     authDate        Date,
942     authCode        AuthCode,
943     authRatio       FloatingPoint,
944     currConv        [0] CurrConv  OPTIONAL
945 }

947 CapStatus ::= SEQUENCE {
948     capDate         Date,
949     capCode         CapCode,
950     capRatio        FloatingPoint
951 }

1142 AuthCode ::= ENUMERATED {
1143     approved                ( 0),
1144     unspecifiedFailure      ( 1),
1145     declined                ( 2),
1146     noReply                 ( 3),
1147     callIssuer              ( 4),
1148     amountError             ( 5),
1149     expiredCard             ( 6),
1150     InvalidTransaction      ( 7),
1151     systemError             ( 8),
1152     piPreviouslyUsed        ( 9),
1153     recurringTooSoon        (10),
1154     recurringExpired        (11),
```

```
1155    piAuthMismatch              (12),
1156    installRecurMismatch        (13),
1157    captureNotSupported         (14),
1158    signatureRequired           (15),
1159    cardMerchBrandMismatch      (16)
1160 }

955 CreditStatus ::= SEQUENCE {
956    creditDate          Date,
957    creditCode          CapRevOrCredCode,
958    creditRatio         FloatingPoint
959 }

1394 CapCode ::= ENUMERATED {
1395    success                 (0),
1396    unspecifiedFailure      (1),
1397    duplicateRequest        (2),
1398    authExpired             (3),
1399    authDataMissing         (4),
1400    invalidAuthData         (5),
1401    capTokenMissing         (6),
1402    invalidCapToken         (7),
1403    batchUnknown            (8),
1404    batchClosed             (9),
1405    unknownXID              (10),
1406    unknownLID              (11)
1407 }
```

Keterangan mengenai *field* yang terdapat di dalam **PResPayload** :

PResPayload	**{CompletionCode, [Results], [PRsExtensions]}**
CompletionCode	Menunjukkan status dari transaksi
Results	**{[AcqCardMsg], [AuthStatus], [CapStatus], [CredStatusSeq]}**
PrsExtensions	*Field* ini tidak dienkrip, sehingga tidak boleh mengandung data yang sensitif.
AcqCardMsg	Disalin dari **AuthRes**.
AuthStatus	**{AuthDate, AuthCode, AuthRatio, [CurrConv]}**
CapStatus	**{CapDate, CapCode, CapRatio}** *Field* ini muncul jika **CapReq** yang berhubungan dengan proses otorisasi telah diproses.
CredStatusSeq	**{CreditStatus +}** *Field* ini muncul jika **CredReq** yang berhubungan dengan proses otorisasi telah diproses.
AuthDate	Tanggal proses otorisasi dilakukan;disalin dari **AuthRRTags.Date**

Tabel III-20. PresPayload

AuthCode	Kode yang menunjukkan hasil dari proses otorisasi*;* disalin dari **AuthResPayload**.
AuthRatio	**AuthReqAmt ÷ PurchAmt** **AuthReqAmt** didapat dari **AuthReqPayload** **PurchAmt** didapat dari **OIData**.
CurrConv	**{CurrConvRate, CardCurr}** Informasi konversi mata uang, disalin dari **AuthResPayload.**
CapDate	Tanggal dilakukannya proses *capture*, disalin dari **CapPayload.**
CapCode	Kode yang menunjukkan status dari proses *capture*, disalin dari **CapResPayload**.
CapRatio	**CapReqAmt ÷ PurchAmt** **CapReqAmt** didapat dari **CapPayLoad** **PurchAmt** didapat dari **OIData.**
CreditStatus	**{CreditDate, CreditCode, CreditRatio}** Informasi dari hasil proses *credit*. *Field* ini muncul jika **CreditReq** telah diproses.
CreditDate	Tanggal dilakukanya proses *credit*, disalin dari **CapRevOrCredReqData.**
CreditCode	Kode yang menunjukkan status dari proses *credit*, disalin dari **CapRevOrCredResPayload**
CreditRatio	**CapRevOrCredReqAmt ÷ PurchAmt** **CapRevOrCredReqAmt** didapat dari **CapRevOrCredReqData**. **PurchAmt** didapat dari **OIData**.

Tabel III-20. PResPayload (lanjutan)

meaninglessRatio	**PurchAmt** = 0; rasio tidak bisa dihitung.
orderRejected	*Merchant* tidak bisa memroses pesanan.
orderReceived	Otorisasi kartu pembayaran belum dilakukan.
orderNotReceived	*Inquiry* (pemeriksaan status transaksi) diterima sebelum ada pesanan.
authorizationPerformed	Otorisasi sudah dilakukan, lihat **AuthStatus** untuk detilnya.
capturePerformed	*Capture* sudah dilakukan, lihat **CapStatus** untuk detilnya.
creditPerformed	*Credit* sudah dilakukan, lihat **CreditStatus** untuk detilnya.

Tabel III-21. Nilai untuk CompletionCode

Merchant membuat **PRes** :

1) Mulai membuat **PRes** :

   a) Buat **PResData** sebagai berikut:

   b) Isi **TransIDs** semua *field* yang terdapat pada **TransIDs** yang diterima dari *Cardholder* atau *Payment Gateway*.

   c) Salin **RRPID** dari **PReq** atau **InqReq**

   d) Salin **Chall-C** dari **PReq** atau **InqReq**.

   e) Jika **Thumb** untuk **BrandCRLIdentifier** belum diterima, isi **BrandCRLIdentifier** dengan yang sekarang.

   f) Buat **PResPayloadSeq** sebagai berikut:

      1) Jika **PReq** mengandung **PurchAmt** yang nilainya 0, buat **PResPayload** dengan **CompletionCode** diset ke **meaninglessRatio** dan semua *field* lainnya dikosongkan. Masuk ke langkah 2.

      2) Jika *Payment Gateway* menolak transaksi, buat **PResPayload** dengan ketentuan:

         - Set **CompletionCode** ke **orderRejected**.
         - Salin **AcqCardMsg** dari **AuthRes** (jika ada).
         - Masuk ke langkah 2.

      3) Jika *Payment Gateway* belum memberikan jawaban terhadap permintaan otorisasi oleh *Merchant*, buat **PResPayload** dengan **CompletionCode** diset ke **orderReceived** dan *field* lainnya dikosongkan. Masuk ke langkah 2.

      4) Jika pesan ini merupakan jawaban dari pesan **InqReq**, dan transaksi tidak ditemukan dalam catatan transaksi, buat **PResPayload** dengan **CompletionCode** diset ke **orderNotReceived** dan *field* lainnya dikosongkan. Masuk ke langkah 2.

      5) Jika *Payment Gateway* telah menjawab permintaan otorisasi kartu pembayaran oleh *Merchant*, buat **PResPayloadSeq** seperti dijelaskan pada proses berikutnya.

2) Tandatangani pesan ini menggunakan operator **S**.

3) Bungkus pesan ini dalam **MessageWrapper** dan kirim ke *Cardholder*.

Proses pembuatan **PResPayloadSeq** :

1) Jika otorisasi telah dilakukan :
    a) Set **CompletionCode** ke **authorizationPerformed**.
    b) Buat **Results** seperti dijelaskan pada proses pembuatan **Results**, *field* **CapStatus** dan **CredStatusSeq** dikosongkan.
2) Jika proses *capture* telah dilakukan :
    a) Set **CompletionCode** ke **capturePerformed**.
    b) Buat **Results** seperti dijelaskan pada proses pembuatan **Results**, *field* **CredStatusSeq** dikosongkan.
3) Jika proses *credit* telah dilakukan:
    a) Set **CompletionCode** ke **creditPerformed**.
    b) Buat **Results** seperti dijelaskan pada proses pembuatan **Results**.
4) Opsional: tambahkan **PrsExtensions**.

Proses pembuatan **Result** :

1) Salin **AcqCardMsg** dari **AuthRes** jika ada.
2) Jika transaksi telah diotorisasi, buat **AuthStatus** sebagai berikut:
    a) Salin **AuthDate** dari catatan transaksi.
    b) Salin **AuthCode** dari catatan transaksi.
    c) Hitung **AuthRatio** dari **AuthReqAmt** ÷ **PurchAmt**.
    d) Salin data konversi mata uang jika ada di dalam **AuthRes**.
3) Jika transaksi telah mengalami proses *capture*, buat **CapStatus** sebagai berikut:
    a) Salin **CapDate** dari catatan transaksi.
    b) Salin **CapCode** dari catatan transaksi.
    c) Hitung **CapRatio** dari **CapReqAmt** ÷ **PurchAmt**.

4) Buat **CredStatusSeq** sebagai *sequence* dari **CredStatus** dari setiap proses *credit* yang telah dilakukan dan tidak dikembalikan. Buat **CredStatus** sebagai berikut:
    a) Salin **CreditDate** dari catatan transaksi.
    b) Salin **CreditCode** dari catatan transaksi.
    c) Hitung **CreditRatio** dari **CapRevOrCredReqAmt** ÷ **PurchAmt**.

Cardholder memroses **PRes** :

1) Terima **PRes** dari *Merchant.*
2) Periksa tanda tangan *Merchant* dari **PRes**.
3) Cari transakasi dari catatan transaksi berdasarkan **Trans.LID-C**. Jika tidak ditemukan:
    a) Kirimkan pesan **Error** dengan **ErrorCode** diset ke **unknownLID**.
    b) Hentikan memroses **PRes**.
4) Cek **TransIDs.XID** dengan **XID** yang terdapat di catatan transaksi. Jika tidak cocok:
    a) Kirimkan pesan **Error** dengan **ErrorCode** diset ke **unknownXID**.
    b) Hentikan memroses **PRes**.
5) Cocokkan **RRPID** dengan **RRPID** yang terdapat di catatan transaksi. Jika tidak cocok:
    a) Kirimkan pesan **Error** dengan **ErrorCode** diset ke **unknownRRPID**.
    b) Hentikan memroses **PRes**.
6) Cocokkan **Chall-C** dengan **Chall-C** yang terdapat di catatan transaksi. Jika tidak cocok:
    a) Kirimkan pesan **Error** dengan **ErrorCode** diset ke **challengeMismatch**.
    b) Hentikan memroses **PRes**.
7) Simpan **BrandCRLIdentifier** dan periksa apakah sertifikat yang dipakai untuk menandatangani pesan terdaftar dalam CRL, jika ya, maka hentikan memroses pesan, karena sertifikat tersebut sudah tidak valid.
8) Untuk setiap **PResPayload** di dalam **PResPayloadSeq** lakukan hal-hal berikut:
    a) Jika **CompletionCode** menunjukkan proses *credit* telah diselesaikan, maka untuk

setiap struktur data di dalam **CreditSeq** :

- hitung *Credit Amount* dari **PurchAmt** x **CredRatio**
- laporkan **CreditDate** dan **CapCode** ke pengguna
- hitung *Capture Amount* dari **PurchAmt** x **CapRatio**

b) Jika **CompletionCode** menunjukkan proses *capture* telah diselesaikan, maka:

- laporkan **CapCode** ke pengguna
- hitung *Capture Amount* dari **PurchAmt** x **CapRatio**

c) Jika **CompletionCode** menunjukkan proses otorisasi telah diselesaikan, maka:

- laporkan **AuthCode** ke pengguna
- hitung *Authorization Amount* dari **PurchAmt** x **AuthRatio**.

d) Jika bukan salah satu dari tiga poin di atas, maka laporkan hasil transaksi berdasarkan **CompletionCode**.

e) Jika ada **AcqCardMsg**, dekrip dan berikan ke *Cardholder*.

3.11.3 InqReq/InqRes

Pasangan pesan ini berguna untuk memeriksa status dari transakasi. Pasangan pesan ini dapat dipertukarkan berulang-ulang, dan hanya bisa dilakukan setelah *Cardholder* mengirim **PReq**. Pasangan pesan ini bersifat opsional.

Setiap pasangan pesan ini harus mempunyai **TransIDs** yang sama dengan **TransIDs** dari transaksi yang ingin diperiksa. Sertifikat yang dipakai untuk menandatangani **InqReq** harus sama dengan sertifikat yang dipakai untuk menandatangani **PReq**. Hal ini untuk mencegah *Cardholder* dapat memeriksa status transaksi yang bukan miliknya.

Spesifikasi **InqReq** dalam ASN.1 :

```
963 InqReq ::= CHOICE {
964     inqReqSigned            [0] EXPLICIT InqReqSigned,
965     inqReqUnsigned          [1] EXPLICIT InqReqData
966 }

968 InqReqSigned ::= S { C, InqReqData }

970 InqReqData ::= SEQUENCE {           -- Signed by cardholder, if signed
```

```
971     transIDs                TransIDs,
972     rrpid                   RRPID,
973     chall-C2                Challenge,
974     inqRqExtensions         [0] MsgExtensions {{InqRqExtensionsIOS}}  OPTIONAL
975 }
```

Keterangan mengenai *field* yang terdapat di dalam **InqReq**:

InqReq	**< InqReqSigned, InqReqData >**
InqReqSigned	**S(C, InqReqData)**
InqReqData	**{TransIDs, RRPID, Chall-C2, [InqRqExtensions]}**
TransIDs	Disalin dari pesan yang diterima paling akhir dari pesan-pesan berikut: **PReq**, **PRes**, **InqRes**.
RRPID	Identitas untuk pasangan pesan ini.
Chall-C2	Variabel tantangan dari *Cardholder* untuk kesegaran tanda tangan *Merchant.*
InqRqExtensions	Pesan ini tidak dienkrip, sehingga ekstensi dari pesan ini tidak boleh mengandung data yang sensitif.

Tabel III-21. InqReq

Jika *Cardholder* memiliki sertifikat, maka yang dikirim adalah **InqReqSigned**, jika tidak maka yang dikirim adalah **InqReqData**.

Cardholder membuat **InqReq**:

1) Buat **InqReqData** sebagai berikut:
   a) Salin **TransIDs** dari **PReq** sebelumnya.
   b) Buat **RRPID** baru.
   c) Buat **Chall-C** baru.
   d) Opsional: tambahkan **InqRqExtentions**.
2) Jika *Cardholder* mengirimkan **PReq** yang ditandatangani, tanda tangani **InqReqData**.
3) Bungkus pesan dalam **MessageWrapper** dan kirimkan ke *Merchant.*

Merchant memroses **InqReq** :

1) Terima **InqReq** dari *Cardholder*.
2) Jika yang diterima adalah **InqReqData** (**InqReq** yang tidak ditandatangani), cek apakah

sertifikat *Payment Gateway* memperbolehkan transaksi yang tidak ditandatangani. Jika tidak boleh , maka:

a) Kembalikan pesan **InqRes** dengan **CompletionCode** diset ke **signatureRequired**.

b) Hentikan memroses **InqReq**.

Jika boleh, maka masuk ke langkah 4.

3) Jika yang diterima adalah **InqReqSigned**, cek tanda tangannya. Jika tanda tangan tidak valid:

a) Kembalikan pesan **Error** dengan **ErrorCode** diset ke **signatureFailure** .

b) Hentikan memroses **InqReq**.

4) Cek **TransIDs** dengan **TransIDs** pada **MessageWrapper**. Jika tidak sama:

a) Kembalikan pesan **Error** dengan **ErrorCode** diset ke **wrapperMsgMismatch**.

b) Hentikan memroses **InqReq**.

5) Cari transaksi pada catatan transakasi berdasarkan **TransIDs.XID**. Jika tidak ditemukan:

a) Kembalikan **InqRes** dengan **CompletionCode** diset ke **orderNotReceived**

b) Hentikan memroses **InqReq**.

6) Jika **PReq** ditandatangani, maka cek apakah *Cardholder* yang sama yang telah menandatangani **PReq** dan **InqReq**. Jika tidak:

a) Kembalikan pesan **Error** dengan **ErrorCode** diset ke **unknownXID**.

b) Hentikan memroses **InqReq**.

7) Buat **PResPayloadSeq**

Isi dan cara pembuatan **InqRes** sama dengan isi dan cara pembuatan **PRes.** Pemrosesan **InqRes** oleh *Cardholder* sama juga dengan pemrosesan **PRes**.

BAB IV
ANALISA DAN PERANCANGAN

Dalam bab ini dibahas mengenai analisa penulis dalam mengimplementasi SET dan dilanjutkan dengan perancangan objek-objek yang diperlukan dalam implementasi SET.

4.1 SET MENJAWAB KEBUTUHAN BISNIS

Seperti telah disebutkan pada bab 2 mengenai kebutuhan bisnis yang diperlukan untuk perdagangan yang aman di Internet, berikutnya akan dibahas bagaimana cara SET menjawab empat kebutuhan tersebut.

4.1.1 Kerahasiaan Data (*Confidentiality*)

Sebagaimana telah disebutkan sebelumnya bahwa Internet adalah jalur komunikasi yang tidak aman. Siapa saja dengan keahlian dan peralatan yang memadai dapat menyadap informasi yang lewat di Internet. Bahkan data yang disadap tersebut dapat diubah oleh orang-orang tersebut. Oleh karena itu kerahasiaan data perlu dijaga, jangan sampai orang-orang yang tidak berhak bisa melihat isi data yang sensitif dari pesan-pesan SET.

Untuk data pembayaran, SET menggunakan teknik amplop digital seperti yang sudah dijelaskan pada bab 2. Untuk data-data yang bukan data pembayaran, tapi kerahasiaan data tersebut diperlukan juga, SET memakai referensi dari data tersebut dan bukannya data itu sendiri. Sebagai contoh, SET tidak mempertukarkan OI , tapi mengikutsertakan *hash* dari OI di dalam *Purchase Request* (**PReq**).

SET menggunakan algoritma enkripsi DES dan RSA untuk membuat amplop digital. Enkripsi kunci simetrik menggunakan algoritma DES, dan algoritma RSA digunakan untuk mengenkrip kunci simetrik tadi. Panjang kunci RSA yang disyaratkan oleh SET adalah 1024 bit untuk *Cardholder, Merchant,* dan *Payment Gateway*.

4.1.2 Autentikasi Pihak-Pihak Yang Terlibat (*Authentication*)

Autentikasi data menjamin bahwa data yang dikirim adalah benar-benar dikirim oleh

pihak yang dimaksud. Misalnya data dari *Cardholder* benar-benar dikirim oleh *Cardholder* yang dimaksud, demikian pula halnya dengan data yang dikirim oleh *Merchant* dan *Payment Gateway*. SET menjamin autentikasi data menggunakan teknik tanda tangan digital dan sertifikat digital.

a. Autentikasi Entitas

Tanda tangan digital mensyaratkan bahwa kunci publik yang dipakai untuk memeriksa tanda tangan digital adalah benar-benar milik entitas yang menandatanganinya. Untuk itu diperlukan suatu pihak ketiga yang menerbitkan sertifikat digital yang menyatakan bahwa kunci publik tersebut benar-benar milik entitas yang bersangkutan. Sertifikat digital ini disimpan di dalam komputer milik masing-masing entitas. Penerima pesan menggunakan sertifikat digital untuk memeriksa kebenaran kunci publik pengirim.

Oleh karena itu, keunikan dari tanda tangan digital dan nilai *hash* di bawahnya, dikombinasikan dengan sertifikat digital dapat menjamin autentikasi pengirim.

b. *Autentikasi* Cardholder

Merchant dan *Acquirer* akan memeriksa bahwa *Cardholder* memakai nomor kartu pembayaran yang valid. SET memakai suatu mekanisme yang menghubungkan nomor kartu pembayaran dengan identitas *Cardholder*. Mekanisme tersebut adalah dengan cara menghubungkan nomor kartu pembayaran dengan kunci publik *Cardholder.* Hubungan ini terdapat pada sertifikat digital yang dimiliki oleh *Cardholder*.

c. *Autentikasi* Merchant

Suatu *Merchant* tertentu menerima verifikasi dari *Acquirer* melalui penerbitan sertifikat digital. *Acquirer* menerima permintaan sertifikat dari *Merchant* dan memberikan sertifikat digital tersebut melalui *Merchant Certificate Authority* (MCA). Sertifikat tersebut menjamin bahwa *Merchant* mempunyai suatu kesepakatan yang sah dengan *Acquirer*, sehingga dapat menerima kartu pembayaran untuk melakukan transaksi.

Cardholder dan *Payment Gateway* mengautentikasi *Merchant* dengan cara memverifikasi tanda tangan digital yang terdapat pada sertifikat digital dan menelusuri hirarki sertifikat digital.

d. Autentikasi Payment Gateway

Sertifikat digital milik *Payment Gateway* diterbitkan oleh *Payment Gateway Certificate Authority* (PCA) milik *Brand*. PCA tersebut memeriksa kebenaran dari sertifikat *Acquirer* sebelum menerbitkan sertifikat digital untuk *Payment Gateway.* Dan karena *Payment Gateway* dioperasikan oleh *Acquirer*, maka sertifikasi tersebut dapat menjamin bahwa *Payment Gateway* sudah disahkan oleh *Brand* dan *Acquirer*.

Merchant mengautentikasi *Payment Gateway* dengan cara memverifikasi tanda tangan digital yang terdapat pada sertifikat digital dan menelusuri hirarki sertifikat digital.

Cardholder tidak perlu mengautentikasi *Payment Gateway*, karena tidak ada komunikasi antara *Cardholder* dan *Payment Gateway.* Tapi *Cardholder* memerlukan sertifikat digital milik *Payment Gateway* untuk mengenkrip PI sehingga hanya *Payment Gateway* yang bisa membaca isi PI. Sertifikat digital tersebut disediakan oleh *Merchant* dan diverifikasi oleh *Cardholder* untuk agar yakin bahwa sertifikat tersebut benar milik *Payment Gateway*.

SET menggunakan algoritma RSA dan SHA-1 untuk membuat tanda tangan digital, serta standar X.509 untuk sertifikat digital.

4.1.3 Keutuhan Data (*Integrity*)

Keutuhan data adalah jaminan bahwa data yang dikirim tidak mengalami perubahan selama perjalanan dari pengirim ke penerima. Keutuhan data tersebut dijamin menggunakan fungsi *hash*.

Fungsi *hash* yang dipakai oleh SET adalah fungsi *hash* menggunakan algoritma SHA-1.

4.1.4 Interoperability

Untuk menjamin *interoperability* antara implementasi SET yang dibuat oleh satu *vendor* dengan *vendor* lainnya, SET menggunakan menggunakan protokol dan format pesan yang spesifik. Pesan-pesan SET didefinisikan menggunakan satndar ASN.1 dan pengkodeannya menggunakan aturan DER. Pemakaian ASN.1 dimaksudkan agar tidak terjadi ambiguitas dalam pendefinisian pesan-pesan dan data-data SET. DER memastikan bahwa pengkodean data dilakukan secara tepat sampai setiap bit-bitnya dan hanya ada satu macam pengkodean data.

SET memakai algoritma-algoritma kriptografi yang terbuka, maksudnya bukanlah

milik suatu perusahaan atau badan tertentu dan bebas diimplementasikan oleh siapa saja. Algoritma-algoritma yang dipakai tersebut sudah menjadi standar dan sudah dipakai secara luas, yaitu RSA untuk enkripsi asimetrik, DES untuk enkripsi simetrik, dan SHA-1 untuk fungsi *hash*.

SET memakai standar yang spesifik untuk pembungkusan data, yaitu standar PKCS #7 dari RSA Lab. Standar PKCS #7 yang dipakai dalam SET:

- **SignedData** untuk membungkus data yang ditandatangani.
- **EnvelopedData** untuk membungkus data yang dienkripsi.
- **EncryptedData** untuk membungkus data yang dienkripsi secara simetrik.
- **DigestedData** untuk membungkus data yang di-*hash.*

SET memakai kartu pembayaran yang sudah dipakai secara luas, seperti kartu kredit. Dan untuk sertifikat digital, SET memakai standar X.509.

4.2 PKCS #7

PKCS #7 adalah standar yang dibuat oleh *RSA Laboratories*, dimaksudkan untuk standarisasi sintaks data-data yang mengandung kriptografi seperti tanda tangan digital dan amplop digital [PKCS #7]. Pada PKCS #7 didefinisikan 6 macam struktur data, yaitu: *data, signed data, enveloped data, signed and enveloped data, digested data,* dan *encrypted data*. Yang dipakai oleh SET dan yang akan dibahas dalam penelitian hanyalah 4 macam seperti yang sudah disebutkan pada sub-bab sebelumnya.

4.2.1 SignedData

Struktur data **SignedData** berguna untuk menyimpan data dan tanda tangan digitalnya. **SignedData** terdiri dari isi data yang akan ditandatangani serta *digest* yang dienkrip dari data tersebut. *Digest* yang dienkrip itu adalah tanda tangan digital dari data. **SignedData** memungkinkan penyimpanan lebih dari satu tanda tangan oleh lebih dari satu penandatangan.

Langkah-langkah untuk membuat **SignedData**:

1. Untuk masing-masing penandatangan, buat *digest* dari data yang akan ditandatangani. Algoritma untuk membuat *digest* dari data dapat berbeda-beda untuk masing-masing penandatangan. Jika algoritma untuk membuat *digest* tersebut sama untuk setiap penandatngan, maka *digest* dari data cukup dibuat sekali saja.

2. Untuk masing-masing penandatangan, enkrip *digest* dari data tadi menggunakan kunci privat masing-masing. Hasil enkripsi ini yang menjadi tanda tangan digital
3. Untuk masing-masing penandatangan, simpan *digest* yang dienkrip beserta informasi mengenai penandantangan di dalam struktur **SignerInfo**.
4. Simpan data yang ditandatangani beserta **SignerInfo** di dalam struktur **SignedData**.

Pihak penerima data memverifikasi tanda tangan digital dengan cara mendekrip tanda tangan digital menggunakan kunci publik penandatangan, dan membandingkannya dengan *digest* yang dihasilkan dari data yang diterimanya. Kunci publik penandatangan bisa didapatkan dari sertifikat digital yang disimpan di dalam **SignedData** atau dari sertifikat digital yang sudah dimilik penerima dengan mencocokkan nama dan nomor seri sertifikat untuk mendapatkan kunci publik yang sesuai.

Spesifikasi **SignedData** dalam kode *ASN.1* :

```
SignedData ::= SEQUENCE {
        version                 Version,
        digestAlgorithms        DigestAlgorithmIdentifiers,
        contentInfo             ContentInfo,
        certificates            [0] IMPLICIT ExtendedCertificatesAndCertificates
                                OPTIONAL,
        crls                    [1] IMPLICIT CertificateRevocationLists OPTIONAL,
        signerInfos             SignerInfos }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

SignerInfos ::= SET OF SignerInfo
```

Keterangan untuk masing-masing *field* yang terdapat dalam **SignedData** :

- **version** berisi nomor versi dari **SignedData**
- **digestAlgorithms** berisi algoritma-algoritma yang dipakai untuk menghasilkan *digest* dari data. Untuk keperluan SET, maka diisi dengan SHA-1.
- **contentInfo** berisi data yang akan ditandatangani
- **certificates** berisi sertifikat digital milik penandatangan atau sertifikat tambahan yang tidak dipakai untuk memeriksa tanda tangan data yang dikirim sekarang. *Field* ini bersifat opsional, jika ada maka kunci publik didapat dari sertifikat-sertifikat tersebut, jika tidak maka kunci publik didapat dari sertifikat yang sudah dimiliki oleh penerima.
- **crls** berisi daftar sertifikat yang sudah tidak berlaku, berguna untuk menentukan apakah sertifikat yang terdapat dalam **certificates** masih berlaku atau tidak. Jika

masih berlaku, maka validasi sertifikat dilakukan dengan cara penelusuran hirarki otoritas sertifikat.

- **signerInfos** berisi informasi mengenai penandatangan beserta tanda tangan digital yang dihasilkan oleh penandatangan tersebut untuk data yang ditandatangani.

Spesifikasi **SignerInfo** dalam kode *ASN.1* :

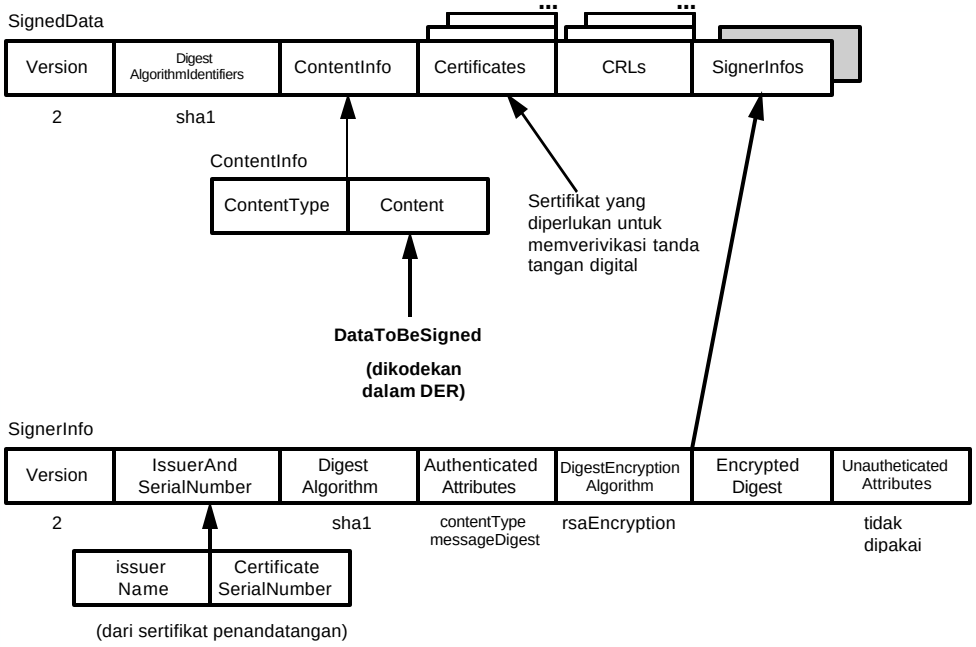
```
SignerInfo ::= SEQUENCE {
        version                     Version,
        issuerAndSerialNumber       IssuerAndSerialNumber,
        digestAlgorithm             DigestAlgorithmIdentifier,
        authenticatedAttributes     [0] IMPLICIT Attributes OPTIONAL,
        digestEncryptionAlgorithm   DigestEncryptionAlgorithmIdentifier,
        encryptedDigest             EncryptedDigest,
        unauthenticatedAttributes   [1] IMPLICIT Attributes OPTIONAL }

EncryptedDigest ::= OCTET STRING
```

Keterangan untuk masing-masing *field* yang terdapat dalam **SignerInfo**:

- **version** berisi nomor versi dari **SignerInfo**
- **issuerAndSerialNumber** berisi nama dan nomor serial yang terdapat pada sertifikat penandatangan, berguna sebagai identifikasi sertifikat.
- **digestAlgorithm** berisi pengenal algoritma untuk menghasilkan *digest*, dalam konteks SET, berisi pengenal algoritma SHA-1.
- **authenticatedAttributes** berisi atribut yang ditandatangani oleh penandatangan, ada dua *field*, yaitu tipe dari data yang ditandatangani dan *digest* dari data tersebut.
- **digestEncryptionAlgorithm** berisi pengenal algoritma yang digunakan untuk mengenkrip *digest* dari data. Dalam dalam konteks SET, berisi enkripsi RSA.
- **encryptedDigest** berisi *digest* yang dienkrip menggunakan kunci privat penandatangan. *Field* ini lah yang memuat tanda tangan digital.
- **unauthenticatedAttributes** berisi atribut-atribut yang tidak ditandatangani. Tidak dipakai dalam SET.

Field yang terdapat pada **SignedData** dan **SignerInfo** dirangkum dalam gambar berikut:



Gambar IV-1. SignedData

Keterangan tambahan:

- Untuk *field* **Content**, data yang akan ditandatangani dikodekan ke dalam format DER terlebih dahulu sebelum ditandangani dan disimpan dalam *field* tersebut.
- Maksimal jumlah penandatangan yang diperbolehkan dalam SET adalah dua, jadi *field* **SignerInfos** memiliki maksimal 2 **SignerInfo**.
- Sertifikat tambahan yang terdapat pada **certificates** dipakai untuk menyampaikan sertifikat *Payment Gateway* kepada *Cardholder* oleh *Merchant.*
- Operator kriptografi yang memakai struktur **SignedData** adalah **S(r,t)**.

4.2.2 EnvelopedData

Struktur **EnvelopedData** digunakan untuk menyimpan data yang dikunci menggunakan teknik amplop digital. **EnvelopedData** berisi data yang dienkrip menggunakan kunci simetris dan kunci simetris yang dienkrip menggunakan kunci publik penerima.

Langkah-langkah untuk membuat **EnvelopedData** :

1. Sebuah kunci simetris untuk mengenkrip data dihasilkan secara acak.
2. Untuk masing-masing penerima, kunci simetris tersebut dienkrip menggunakan kunci publik masing-masing penerima.

3. Untuk masing-masing penerima, kunci simetris yang dienkrip tadi beserta informasi mengenai penerima disimpan dalam struktur **RecepientInfo**.
4. Data yang akan diamplopkan dienkrip menggunakan kunci simetris.
5. Data yang dienkrip tadi beserta **RecepientInfo** disimpan dalam struktur **EnvelopedData**.

Penerima data yang diamplopkan tadi membuka amplop digital dengan cara mendekrip menggunakan kunci privat miliknya kunci simetris yang terenkrip dan menggunakan kunci simetris tersebut untuk mendekrip data yang terenkrip.

Spesifikasi **EnvelopedData** dalam kode *ASN.1* :

```
EnvelopedData ::= SEQUENCE {
        version                 Version,
        recipientInfos          RecipientInfos,
        encryptedContentInfo    EncryptedContentInfo }

RecipientInfos ::= SET OF RecipientInfo

EncryptedContentInfo ::= SEQUENCE {
        contentType                     ContentType,
        contentEncryptionAlgorithm      ContentEncryptionAlgorithmIdentifier,
        encryptedContent                [0] IMPLICIT EncryptedContent OPTIONAL }

EncryptedContent ::= OCTET STRING
```

Keterangan untuk masing-masing *field* yang terdapat dalam **EnvelopedData**:

- **version** berisi nomor versi dari **EnvelopedData.**
- **recepientInfos** berisi koleksi informasi dari penerima.
- **encryptedContentInfo** berisi informasi data yang terenkripsi menggunakan kunci simetri.

Struktur **EncryptedContentInfo** berisi *field* sebagai berikut:

- **contentType** berisi tipe dari data yang diamplopkan.
- **contentEncryptionAlgorithm** berisi algoritma yang dipakai untuk mengenkripsi data. Untuk keperluan SET diisi dengan DES-CBC.
- **encryptedContent** berisi hasil enkripsi dari data.

Spesifikasi **RecepientInfo** dalam kode *ASN.1* :

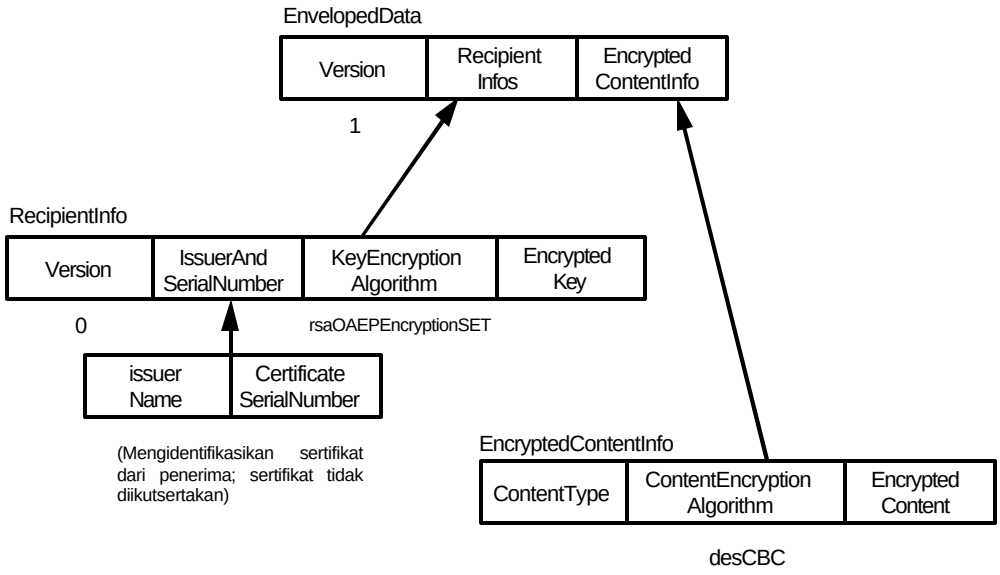
```
RecipientInfo ::= SEQUENCE {
        version                 Version,
```

```
    issuerAndSerialNumber IssuerAndSerialNumber,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey           EncryptedKey }

EncryptedKey ::= OCTET STRING
```

Keterangan untuk masing-masing *field* yang terdapat dalam **RecepientInfo**:

- **version** berisi nomor versi dari **RecepientInfo.**
- **issuerAndSerialNumber** berisi nama dan nomor serial yang terdapat pada sertifikat penerima, berguna sebagai identifikasi sertifikat.
- **keyEncriptionAlgorithm** berisi pengenal algoritma yang dipakai untuk mengenkrip kunci simetris. Dalam hal ini berisi algoritma RSA.
- **encryptedKey** berisi kunci simetris yang telah dienkripsi.



Gambar IV-2. EnvelopedData

Keterangan tambahan :

- Operator kriptografi yang memakai **EnvelopedData** adalah **E(r,t)**.
- Dalam SET hanya diperbolehkan satu **RecepientInfo**, karena dalam setiap komunikasi pada SET hanya melibatkan dua pihak (satu pengirim dan satu penerima), tidak pernah terjadi ada lebih dari satu penerima.
- Dalam spesifikasi SET, *field* **encryptedKey** diisi dengan blok OAEP.

4.2.3 EncryptedData

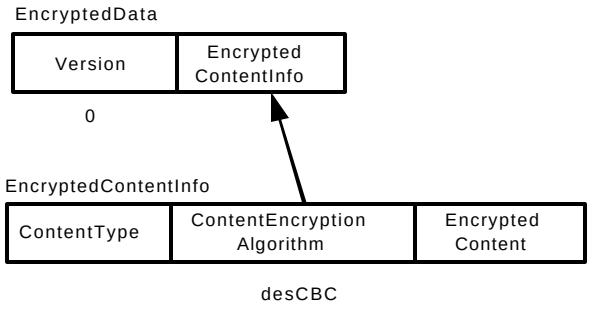
Struktur **EncryptedData** digunakan untuk menyimpan data yang terenkripsi tanpa menyertakan informasi mengenai penerima ataupun kunci yang dipergunakan untuk mengenkripsi.

Spesifikasi **EncryptedData** dalam kode *ASN.1* :

```
EncryptedData ::= SEQUENCE {
        version                 Version,
        encryptedContentInfo    EncryptedContentInfo    }
```

Keterangan untuk masing-masing *field* yang terdapat dalam **EncryptedData**:

- **version** berisi nomor versi dari **EncryptedData.**
- **encryptedContentInfo** berisi data yang ternekrip seperti yang telah dijelaskan pada sub bab 4.2.2.



Gambar IV-3. EncryptedData

Operator kriptografi yang memakai struktur **EncryptedData** adalah **EK(k,t)**.

4.2.4 DigestedData

Struktur **DigestedData** digunakan untuk menyimpan data yang di-*digest* beserta data itu sendiri. Data yang di-*digest* tersebut digunakan dalam pengecekan integritas data.

Langkah-langkah untuk membuat **DigestedData** :

1. *Digest* dari data dihasilkan menggunakan algoritma *message-digest* tertentu.
2. Algoritma tersebut beserta data dan *digest* dari data disimpan dalam struktur **DigestedData.**

Penerima memverifikasi *digest* yang disimpan dengan suatu *digest* baru yang dihasilkan dari data yang diterima.

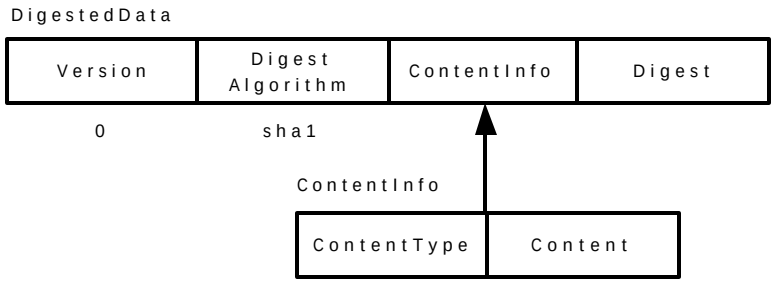
Spesifikasi **DigestedData** dalam kode *ASN.1* :

```
DigestedData ::= SEQUENCE {
        version          Version,
        digestAlgorithm  DigestAlgorithmIdentifier,
        contentInfo      ContentInfo,
        digest           Digest }

Digest ::= OCTET STRING
```

Keterangan untuk masing-masing *field* yang terdapat dalam **DigestedData**:

- **version** berisi nomor versi dari **DigestedData.**
- **digestAlgorithm** berisi pengenal algoritma untuk menghasilkan *digest*, dalam hal ini berisi pengenal algoritma SHA-1.
- **contentInfo** berisi data yang di-*digest.*
- **digest** berisi *digest* dari data.



Gambar IV-4. DigestedData

4.2.5 ContentInfo

Struktur ini dipakai untuk merepresentasikan data dalam bentuk yang general. Struktur **ContentInfo** dalam ASN.1 :

```
ContentInfo ::= SEQUENCE {
        contentType     ContentType,
        content         [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

content berisi data yang direpresentasikan oleh struktur ini, dan tipe dari data tersebut dinyatakan oleh **contentType.**

Object identifier (OID) adalah suatu string yang menyatakan tipe dari suatu objek. Isi dari

OID ditentukan oleh pihak yang membuat objek tersebut. SET mempunyai daftar OID yang didefinisikan di dalam spesifikasinya.

4.3 PEMILIHAN BAHASA PEMROGRAMAN

Komunikasi antara entitas-entitas pada SET adalah berdasarkan pada pembentukan dan pertukaran pesan. Pesan-pesan SET tersusun atas komponen-komponen pembentuk pesan dan setiap komponen-komponen tersebut memakai perangkat-perangkat kriptografi yang didefinisikan dalam spesifikasi SET. Setiap pesan, komponen pembentuk pesan, dan perangkat kriptografi tersebut bisa direpresentasikan dalam bentuk objek. Oleh karena itu dipilihlah bahasa pemrograman yang berorientasi objek.

Dari berbagai macam bahasa pemrograman yang berorientasi objek, peneliti memilih bahasa *Java*. Kenapa *Java* yang dipilih?

- *Java* bersifat *platform independent,* yaitu suatu program yang ditulis dalam *Java* dapat dijalankan pada *platform* apa saja.
- Bahasa pemrograman berorientasi objek yang relatif mudah digunakan. Misalnya untuk kasus manajemen memori, *Java* menggunakan konsep *garbage collection*, yaitu setiap memori yang terpakai tidak perlu dibebaskan secara manual.
- Alat-alat bantu (*tools*) yang ditulis dalam *Java* relatif lengkap untuk implementasi SET. Seperti objek-objek untuk melakukan enkripsi, baik simetris maupun asimetris, penyandian searah, pembuatan kunci privat dan publik, penyimpanan kunci privat dan publik, dan pengkodean menggunakan metode DER

Versi *Java* yang dipakai adalah *Java 2.* Untuk alat bantu kriptografi, peneliti menggunakan *tools* yang dibuat oleh *Australian Business Asociation (ABA)* yang terkumpul dalam *Java Cryptographyc Extension (JCE)* versi 1.1. Untuk implementasi pengkodean menggunakan DER, peneliti menggunakannya sesuai dengan implementasi *Sun.*

4.4 PERANCANGAN OBJEK

Untuk memudahkan implementasi, setiap operator-operator kriptografi, struktur data, dan pesan-pesan SET dibuat dalam objek-objek dan ditulis sebagai *class* dalam bahasa *Java*.

Pengelompokkan objek-objek tersebut adalah:

4.4.1 Operator Kriptografi

Yaitu objek-objek yang membungkus operasi-operasi kriptografi, yang termasuk dalam tipe ini adalah:

- **H(t)**
- **DD(t)**
- **L(t1,t2)**
- **HMAC(t,k)**
- **SO(s,t)**
- **S(s,t)**
- **E(r,t)**
- **EH(r,t)**
- **EX(r,t,p)**
- **EXH(r,t,p)**
- **EK(kd,t)**
- **EncK(kd,s,t)**

4.4.2 Pembungkus Pesan (*message encapsulation*)

Yaitu pembungkus pesan-pesan pembayaran. Setiap pesan-pesan pembayaran dalam SET dibungkus dalam objek ini.

- **MessageWrapper** — Level paling atas dari objek dalam SET, merupakan pembungkus pesan-pesan SET, dan tidak melibatkan fungsi kriptografi apapun.
- **Error** — Objek yang dihasilkan jika terjadi suatu kesalahan dalam pemrosesan pesan-pesan SET. Objek ini menggantikan pesan dan dibungkus di dalam **MessageWrapper**.

4.4.3 Komponen Pembentuk Pesan Pembayaran (*payment message component*),

Yaitu objek-objek yang menjadi komponen pembentuk pesan-pesan pembayaran. Objek-objek tersebut memuat data-data yang diperlukan selama transaksi. Yang termasuk dalam tipe ini adalah:

- **TransIDs**
- **PI**
- **InstallRecurData**
- **AcqCardMsg**
- **PANData**
- **PANToken**

Masing-masing objek tersebut mempunyai objek-objek pembentuk di dalamnya. Tidak

disebutkan di sini, dan untuk melihat daftar yang lengkap dari objek-objek tersebut, dapat dilihat pada bab mengenai implementasi.

4.4.4 Pesan Pembayaran (*payment messages*)

Yaitu pesan-pesan yang memuat data-data transaksi. Yang termasuk dalam tipe ini adalah: pasangan **PInitReq/PInitRes, PReq/PRes**, dan **InqReq/InqRes**.

4.4.5 Pesan-pesan yang berhubungan dengan sertifikat.

Tidak diimplementasikan dalam penelitian ini dan tidak dibahas dalam penulisan ini.

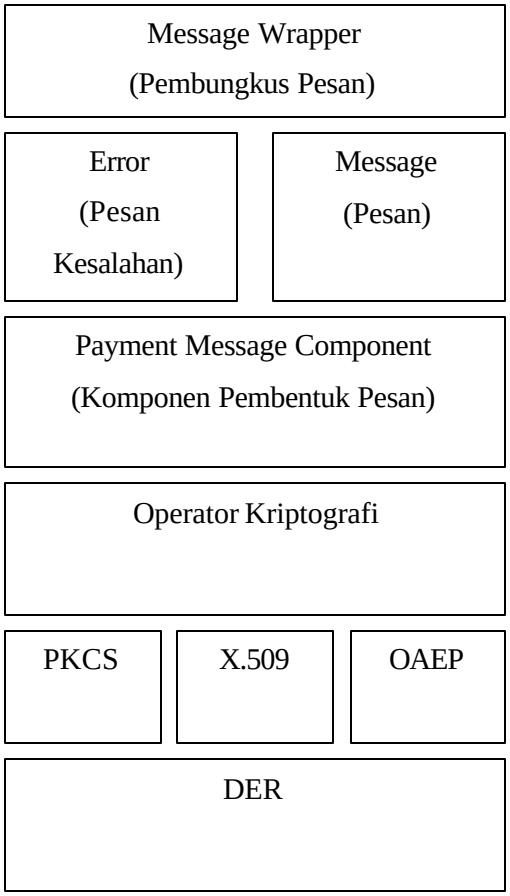
Pengelompokkan objek-objek tersebut di atas hanyalah objek-objek utama, sedangkan objek-objek pembentuk yang terkandung di dalamnya tidak dituliskan. Untuk daftar yang lebih lengkap dari objek-objek yang dihasilkan, dapat dilihat pada bab berikutnya.

Setiap pesan yang dipertukarkan antar entitas, dikodekan dalam bentuk representasi DER. Oleh karena itu setiap objek harus mempunyai *method* untuk mendapatkan representasi DER dari objek tersebut. Setiap objek mempunyai minimal dua *constructor*, yaitu *constructor* untuk membentuk objek tersebut dari objek-objek pembentuknya, dan yang kedua, *constructor* untuk mengembalikan objek tersebut dari representasi DER-nya.

Untuk objek-objek yang mempunyai fungsi-fungsi kriptografi, pada *constructor* kedua (untuk mengembalikan objek dari representasi DER-nya) diberikan parameter tambahan yaitu kunci yang sesuai untuk melakukan fungsi kriptografi, sehingga bisa langsung melakukan proses kriptografi yang diperlukan (enkripsi, dekripsi, penyandian searah, atau pengecekan tanda tangan).

4.5 STRUKTUR LAPISAN OBJEK DALAM SET

Objek-objek yang dipakai dalam SET di susun dalam struktur lapisan objek sebagai berikut:



Gambar IV-6. Struktur lapisan objek

Objek-objek yang berada di lapisan bawah dipakai untuk membentuk objek-objek di lapisan atasnya.

Objek-objek DER dipakai untuk mengkodekan pesan dalam format DER. Untuk objek-objek DER ini, peneliti memakai objek yang dibuat oleh *Sun*.

BAB V
IMPLEMENTASI

Bab ini membahas mengenai bagaimana implementasi *object library* yang dipakai dalam implementasi SET. Pembahasan dimulai dari karakteristik yang dimiliki oleh setiap objek sampai dengan pengelompokkan objek-objek tersebut dalam paket-paket *class*.

5.1 KARAKTERISTIK OBJEK

Berikut ini dibahas mengenai implementasi objek-objek yang dibuat dalam implementasi SET. Karakteristik objek yang dibahas dalam sub-bab ini untuk objek-objek yang tidak memakai fungsi-fungsi kriptografi (misalnya enkripsi simetris) secara langsung. Objek-objek tersebut ada yang memakai operasi kriptografi, tapi melalui objek-objek operator kriptografi. Karakteristik objek-objek kriptografi tersebut akan dibahas dalam sub-bab berikutnya.

5.1.1 *Field* Dalam Objek

Setiap *class* memiliki *field* yang disesuaikan dengan kebutuhan. Kebutuhan tersebut berbeda-beda untuk setiap *class*, tergantung dari spesifikasi objek tersebut dalam format ASN.1.

Misalnya untuk *class* **PIHead**:

Spesifikasi **PIHead** dalam ASN.1 adalah :

```
833 PIHead ::= SEQUENCE {
834     transIDs                TransIDs,
835     inputs                  Inputs,
836     merchantID              MerchantID,
837     installRecurData        [0] InstallRecurData  OPTIONAL,
838     transStain              TransStain,
839     swIdent                 SWIdent,
840     acqBackKeyData          [1] EXPLICIT BackKeyData  OPTIONAL,
841     piExtensions            [2] MsgExtensions {{PIExtensionsIOS}} OPTIONAL
842 }
```

Maka *field* yang dimiliki oleh *class* tersebut adalah:

```
private TransIDs transIDs;
```

```
private Inputs inputs;
private String merchantID;
private InstallRecurData installRecurData;
private HMAC transStain;
private String swIdent;
private BackKeyData acqBackKeyData;
```

Setiap *field* tersebut diberi penanda bertipe *private* agar tidak bisa diakses secara langsung. Untuk mengaksesnya digunakan *method* yang mengembalikan suatu nilai sesuai dengan tipe objek yang dimaksud. Hal ini diterapkan dengan tujuan agar setiap *field* yang terdapat dalam *class* tidak dapat diubah, tapi hanya bisa diambil nilainya.

Penamaan variabel pada setiap *field* tersebut disamakan dengan namanya pada spesifikasi struktur objek dalam ASN.1.

5.1.2 Constructor

Setiap *class* mempunyai dua macam *constructor*, yang pertama untuk membangun objek tersebut dari objek-objek penyusunnya, dan yang kedua untuk membangun objek tersebut dari representasi DER-nya.

Misalkan pada *class* **PIHead**:

- *Constructor* pertama:

```
public PIHead(
    TransIDs transIDs,
    Inputs inputs,
    String merchantID,
    InstallRecurData installRecurData,
    HMAC transStain,
    String swIdent,
    BackKeyData acqBackKeyData) throws Exception
```

Artinya, *class* **PIHead** dibangun dari objek-objek pembentuknya, yaitu:

Objek	Tipe
transIDs	TransIDs
inputs	Inputs
merchantID	String
installRecurData	InstallRecurData
transStain	HMAC
swIdent	String
acqBackKeyData	BackKeyData

Tabel V-1. Objek-objek pembentuk PIHead

Di dalam *constructor* tersebut, *field* yang terdapat dalam *class* **PIHead** diisi dengan nilai yang sesuai:

```
this.transIDs = transIDs;
this.inputs = inputs;
this.merchantID = merchantID;
this.installRecurData = installRecurData;
this.transStain = transStain;
this.swIdent = swIdent;
this.acqBackKeyData = acqBackKeyData;
```

- *Constructor* kedua:

```
public PIHead(DerInputStream din) throws Exception
```

Maksudnya, *class PIHead* dibangun dari representasi DER-nya. Representasi DER dari objek tersebut didapat dari **din** yang bertipe *DerInputStream*.

Cara *decode* objek tersebut dari representasi DER-nya:

```
public PIHead(DerInputStream din) throws Exception {
    DerValue[] adv = din.getSequence(8);

    this.transIDs = new TransIDs(new DerInputStream(adv[0].toByteArray()));
    this.inputs = new Inputs(new DerInputStream(adv[1].toByteArray()));
    this.merchantID = adv[2].getPrintableString();

    //isi installRecurData
    if (adv[3].tag==(byte)0xA0) {
            adv[3].resetTag(DerValue.tag_Sequence);
            installRecurData = new InstallRecurData(new DerInputStream(
            adv[3].toByteArray()));
    }
    else
```

```
                installRecurData = null;

        this.transStain = new HMAC(new DerInputStream(adv[4].toByteArray()));
        this.swIdent = adv[5].getPrintableString();

        //isi acqBackKeyData
        if (adv[6].tag==(byte)0xA1)
                acqBackKeyData = new BackKeyData(adv[6].data);
        else
                acqBackKeyData = null;
}
```

Dapat dilihat dari potongan program di atas, bahwa setiap objek penyusun objek **PIHead** didapatkan dengan membangun masing-masing objek tersebut dari representasi DER-nya.

5.1.3 *Method* Yang Wajib Dimiliki Setiap *Class*

Selain ketentuan mengenai *constructor* tersebut, setiap *class* mempunyai *method* untuk mendapatkan representasi DER dari objek tersebut.

Method untuk mendapatkan representasi DER:

```
public void derEncode(OutputStream out) throws IOException
```

Method tersebut melakukan pengkodean dengan ketentuan DER, dan menyimpan hasilnya di dalam variabel **out** yang bertipe *OutputStream.*

Cara mengkodekan suatu objek ke dalam representasi DER adalah sebagai berikut:

- Siapkan dulu sebuah objek bertipe **DerOutputStream**, misalkan diberi nama **dout**. Objek ini sudah dibuat oleh *Sun*, dan peneliti tinggal memakainya untuk menghasilkan representasi objek dalam DER.
- Kemudian objek-objek penyusun yang dimiliki oleh objek yang akan dikodekan satu persatu dan representasi DER dari masing-masing objek dimasukkan ke dalam **dout**. Untuk objek yang mempunyai *method derEncode* (*method* yang digunakan untuk mengkodekan), *method* tersebut dipakai dengan **dout** sebagai parameternya. Untuk objek yang bertipe sederhana (lihat keterangan mengenai ASN.1 pada bab 2), objek **DerOutputStream** mempunyai *method* untuk mengkodekannya.
- Setelah semua representasi DER dari objek-objek pembentuk tersebut masuk ke dalam **dout**, maka tag dari objek utama ditulis ke dalam **dout** (keterangan mengenai tag dapat dilihat pada pembahasan mengenai ASN.1 pada bab 2).

Contoh untuk *class* **PIHead,** cara mengkodekannya :

```
public void derEncode(OutputStream out) throws IOException {
        DerOutputStream dout = new DerOutputStream();
        transIDs.derEncode(dout);
        inputs.derEencode(dout);
        dout.putPrintableString(merchantID);

        //kalau installRecurData ada, tulis, kalau tidak, tulis null
        if (installRecurData != null) {
                DerOutputStream midout = new DerOutputStream();
                installRecurData.derEncode(midout);
                dout.writeImplicit((byte)0xA0,midout);
        }
        else
                dout.putNull();

        transStain.encode(dout);
        dout.putPrintableString(swIdent);

        //kalau acqBackKeyData ada, tulis, kalau tidak, tulis null
        if (acqBackKeyData != null) {
                DerOutputStream midout1 = new DerOutputStream();
                acqBackKeyData.encode(midout1);
                dout.write((byte)0xA1,midout1);
        }
        else
                dout.putNull();

        //tulis PIExtension
        dout.putNull();

        //tulis tag sequence
        DerOutputStream dout2 = new DerOutputStream();
        dout2.write(DerValue.tag_Sequence,dout);
        out.write(dout2.toByteArray());
}
```

Selain *method* untuk mendapatkan representasi DER dari objek yang bersangkutan, setiap *class* wajib memiliki *method* untuk mendapatkan *field* yang dimilikinya.

Contoh untuk *class* **PIHead** :

```
public TransIDs getTransIDs() throws Exception {
        return transIDs;
}
public Inputs getInputs() throws Exception {
        return inputs;
}
public String getMerchantID() throws Exception {
        return merchantID;
}
public InstallRecurData getInstallRecurData() throws Exception {
        return inputs;
}
```

```
public HMAC getTransStain() throws Exception {
        return transStain;
}
public String getSwIdent() throws Exception {
        return swIdent;
}
public BackKeyData getAcqBackKeyData() throws Exception {
        return inputs;
}
```

5.2 KARAKTERISTIK OBJEK OPERATOR KRIPTOGRAFI

Objek-objek yang termasuk dalam kelompok ini mempunyai karakteristik khusus, yaitu mempunyai proses-proses kriptografi, seperti enkripsi, dekripsi, dan fungsi *hash* satu arah. Oleh karena itu objek-objek tersebut mempunyai parameter tambahan pada *constructor*-nya, yaitu kunci yang dipakai untuk melakukan proses kriptografi. Kunci tersebut bisa berupa kunci privat, kunci publik yang disimpan dalam sertifikat, ataupun kunci simetris.

Objek-objek operator kriptografi melakukan operasi kriptografi terhadap data yang diberikan padanya. Data-data tersebut diberikan dalam bentuk **ContentInfo**, karena objek-objek yang merepresentasikan data tersebut mempunyai *object identifier* yang didefinisikan dalam SET. Untuk daftar secara lengkap dapat dilihat pada dokumen spesifikasi SET.

Misalnya untuk operator **EK**:

Objek ini melakukan enkripsi kunci simetris, oleh karena itu parameter tambahan yang diperlukan adalah kunci simetris yang dipakai. Objek ini menerima dua parameter dalam *constructor*-nya, yaitu kunci simetris dan **ContentInfo**.

```
public EK(DESKey k, ContentInfo ci) throws Exception
```

Kemudian **ci** dienkrip menggunakan fungsi enkripsi DES dengan kunci **k** sebagai kunci simetrisnya. Hasil akhir dari operator **EK** adalah **EncryptedData**, sesuai dengan spesifikasi **EK** dalam ASN.1 :

```
2937 EK { KeyData, ToBeEnveloped } ::= EncryptedData
2938    (CONSTRAINED BY { ToBeEnveloped, -- encrypted with -- KeyData})
2939    (WITH COMPONENTS { ..., encryptedContentInfo
2940            (WITH COMPONENTS { ..., encryptedContent PRESENT}) })
```

Untuk lengkapnya, lihat potongan *code* berikut:

```
public EK(DESKey k, ContentInfo ci) throws Exception {
        key = k;
```

```
        toBeEnveloped = ci;
        encContentInfo = makeEncryptedContentInfo(key,toBeEnveloped);
        encryptedData = new EncryptedData(encContentInfo);
    }
```

Fungsi enkripsi DES terdapat pada fungsi makeEncryptedContentInfo dan tidak diterangkan lebih lanjut.

Untuk mendapatkan kembali data yang dienkrip tadi, **EK** di konstruksi kembali dari representasi DER-nya. Parameter yang diterima oleh *constructor* adalah kunci untuk melakukan dekripsi (kunci yang sama dengan yang dipakai untuk mengenkrip) dan variabel yang menyimpan representasi DER dari **EK**.

```
public EK(DESKey k, DerInputStream din) throws Exception
```

Kemudian **EncryptedData** dikonstruksi dari representasi DER-nya yang tersimpan dalam **din**, lalu **EncryptedContent** yang tersimpan dalam **EncryptedData** didekrip secara simetris menggunakan kunci **k**. Hasilnya disimpan dalam bentuk **ContentInfo**.

Untuk lengkapnya, lihat potongan *code* berikut:

```
public EK(DESKey k, DerInputStream din) throws Exception {
        key = k;
        encryptedData = new EncryptedData(din);

        encContentInfo = encryptedData.getEncryptedContentInfo();

        //ambil parameter yang berisi IVData
        parameter = encContentInfo.getContentEncryptionAlgorithm();

        //ambil IVData
        byte[] param = parameter.getEncodedParams();
        DerValue derIv = new DerValue(param);
        IVData = derIv.getOctetString();

        //ambil encryptedContent untuk mengambil toBeEnveloped
        byte[] encryptedContent = encContentInfo.getEncryptedContent();
        byte[] plain = decryptDES(encryptedContent, key);

        //buat toBeEnveloped
        toBeEnveloped = new ContentInfo(encContentInfo.getContentType(),
                                        new DerValue(plain));
}
```

Method untuk mendapatkan representasi DER dari **EK** mirip dengan yang terdapat pada objek-objek lainnya yang bukan operator kriptografi.

Objek-objek operator kriptografi lainnya mempunyai karakteristik yang mirip dengan operator **EK** tersebut sehingga tidak dibahas lebih lanjut.

5.3 HIRARKI OBJEK

Untuk memudahkan pengorganisasian objek-objek yang dibuat, maka objek-objek tersebut dikelompokkan dalam paket-paket objek yang disajikan dalam betuk tabel-tabel di bawah ini. Dalam setiap tabel tersebut tertulis nama *class* dan nama pembuatnya. Maksudnya adalah peneliti hanya membahas dan menguji objek-objek yang dibuat oleh peneliti (Yudi), dan diasumsikan bahwa objek-objek yang tidak dibuat oleh peneliti, telah diuji oleh pembuat masing-masing.

5.3.1 Paket digsec.set.core.crypto

Berisi objek-objek yang melakukan fungsi kriptografi.

Nama *Class*	Pembuat
DetachedDigest	Arif
E	Arif
EH	Arif
EK	Yudi
EX	Arif
EXH	Arif
HMAC	Yudi
Linkage	Arif
OAEP	Arif
OAEPEncoder	Arif
S	Yudi
SO	Yudi

Tabel V-2. Paket digsec.set.core.crypto

5.3.2 Paket digsec.set.core.pkcs

Berisi objek-objek yang didefiniskan oleh standar PKCS.

Nama Class	Pembuat
AlgorithmId	Sun
ContentInfo	Sun
DigestedData	Arif
EncryptedContentInfo	Arif
EncryptedData	Yudi
EnvelopedData	Arif
PKCS7	Sun
PKCS9Attribute	Sun
PKCS9Attributes	Sun
RecipientInfo	Arif
SignerInfo	Sun
ParsingException	Sun

Tabel V-3. Paket digsec.set.core.pkcs

5.3.3 Paket digsec.set.core.encapsulation

Berisi objek-objek yang melakukan enkapsulasi data.

Nama Class	Pembuat
Enc	Arif
EncB	Arif
EncBX	Arif
EncK	Yudi
EncX	Arif

Tabel V-4. Paket digsec.set.core.encapsulation

5.3.4 Paket digsec.set.payment.component

Berisi objek-objek pembentuk pesan pembayaran

Nama Class	Pembuat
AuthToken	Arif
AuthTokenData	Arif
BackKeyData	Yudi
CountryCode	Yudi
CurrencyAmount	Yudi
HOD	Yudi
HODInput	Yudi
HOIData	Yudi
HPIData	Yudi
Inputs	Yudi
InstallRecurData	Arif
InstallRecurInd	Arif
Location	Arif
MerTermIDs	Arif
OIData	Yudi
PANData	Arif
PANToken	Arif
PI	Yudi
PIData	Yudi
PIDataUnsigned	Yudi
PIDualSigned	Yudi
PIHead	Yudi
PIOILink	Yudi
PISignature	Yudi
PITBS	Yudi
PIUnsigned	Yudi
Recurring	Arif
RRTags	Arif
TransIDs	Arif

Tabel V-5. Paket digsec.set.payment.component

5.3.5 Paket digsec.set.payment.message

Berisi pesan-pesan pembayaran SET dan komponen pesan yang spesifik hanya untuk pesan-pesan tersebut.Di dalam kelompok ini terdapat 3 paket objek, yang sesuai dengan namanya masing-masing, menunjukkan objek pasangan pesan.

Nama Paket	Nama *Class*	Pembuat
digsec.set.payment.message.pinitreqres	PInitReq	Yudi
	PinitResData	
	PInitRes	
digsec.set.payment.message.preqres	OIData	
	OIDualSigned	
	OIUnsigned	
	PReqDualSigned	
	PReqUnsigned	
	PReq	
	PResPayLoad	
	CompletionCode	
	Result	
	AuthStatus	
	CapStatus	
	CreditStatus	
	PResPayload	
	PResData	
	PRes	
digsec.set.payment.message.inqreqres	InqReqData	
	InqReqSigned	
	InqReq	
	InqRes	

Tabel V-6. Paket digsec.set.payment.message

5.3.6 Paket digsec.set.payment.message.wrapper

Berisi objek-objek pembungkus pesan SET

Nama Paket	Nama Class	Pembuat
digsec.set.payment.message.wrapper	Message	Yudi
	MessageHeader	
	MessageIDs	
	MessageWrapper	

Tabel V-7. Paket digsec.set.payment.message.wrapper

5.3.7 Paket digsec.set.payment.message.error

Berisi objek pembungkus pesan kesalahan.

Nama Paket	Nama Class	Pembuat
digsec.set.payment.message.error	Error	Yudi
	ErrorCode	
	ErrorMsg	
	ErrorTBS	

Tabel V-8. Paket digsec.set.payment.message.error

BAB VI
UJI COBA

6.1 SKENARIO PENGUJIAN

Setiap objek yang dibuat diuji dengan cara membuat objek tersebut dari objek-objek pembentuknya. Kemudian objek tersebut dikodekan ke dalam representasi DER. Setelah itu objek tersebut dikembalikan ke objek semula dari representasi DER-nya.

Untuk objek-objek operator kriptografi, pengujian ditambahkan dengan memeriksa keberhasilan fungsi-fungsi kriptografinya. Pengujian pertama dengan memberikan kunci yang sesuai dan pengujian kedua dengan kunci yang salah

Setiap objek dibuatkan *method* untuk memeriksa isi dari objek tersebut. Pemeriksaan isi objek tersebut yaitu dengan mencetak ke layar, isi dari objek tersebut sampai ke objek pembentuk tingkat paling bawah. Untuk mengetahui apakah objek tersebut berhasil dibentuk dari objek-objek pembentuknya, maka *method* untuk melihat isi dari objek tersebut dijalankan dan diperiksa isinya. Apakah semua objek-objek pembentuknya ada di dalam objek tersebut. *Method* tersebut diberi nama *toString*.

Objek-objek yang diuji hanya yang dibuat oleh peneliti. Objek-objek yang dibuat oleh pihak lain diasumsikan sudah diuji oleh pembuatnya masing-masing.

6.2 HASIL YANG DIHARAPKAN

Hasil yang diharapkan dari pengujian adalah setiap objek berhasil dibuat dari objek-objek pembentuknya, berhasil dikodekan ke dalam representasi DER, dan berhasil dikembalikan kembali ke objek semula dari representasi DER-nya.

Untuk objek operator kriptografi, hasil yang diharapkan ketika diberikan kunci yang sesuai, adalah objek berhasil dibentuk. Pada pengujian dengan kunci yang salah, diharapkan objek memberikan suatu *exception*.

6.3 PENGUJIAN

Berikutnya dibahas mengenai aspek apa saja yang diuji dalam setiap objek yang dihasilkan, serta hasil yang diharapkan.

6.3.1 Pengujian Objek Operator Kriptografi

Aspek-aspek yang diuji untuk objek-objek operator kriptografi adalah:

- Pembentukan objek, dengan kondisi:
  - Kunci yang diberikan sesuai dengan operatornya, misalnya kunci simetris untuk operator EK dan kunci asimetris untuk operator S dan SO. Hasil yang diharapkan adalah objek berhasil dibentuk dan operasi kriptografinya berhasil.
  - Data yang diberikan kepada objek operator kriptografi dalam bentuk ContentInfo
  - Dari kedua poin di atas, diberikan parameter yang salah, diharapkan objek mengembalikan *exception.*
- *Method toString* dijalankan untuk melihat isi dari objek yang dibentuk, hasil yang diharapkan adalah isi objek tersebut terlihat pada layar.
- Objek yang dibentuk dikodekan ke dalam format DER
- Rekonstruksi objek dari representasi DER-nya, dengan kondisi:
  - Kunci yang diberikan sesuai dengan operatornya, misalnya kunci simetris untuk operator EK dan kunci asimetris yang merupakan pasangannya untuk operator S dan SO
  - Representasi DER yang diberikan merupakan hasil dari pengkodean objek yang telah dibentuk sebelumnya
  - Dari kedua poin di atas, diberikan parameter yang salah, diharapkan objek mengembalikan *exception.*

6.3.2 Pengujian Objek Non Kriptografi

Aspek-aspek yang diuji untuk objek-objek operator kriptografi adalah:

- Pembentukan objek, dengan kondisi:
  - Objek dibentuk dengan cara membentuknya dari objek-objek lapisan bawahnya
  - Dicoba juga dengan memberikan objek yang tidak sesuai dengan paramter yang seharusnya. Hasil yang diharapkan adalah objek mengembalikan *exception*.

- *Method toString* dijalankan untuk melihat isi dari objek yang dibentuk, hasil yang diharapkan adalah isi objek tersebut terlihat pada layar.
- Objek yang dibentuk dikodekan ke dalam format DER.
- Rekonstruksi objek dari representasi DER-nya, dengan kondisi:
  - Representasi DER yang diberikan merupakan hasil dari pengkodean objek yang telah dibentuk sebelumnya. Hasil yang diharapkan adalah objek tersebut berhasil dibentuk kembali.
  - Diberikan representasi DER yang salah, diharapkan objek mengembalikan *exception.*

6.4 HASIL PENGUJIAN

6.4.1 Objek Operator Kriptografi

Hasil pengujian objek-objek operator kriptografi dikumpulkan dalam tabel berikut:

Objek	Kondisi									
	Konstruksi				Inspeksi Objek	Pengkodean	Rekonstruksi			
	Kunci		ContentInfo				Kunci		DER	
	V	I	V	I			V	I	V	I
EK										
EncK										
H										
HMAC										
S										
SO										

Tabel VI-1. Hasil pengujian objek operator kriptografi

Keterangan :

- Kondisi-kondisi diberikan seperti yang disebutkan pada sub-bab 6.3.1
- Simbol V menunjukkan bahwa parameter yang diberikan valid, dan simbol I menunjukkan parameter tersebut invalid.

- Tanda menunjukkan hasil pengujian sesuai dengan yang diharapkan.

6.4.2 Objek Non Kriptografi

Objek	Kondisi					
	Konstruksi		Inspeksi Objek	Pengkodean	Rekonstruksi	
	Objek Pembentuk				DER	
	V	I			V	I
BackKeyData						
CountryCode						
CurrencyAmount						
HOD						
HODInput						
HOIData						
HPIData						
Inputs						
OIData						
PI						
PIData						
PIDataUnsigned						
PIDualSigned						
PIHead						
PIOILink						
PISignature						
PITBS						
PIUnsigned						

Tabel VI-2. Hasil pengujian objek komponen pesan

Objek	Kondisi					
	Konstruksi		Inspeksi Objek	Pengkodean	Rekonstruksi	
	Objek Pembentuk				DER	
	V	I			V	I
PInitReq						
PinitResData						
PInitRes						
OIData						
OIDualSigned						
OIUnsigned						
PReqDualSigned						
PReqUnsigned						
PReq						
PResPayLoad						
CompletionCode						
Result						
AuthStatus						
CapStatus						
CreditStatus						
PResPayload						
PResData						
PRes						
InqReqData						
InqReqSigned						
InqReq						
InqRes						

Tabel VI-3. Hasil pengujian objek pesan

Objek	Kondisi					
	Konstruksi Objek Pembentuk		Inspeksi Objek	Pengkodean	Rekonstruksi DER	
	V	I			V	I
Message						
MessageHeader						
MessageIDs						
MessageWrapper						

Tabel VI-4. Hasil pengujian objek pembungkus pesan

Objek	Kondisi					
	Konstruksi Objek Pembentuk		Inspeksi Objek	Pengkodean	Rekonstruksi DER	
	V	I			V	I
Error						
ErrorCode						
ErrorMsg						
ErrorTBS						

Tabel VI-5. Hasil pengujian objek pesan kesalahan

Keterangan :

- Kondisi-kondisi diberikan seperti yang disebutkan pada sub-bab 6.3.2
- Simbol V menunjukkan bahwa parameter yang diberikan valid, dan simbol I menunjukkan parameter tersebut invalid.
- Tanda menunjukkan hasil pengujian sesuai dengan yang diharapkan.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan yang diambil penulis dari hasil analisa dan implementasi protokol SET. Kemudian dilanjutkan dengan saran-saran untuk pengembangan selanjutnya.

7.1 KESIMPULAN

Beberapa kesimpulan yang dapat diambil oleh penulis setelah menganalisa protokol komunikasi antara *Cardholder* dan *Merchant* dan mengimplementasikan objek-objek pendukungnya:

1) Protokol SET dapat dikatakan aman karena memakai teknik-teknik kriptografi dan memanfaatkan sertifikat digital.

2) Protokol komunikasi antara *Cardholder* dan *Merchant* memakai 3 pasang pesan,yaitu: PInitReq/PInitRes, PReq/PRes, dan InqReq/InqRes. Dari ketiga pasang pesan tersebut, pasangan PReq/PRes adalah yang paling vital, karena pada pasangan pesan inilah informasi kartu pembayaran dikirimkan. Oleh karena itu, pada pasangan pesan inilah keamanan pesan paling diperhatikan.

3) Kalau dilihat dari penulisan format pesan dalam SET yang ditulis memakai aturan ASN.1, maka sangat cocok kalau pesan-pesan tersebut diimplementasikan dalam bentuk objek.

4) Implementasi objek-objek dalam SET bisa dilakukan menggunakan bahasa *Java.* Kalau diimplementasikan memakai *Java*, keuntungannya antara lain dapat dijalankan di semua *platform* dan alat-alat bantu kriptografinya sudah lengkap.

5) Objek-objek pesan yang dibuat belum mengimplementasikan ekstensi dari masing-masing objek tersebut, karena isi ekstensi tergantung dari kebijakan *Brand*, dan peneliti tidak dapat menemukan informasi yang cukup mengenai kebijakan-kebijakan tersebut serta implementasinya dalam SET .

6) Objek-objek yang dihasilkan telah diuji fungsionalitasnya namun belum dapat langsung dipakai untuk implementasi protokol pembayaran antara *Cardholder* dan *Merchant* secara utuh.

7.2 SARAN

Berikut ini beberapa saran yang dapat diberikan oleh penulis:

1) Protokol SET sebaiknya dipakai di Indonesia karena sampai penelitian ini dilakukan, belum ada perusahaan di Indonesia yang memakai SET.
2) Objek-objek yang dihasilkan bisa dipakai untuk program aplikasi *wallet* yang ditulis dalam bahsa *Java* dan dipakai oleh *Cardholder*.
3) Untuk membuat *Merchant Server* (program yang dijalankan oleh *Merchant*), sebaiknya tidak memakai bahasa *Java*, karena masalah kecepatan eksekusinya.

DAFTAR ACUAN

[Hof 99] Robert D. Hof : What Every CEO Needs to Know about Electronic Business; *Business Week* 22/3 (1999).

[HoMS 98] Robert D. Hof, Gary McWilliams, Gabrielle Saveri: The "Click Here" Economy; *Business Week* 22/6 (1998).

[Kali 93] Burton S. Kaliski Jr.: *A Layman's Guide to a Subset of ASN.1, BER, and DER.;* RSA Laboratories, 1993.

[Larm99] Larmout, Prof. John: *Complete ASN.1.;* Open System Solution., 1999.

[McSi 97] Louise McEvoy, Kevin Simzer: *Entrust/CommerceCA (Entrust Technology White Paper)*; Entrust Technology, 1997.

[PKCS #7] RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard.* Version 1.5, November 1993.

[Schn 96] Bruce Schneier: *Applied Cryptography, 2nd ed.;* John Wiley & Sons, Inc., New York 1996.

[ToYo 98] Chris Le Tocq, Steve Young: *SET Comparative Performance Analysis (A White Paper from GartnerGroup)*; Gartner Consulting, California 1998.

[ViMa97a] Visa dan Mastercard, *Secure Electronic Transaction Specification Book 1: Business Description;* Visa dan Mastercard, 1997.

[ViMa97b] Visa dan Mastercard, *Secure Electronic Transaction Specification Book II: Programmer's Guide;* Visa dan Mastercard, 1997.

[ViMa97c] Visa dan Mastercard, *Secure Electronic Transaction Specification Book II1: Formal Protocol Definition;* Visa dan Mastercard, 1997.

[Visa 99] Visa: Homepage, 1999. http://www.visa.com.