BAB II

LANDASAN TEORI

2.1 Ijazah Universitas Multimedia Nusantara

Ijazah Universitas Multimedia Nusantara adalah ijazah yang dikeluarkan oleh kampus Universitas Multimedia Nusantara. Ijazah ini diberikan kepada mahasiswa-mahasiswa sebagai bukti telah selesai melakukan studi di Universitas Multimedia Nusantara. Berdasarkan wawancara yang dilakukan (Nunik dan Wira, 2019), terdapat beberapa ketentuan yang harus diselesaikan oleh seorang mahasiswa sebelum menerima ijazah, antara lain:

a. Menyelesaikan semua mata kuliah yang diwajibkan program studi masing-masing.
b. Menyelesaikan minimal 144 SKS (Satuan Kredit Semester).
c. Menyelesaikan magang.
d. Menyelesaikan skripsi.
e. Menyelesaikan kewajiban administrasi seperti kemahasiswaan, keuangan dan perpustakaan.
f. Menyelesaikan poin SKKM (Sistem Kredit Kegiatan Mahasiswa).

Mahasiswa yang telah menyelesaikan ketentuan-ketentuan di atas sudah dapat dibuatkan ijazahnya. Berikut ini adalah tahapan-tahapan pembuatan ijazah:

a. PIN (Penomoran Ijazah Nasional) akan di-*input* oleh pihak BIA kepada mahasiswa yang telah selesai di situs web my.umn.ac.id.
b. Ijazah yang sudah memiliki PIN ditarik datanya.
c. Ijazah dicetak berdasarkan data yang ditarik sebelumnya.

d. Ijazah diserahkan ke dekan dan rektor untuk ditandatangani.

2.2 Confidentiality, Integrity, dan Availability

Keamanan sebuah informasi dibagi menjadi beberapa properti, yaitu *Confidentiality* (kerahasiaan), *Integrity* (integritas), dan *Availability* (ketersediaan). Ketiganya merupakan cara yang terkenal untuk mendefinisikan arti dan kriteria yang harus dipenuhi agar sebuah sistem aman (Chaeikar dkk., 2012). Masing-masing properti memiliki tujuan yang berbeda-beda dalam menangani keamanan informasi.

2.2.1 Confidentiality

Confidentiality adalah sebuah properti dari informasi dengan tujuan agar informasi tersebut tidak tersedia atau diungkapkan kepada pihak, entitas, atau yang tidak berhak (ISO/IEC 27000, 2018). Properti ini memungkinkan informasi hanya dapat diakses oleh segelintir orang. Semakin sedikit pihak yang dapat mengakses informasi, maka tingkat *confidentiality*-nya semakin tinggi, sedangkan semakin banyak pihak yang dapat mengakses informasi, maka tingkat *confidentiality*-nya semakin rendah. Meskipun dengan tingkat *confidentialty* yang rendah, bocornya informasi kepada pihak yang tidak berhak tetap menghilangkan *confidentiality* suatu informasi. C*onfidentiality* suatu aset juga mempengaruhi *availability*-nya. Semakin konfidensial suatu aset (semakin bersifat rahasia), maka cara mengaksesnya pun dipersulit atau sangat dibatasi. Sebaliknya, semakin rendah tingkat konfidensial suatu aset, maka cara mengaksesnya lebih mudah.

2.2.2 Integrity

Integrity adalah sebuah properti dari informasi untuk memelihara akurasi dan kelengkapan dari suatu aset. Suatu aset dinilai tidak memiliki integritas jika data pada aset tidak sesuai dengan data asli atau data tidak lengkap (ISO/IEC 27000, 2018). Berubahnya data dalam aset menghilangkan sifat integritas aset tersebut. Hal ini berkaitan dengan keaslian suatu data. Aset yang pernah diubah atau sudah diubah datanya sudah kehilangan properti integritasnya.

2.2.3 Availability

Availability adalah sebuah properti dari informasi agar dapat diakses dan digunakan atas permintaan kepada yang berwenang (ISO/IEC 27000, 2018). Jika informasi dapat diakses oleh publik maka *availability* dinilai tinggi, sedangkan jika informasi hanya dapat diakses oleh sebagian individu/organisasi/kelompok, maka *availability* dinilai rendah. *Availability*, seperti yang dijelaskan pada Sub-subbab 2.2.1, memiliki hubungan dengan *confidentiality* dalam hal tingkat kesulitan pengaksesan informasi.

2.3 Base Metrics

Menurut standar yang dicantumkan (Scarfone & Mell, 2009), keamanan diuji menggunakan *Base Metrics*. *Base Metics* ini dibagi menjadi *Base Exploitability* dan *Base Impacts*. *Base Exploitability* memiliki properti *Access Vector* (AV), *Authentication* (Au), dan *Access Complexity* (AC). Berikut ini adalah rumus yang digunakan untuk menghitung *Base Impacts*.

$$I = 10{,}41 * \big(1 - (1 - CI) * (1 - II) * (1 - AI)\big) \quad ...(2.1)$$

Base Impacts dibagi menjadi *Confidentiality Impact*, *Integrity Impact*, dan *Availability Impact*. *Confidentiality Impact* (CI) mengukur dampak potensi kerahasiaan jika eksploitasi berhasil dilakukan. Skor pada CI dibagi menjadi *None*, *Partial*, dan *Complete*. Skor *None* mengartikan tidak ada dampak pada kerahasiaan sistem. Skor *Partial* mengartikan terdapat sebagian informasi yang keluar atau mendapatkan akses yang tidak seharusnya. Skor *Complete* mengartikan informasi menjadi terbuka dan terlihat oleh individu atau kelompok yang tidak memiliki hak (Scarfone & Mell, 2009).

Integrity Impact (II) mengukur dampak potensi integritas dari sistem jika eksploitasi berhasil dilakukan. Skor pada II dibagi menjadi *None*, *Partial*, dan *Complete*. Skor *None* mengartikan tidak ada dampak integritas pada sistem. Skor *Partial* mengartikan modifikasi dari konfigurasi atau informasi bisa terjadi. Skor *Complete* mengartikan integritas sistem membahayakan dan penyerang dapat melakukan perubahan penuh terhadap sistem (Scarfone & Mell, 2009).

Availability Impact (AI) mengukur dampak potensi ketersediaan dari sistem jika eksploitasi berhasil dilakukan. Skor pada AI dibagi menjadi *None*, *Partial* dan *Complete*. Skor *None* berarti tidak ada dampak pada ketersediaan sistem. Skor *Partial* mengartikan terdapat pengurangan performa pada sistem, namun sistem tetap dapat berjalan. Skor *Complete* mengartikan sistem tidak tersedia sama sekali (Scarfone & Mell, 2009).

$$E = 20 * AV * Au * AC \quad ...(2.2)$$

Metrik *Access Vector* (AV) mengukur tingkat akses yang diperlukan untuk dapat melakukan eksploitasi konfigurasi. Tingkatan tersebut merupakan *Local*,

Adjacent Network, dan *Network*. *Local* mengartikan eksploitasi dapat dilakukan dengan mendapatkan akses sistem secara fisik. *Adjacent Network* mengartikan eksploitasi dapat dilakukan dengan mendapat akses pada jaringan lokal melalui *bluetooth*, jaringan nirkabel, atau *ethernet* lokal. *Network* mengartikan eksploitasi dapat dilakukan dengan akses yang dilakukan di luar jaringan lokal.

Metrik *Authentication* (Au) mengukur berapa banyak autentikasi yang diperlukan untuk dapat melakukan eksploitasi. Pengukuran metrik ini dibagi menjadi *Multiple*, *Single*, dan *None*. *Multiple* mengartikan celah keamanan membutuhkan autentikasi dua atau lebih. *Single* mengartikan celah keamanan memerlukan sebuah autentikasi. *None* mengartikan celah keamanan tidak membutuhkan autentikasi sama sekali (Scarfone & Mell, 2009).

Metrik *Access Complexity* (AC) mengukur tingkat kesusahan akses yang diperlukan untuk melakukan eksploitasi. Metrik ini dibagi menjadi *High*, *Medium*, dan *Low*. Tingkat *High* mengartikan akses yang didapat untuk melakukan eksploitasi cukup sulit, seperti harus mendapatkan hak akses istimewa. Tingkat *Medium* mengartikan akses didapatkan dengan tidak terlalu mudah, namun tetap membutuhkan pencarian informasi lebih. Tingkat *Low* mengartikan akses didapatkan dengan mudah (Scarfone & Mell, 2009).

Perhitungan pada *Base Metrics* dilakukan dengan menggunakan *Base Score*. *Base Score* merupakan rumus yang digunakan untuk menghitung skor konfigurasi sistem secara angka. Rumus ini menggunakan skor pada *Base Exploitability* dan *Base Impacts* dan mengonversikannya ke dalam angka. Tabel 2.1 berikut ini adalah konversi skor-skor kedua metrik di atas (Scarfone & Mell, 2009).

Tabel 2.1 Nilai skor angka *Base Metrics*

Base Metrics		***Score***	***Value***
Base Exploitability	AV	*Local*	0,395
		Adjacent Network	0,646
		Network	1
	Au	*Multiple*	0,45
		Single	0,56
		None	0,704
	AC	*High*	0,35
		Medium	0,61
		Low	0,71
Base Impacts	CI	*None*	0
		Partial	0,275
		Complete	0,66
	II	*None*	0
		Partial	0,275
		Complete	0,66
	AI	*None*	0
		Partial	0,275
		Complete	0,66

(Scarfone & Mell, 2009)

Perhitungan dilakukan dengan menghitung *Exploitability* (E), *Impact* (I), f(*Impact*), dan terakhir adalah *BaseScore*. Berikut ini adalah masing-masing rumus tersebut.

$$BaseScore = round\_to\_1\_dec\big(((0{,}6 * I) + (0{,}4 * E) - 1{,}5) * f(Impact)\big) \quad ...(2.3)$$

Nilai pada f(*Impact*) tergantung pada perhitungan nilai *Impact* (I). Fungsi f(*Impact*) akan bernilai 0 (nol) jika nilai *Impact* (I) sama dengan 0 (nol). Jika nilai *Impact* (I) tidak sama dengan 0 (nol), maka nilai f(*Impact*) menjadi 1,176 (Scarfone & Mell, 2009).

2.4 Framework Lisk

Lisk merupakan platform aplikasi *blockchain* yang dapat digunakan untuk mengembangkan aplikasi *blockchain*. Pengembangan aplikasi *blockchain* dapat dilakukan menggunakan SDK yang sudah tersedia. SDK pada *Framework* Lisk ditulis menggunakan bahasa pemrograman JavaScript. Lisk memiliki *token* tersendiri yang digunakan dalam ekosistemnya yang bernama Lisk (LSK) (Lisk *Documentation*, 2019).

2.4.1 Algoritma Konsensus

Sebagai *framework* untuk aplikasi *blockchain*, Lisk memiliki algoritma konsensus. Algoritma konsensus yang digunakan adalah *Delegated Proof of Stake* (DPoS). DPoS adalah sebuah algoritma konsensus yang digunakan untuk mempertahankan kesepakatan yang benar di seluruh jaringan, memvalidasi transaksi, dan bertindak sebagai bentuk demokrasi digital. Sebagai bentuk demokrasi, terdapat pemilihan dalam DPoS. Pemilihan dilakukan untuk menentukan *delegate*. *Delegates* (perwakilan) adalah sebuah tipe akun yang telah terdaftar menggunakan *delegate registration transaction* (Lisk *Documentation*, 2019). Pemilihan ini yang membedakan DPoS dengan PoS (*Proof of Stake*) biasa. Konsensus pada PoS, semakin banyak koin yang dimiliki *stakeholder*, *stakeholder* tersebut memiliki kemungkinan lebih mungkin untuk membuat *block* transaksi baru ke dalam *blockchain* (Yoon, 2018), sedangkan pada DPoS, *delegates* berfungsi untuk membuat *block* selanjutnya, memvalidasi *block* tersebut dan memasukkan *block* ke dalam *chain*. Pemasukan *block* ke dalam *chain* dilakukan setiap 10 (sepuluh) detik secara *default*. Kegiatan ini terus berlanjut hingga *delegate* terakhir.

Ketika *delegate* terakhir sudah selesai membuat *block*, maka sebuah putaran berhasil dilakukan dan putaran selanjutnya dilakukan dari *delegate* awal. *Delegate* yang berhasil membuat *block* akan mendapat insentif (Lisk *Documentation*, 2019).

Setiap pemegang *token* (*stakeholder*) dapat memilih perwakilan. *Stakeholder* dapat melakukan *voting* menggunakan *vote transaction*. Perwakilan memiliki peranan untuk menghasilkan *blocks* dan memvalidasi transaksi. Selama *blockchain network* masih berjalan, voting dapat dilakukan oleh *stakeholder*. Perubahan sebuah *stakeholder* menjadi *delegates* tergantung banyaknya *vote* yang diterimanya dalam jaringan tersebut. Perlu diketahui bahwa jumlah *delegates* hanya dibatasi sebanyak 101 perwakilan. Jika *delegates* tidak sampai 101, maka proses konsensus pun tidak dapat dijalankan (Hackfeld, 2019).

Block digunakan untuk menyimpan transaksi. Sebuah *block* dapat menyimpan sejumlah transaksi. Banyaknya transaksi dalam sebuah *block* tergantung dari konfigurasi dari *block* itu sendiri. Transaksi adalah sebuah objek yang digunakan untuk memperbaharui status dari akun. Lisk memiliki transaksi dasar (*default*) yang masing-masing memiliki tujuannya sendiri. Tabel 2.2 di bawah ini menjabarkan transaksi dasar pada Lisk (Lisk *Protocol*, 2019).

Tabel 2.2 Tipe-tipe transaksi pada Lisk

Transaction Type	***Purpose***
Balance transfer	*Transmit funds to another Lisk account*
delegate registration	*Register a delegate for the account*
Vote	*Submit or remove vote(s) for delegates*
Unlock vote	*Unlock locked tokens*

Tabel 2.2 Tipe-tipe transaksi pada Lisk (lanjutan)

Transaction Type	***Purpose***
Multisignature	*Register a multisignature in the account*
PoM	*Submit a proof-of-misbehavior of a delegate*

(Lisk *Protocol*, 2019)

Selain transaksi-transaksi di atas, pengembang dapat membuat transaksi sendiri (*custom transaction*). Pembuatan transaksi seperti ini dapat menyesuaikan dengan aplikasi *blockchain* yang dibuat. Konsep pembuatan *custom transaction* sangat mirip dengan *smart contract* karena transaksi ini juga memiliki logika bisnis khusus (Lisk *Blog*, 2019).

2.4.2 Genesis Block Lisk

Genesis block (blok awal) digunakan untuk mendefinisikan *block* pertama di *blockchain*. *Block* ini mendefinisikan status *blockchain* pada saat jaringan dimulai. *Genesis block* tidak dibuat oleh *delegate* seperti *block* lain, melainkan dibuat secara manual oleh developer ketika membuat aplikasi *blockchain*. Berikut ini adalah data-data yang terdapat dalam sebuah contoh *genesis block* (Lisk *Documentation*, 2019).

Gambar 2.1 Data-data pada *genesis block*

Berikut ini adalah definisi dari masing-masing data yang ada pada *genesis block* (Lisk *Documentation*, 2019).

1. `version` : versi dari *block*
2. `totalAmount` : jumlah *token* yang ditransfer ke dalam *block*
3. `totalFee` : jumlah biaya yang terasosiasi dengan *block*
4. `reward` : jumlah yang diterima untuk membuat *block*
5. `payloadHash` : *hash* dari penggabungan data transaksi dalam *block*
6. `timestamp` : catatan waktu saat *block* dibuat
7. `numberOfTransaction` : jumlah transaksi dalam sebuah *block*
8. `payloadLength` : jumlah data transaksi dalam bentuk *bytes*
9. `previousBlock` : *null* (kosong), karena *block* pertama
10. `generatorPublicKey` : *public key* dari *delegate* yang membuat *block*
11. `transactions` : daftar transaksi di dalam *genesis block*
12. `height` : ketinggian dari *block*
13. `blockSignature` : tanda tangan dari *block* oleh *delegate*

14. `id` : ID dari *block*

2.4.3 Konfigurasi Lisk

Lisk menggunakan sebuah *file* konfigurasi untuk mendefinisikan dan mengatur jalannya *blockchain*. Lisk menyediakan contoh *file* konfigurasi. Secara garis besar, konfigurasi mengatur tiga bagian, yaitu `app`, `components`, dan `modules` (Lisk *Documentation*, 2019). Contoh lengkap konfigurasi dapat dilihat pada Lampiran C. Berikut ini adalah garis besar isi konfigurasi Lisk.

Gambar 2.2 Konfigurasi Lisk

Berikut ini adalah penjelasan singkat dari gambar di atas (Lisk *Documentation*, 2019).

1. `genesisConfig` : mengatur pembuatan *block* dan *reward*
2. `ipc` : mengatur izin *modules* untuk berkomunikasi melalui IPCs
3. `logger` : mengubah opsi *log*
4. `storage` : mengatur koneksi Lisk ke PostgreSQL
5. `cache` : mengatur opsi untuk komponen *cache*
6. `http_api` : mengatur koneksi dan izin API
7. `chain` : mengatur persebaran *chain*, sinkronisasi, dll.
8. `network` : mengatur opsi dari koneksi *node*