Implementasi *State Compression* dengan *NFT (Non-Fungible Token) Pack* pada *Blockchain Solana* untuk Penjualan & Pembelian Paket *NFT* dalam *Game* berbasis *Web3*

Alfian Dwi Nugraha[1], I Made Suartana[2]

[1,2] Jurusan Teknik Informatika/Program Studi S1 Teknik Informatika, Universitas Negeri Surabaya
[1]alfian.20011@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— Penelitian ini bertujuan mengimplementasikan "NFT Pack" dalam ekosistem game Web3 di blockchain Solana dengan menggunakan teknologi State Compression untuk meningkatkan efisiensi dan keamanan transaksi NFT. Melalui integrasi paket NFT yang mendukung lebih dari satu NFT, penelitian ini bertujuan memberikan inovasi baru bagi pemain game Web3. Dengan penerapan State Compression, biaya transaksi dapat dikurangi dan efisiensi penyimpanan data ditingkatkan. Hasil penelitian menunjukkan bahwa implementasi NFT Pack dengan State Compression berhasil, dengan smart contract yang memfasilitasi pembelian, penjualan, dan interaksi NFT dalam paket. Pengujian menunjukkan pengaruh positif terhadap biaya transaksi dan efisiensi tanpa mengurangi keamanan system yang ada sebelumnya. Kesimpulan menghasilkan bahwa studi kasus NFT Pack berhasil dan terbukti aman, serta menyoroti potensi besar implementasi ini dalam meningkatkan efisiensi transaksi dan pengalaman pengguna di ekosistem game Web3. Penggunaan State Compression pada Compressed NFT menghasilkan penghematan biaya yang signifikan kurang lebih 95,3% dibandingkan NFT biasa dan potensi skalabilitas lebih tinggi. Saran untuk penelitian mendatang meliputi penelitian lebih lanjut mengenai keamanan dan skalabilitas, serta pengembangan fitur inovatif untuk meningkatkan interaksi pengguna dengan NFT.**

***Kata Kunci*— Blockchain, Non-Fungible Token, Smart Contract, Solana, Compressed NFT.**

I. PENDAHULUAN

Blockchain adalah inovasi teknologi yang telah merevolusi cara memandang keamanan dan transparansi dalam transaksi digital. Dengan asal usulnya yang erat kaitannya dengan kemunculan mata uang digital pertama yaitu Bitcoin, yang menandai dimulainya era baru dalam sistem pembayaran dan pertukaran informasi. Blockchain merupakan sebuah teknologi di mana informasi digital disimpan dalam basis data publik [1]. Bitcoin diusulkan sebagai mata uang kripto terdesentralisasi pertama, yang memungkinkan transaksi tanpa bergantung pada satu pihak ketiga, seperti bank [2]. Mata uang kripto sendiri adalah kependekan dari mata uang kriptografi, nama tersebut diambil karena proses validasi mata uang tersebut menggunakan konsep kriptografi dan secara digital.

Era digital semakin mendorong munculnya inovasi baru dalam teknologi Blockchain, termasuk Non-Fungible Token (NFT). NFT sebagai salah satu konsep untuk merepresentasikan aset digital telah mengubah paradigma kepemilikan aset dan interaksinya di dunia digital. NFT yang ada dalam Blockchain adalah item digital yang sering dikaitkan dengan materi digital unik, seperti musik atau foto [3]. Dalam konteks game berbasis Web3, NFT memberikan kemampuan untuk memiliki, memperdagangkan, dan mengumpulkan item digital dengan cara yang aman dan terverifikasi. Popularitas NFT di industri game terus meningkat berkat potensinya untuk memberikan pengalaman yang lebih kaya dan personalisasi yang lebih baik bagi para pemain.

Blockchain Solana telah menarik perhatian sebagai salah satu platform yang cocok untuk penerapan NFT, terutama karena kecepatan transaksinya yang tinggi dan biaya yang rendah. Hal ini menjadikannya kandidat ideal untuk menangani transaksi NFT dalam jumlah besar, seperti halnya pembelian dan penjualan NFT dalam game. Ada lebih dari 400 proyek dalam NFT, DeFi, dan aplikasi Web3 umum. Selain kinerjanya yang tinggi, Ini adalah salah satu Blockchain tercepat yang dapat diprogram di dunia kripto karena dapat menangani volume transaksi per detik (TPS) yang tinggi [10].

Blockchain Solana baru-baru ini telah mengeluarkan fitur baru yang disebut sebagai State Compression. Fitur ini memberikan banyak manfaat dan perubahan dalam ekosistem Solana dimasa yang akan datang. Salah satunya adalah NFT, dengan adanya State Compression ini, NFT dapat menjadi lebih murah dikarenakan ukuran data yang disimpan di Blockchain menjadi lebih kecil. Di Solana, State Compression adalah metode pembuatan "sidik jari" (atau hash) dari data off-chain dan menyimpan sidik jari ini secara on-chain untuk verifikasi yang aman [4]. Jadi, dengan adanya fitur terbaru dari Solana ini, data yang disimpan dalam Blockchain hanya berupa sidik jari dari data asli yang disimpan di luar Blockchain, sehingga dapat mengurangi biaya transaksi yang ada.

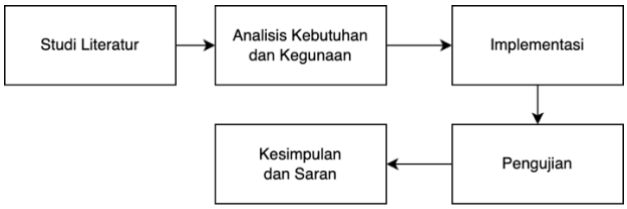
Maka dari itu, peneliti ingin mencoba melakukan penelitian tentang implementasi "NFT Pack" yang merupakan sebuah konsep paket NFT yang dapat berisi lebih dari satu NFT ke dalam game berbasis Web3 untuk memberikan inovasi dan pengalaman baru bagi para pemain game berbasis Web3. Serta menggunakan teknologi State Compression yang ada di Blockchain Solana untuk mengurangi biaya transaksi yang ada.

II. METODE PENELITIAN

Cara paling mudah untuk memenuhi persyaratan format penulisan adalah dengan menggunakan dokumen ini sebagai template. Kemudian ketikkan teks Anda ke dalamnya

Tujuan utama dan harapan peneliti adalah mengimplementasikan NFT Pack ke dalam platform Blockchain Solana untuk diterapkan dalam inovasi pembuatan game berbasis web3 atau penjualan-pembelian NFT dalam satu paket. Tidak lupa menerapkan teknologi terbaru dari Solana yaitu State Compression dengan harapan memperoleh suatu produk inovatif tetapi tidak terhalang dengan masalah keamanan, efisiensi, serta biaya transaksi. Fokus penelitian ini tentunya menerapkan solusi yang sudah dipaparkan terhadap masalah yang dihadapi serta memberi hasil berupa kesimpulan dari apa yang telah diimplementasikan di atas. Pada penelitian ini terbatas hanya menghasilkan produk dalam bentuk Smart Contract di Blockchain Solana dan Web untuk memanajemen NFT Pack serta platform untuk calon pengguna berinteraksi dengan produknya, serta diharapkan dilakukan pengujian menggunakan Unit Testing untuk membantu memastikan alur program sudah berjalan semestinya dan memberikan percobaan untuk membobol keamanan dari Smart Contract yang sudah dibuat.

Penelitian ini bertujuan untuk menyajikan wawasan praktis dan teoritis yang akan peneliti harap dapat berkontribusi pada literatur yang ada sekaligus memberikan rekomendasi untuk praktik terbaik dalam implementasi inovasi yang berhubungan dengan NFT dalam game berbasis Blockchain.



Gbr. 1 Alur Penelitian

Rancangan penelitian yang digunakan adalah dengan pendekatan pengembangan untuk memvalidasi penerapan teknik State Compression pada paket NFT di blockchain Solana. Rancangan ini dipilih untuk memungkinkan evaluasi langsung dari efek dan kinerja teknik kompresi dalam kondisi operasional nyata. Dalam rangka ini, sebuah prototype NFT pack akan dikembangkan dan diterapkan dalam ekosistem game Web3. Prototipe akan diuji dalam skenario nyata untuk mengamati performa transaksional, efisiensi penyimpanan data, serta interaksi pengguna saat melakukan pembelian dan pembukaan paket NFT. Data kinerja akan dikumpulkan secara sistematis dan dianalisis untuk menentukan manfaat konkret dari penggunaan State Compression dalam konteks blockchain Solana. Dengan demikian, rancangan penelitian ini tidak hanya akan menyediakan bukti empiris mengenai kepraktisan dari implementasi State Compression tetapi juga akan menyoroti implikasi nyata terhadap penggunaan NFT dalam game Web3 yang inovatif.

A. *Studi Literatur*

Peneliti mengambil informasi dari jurnal maupun penelitian yang telah dilakukan dari 2018-2023. Sumber yang akan digunakan untuk mencari jurnal adalah Mendeley, Google Scholar, dan beberapa sumber jurnal lainnya. Poin utama yang akan ditekankan adalah literatur tentang Blockchain serta NFT.

Compressed NFT adalah konsep baru yang dibuat menggunakan teknologi State Compression. Menggunakan State Compression untuk menyimpan Fingerprint (tanda tangan) dari metadata yang dibutuhkan NFT. Pada normalnya, NFT menyimpan metadata yang dibutuhkan ke dalam on-chain Solana, akibatnya user harus membayar biaya menjalankan transaksi (fee) serta biaya untuk menyimpan data (rent). Tetapi, dengan menggunakan teknologi State Compression, diharapkan metadata NFT tidak lagi disimpan dalam On-chain, tetapi disimpan di luar dari On-chain, dengan begitu kebutuhan verifikasi data yang diletakkan di luar dari on-chain harus diatasi, oleh karena itu di sini peran State Compression adalah sebagai penyimpan tanda tangan Fingerprint ke dalam On-chain, ini akan dibutuhkan nantinya untuk proses verifikasi untuk membandingkan apakah data yang disimpan diluar on-chain sesuai dengan hasil hash yang disimpan di on-chain menggunakan State Compression. Hal tersebut sesuai dengan tujuan Compressed NFT, Solusinya adalah dengan menyimpan sidik jari terkompresi dari data aset digital di blockchain sambil mempertahankan data aktual dengan solusi database tradisional [5].

B. *Analisa Kebutuhan dan Kegunaan*

Pada tahap ini, peneliti akan melakukan penentuan untuk persyaratan dan kebutuhan yang dibutuhkan atau dimungkinkan untuk perlu ada dalam proses implementasi NFT Pack. Berikut adalah rincian persyaratan fungsional.

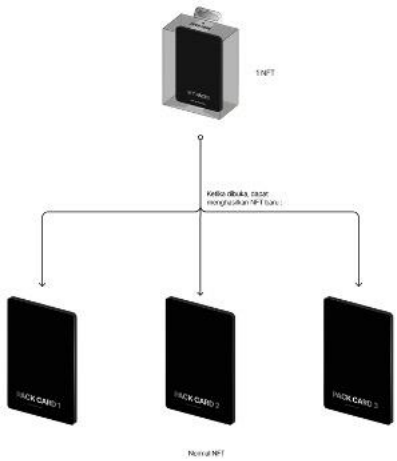
Pada tahap Rincian Persyaratan Fungsional, peneliti akan merincikan apa saja fungsional dari NFT Pack yang akan dibuat

TABEL I
FITUR PERSYARATAN FUNGSIONALITAS

No	Penjelasan Fitur
1	Membuat Pak Set Kartu
2	Menambah Kartu ke dalam Pak Set
3	Menghapus Kartu dari Pak Set
4	Menghapus Pak Set
5	Memperbarui Pak Set
6	Membuka Pak Kartu
7	Mendapatkan Hadiah Acak
8	Mengambil Hadiah dengan minting NFT semua kartu

Tabel diatas adalah fungsional dasar dari fitur yang akan dibuat dalam penelitian ini, menghadirkan penggunaan *Compressed NFT* dalam bentuk studi kasus membuka pak kartu, tentu ada kelompok fungsi untuk memanajemen kartu serta fungsi utamanya.

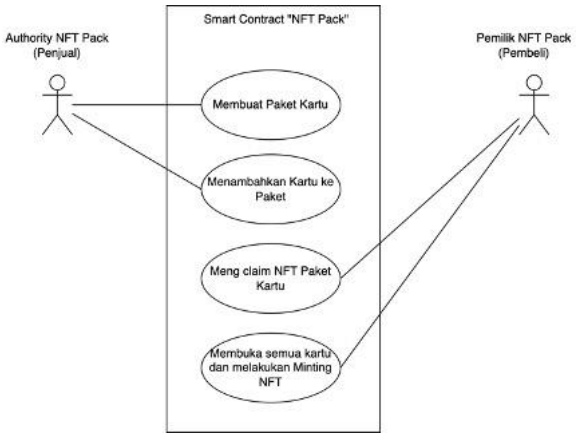
C. Implementasi



Gbr. 2 Visualisasi Produk

Diatas adalah contoh dari visualisasi produk yang akan dibuat, dimana menggambarkan ada NFT yang dapat di buka menjadi 3 NFT lainnya, ini diibaratkan NFT diatas adalah Kartu Pak, lalu dapat dibuka lagi menjadi NFT lainnya.

Pada tahapan ini, peneliti mulai melakukan implementasi dari NFT Pack. Berikut adalah gambaran Use Case sederhana dari menu NFT Pack yang diinteraksi oleh pengguna di Smart Contract nantinya.



Gbr. 3 Alur Interaksi Program

Di atas adalah diagram Use Case sederhana untuk melihat menu yang dapat diakses oleh pembeli atau penjual NFT Pack. Untuk rancangan dari aksi apa saja yang ada di Smart Contract sederhana seperti pada gambar diatas.

Dalam Smart Contract terkhususnya pada Blockchain Solana, tidak semua logika pemrograman pada umumnya bisa langsung diterapkan. Banyak aspek yang mempengaruhi dikarenakan Runtime Blockchain memiliki kriteria dan spesifikasi khusus untuk memastikan Smart Contract yang dijalankan aman. Ditambah, setelah peneliti melakukan beberapa studi literatur terkait Blockchain Solana, tidak semua variabel dapat dimasukkan kedalam Runtimenya. Oleh karena itu, beberapa proses harus dipecah menjadi beberapa bagian.

D. Pengujian

Tahapan ini dibutuhkan untuk menjawab dari rumusan masalah dan tujuan penelitian ini dibuat. Oleh karena itu, serangkaian beberapa pengujian dibuat untuk diharapkan dapat memberikan kesimpulan nantinya.

E. Kesimpulan dan Saran

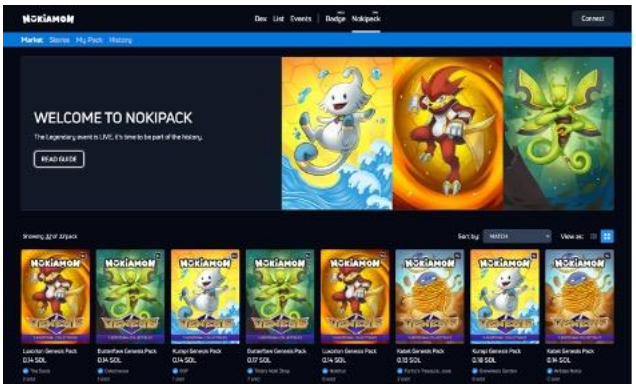
Di tahap ini, peneliti menyajikan hasil pengujian yang telah didapat dari presentase hasil keberhasilan dari Unit Testing serta metrik lainnya yang didapat dari proses penelitian ini seperti pengaruh dampak State Compression dari tahapan sebelumnya yang telah dipaparkan.

III. HASIL DAN PEMBAHASAN

A. Hasil

1) Studi Kasus Web Nokiamon

Nokiamon adalah sebuah ekosistem NFT yang beroperasi pada Solana, menyediakan pengalaman untuk mengumpulkan sejumlah aset digital yang terdiri dari ratusan entitas NFT yang beredar.



Gbr. 4 Tampilan Web Nokiamon

Diatas adalah contoh halaman depan dari Web Nokiamon, dimana memiliki tampilan sederhana yang menampilkan koleksi NFT yang sedang ada di ekosistemnya, dilengkapi juga halaman Marketplace untuk NFT Pack nantinya.

Nokiamon memiliki kesamaan dengan koleksi lain yang telah muncul, seperti Pokemon yang saat ini sudah dikenal oleh hampir semua orang. Pokemon memiliki berbagai media seperti film, permainan, atau bahkan koleksi aset dalam bentuk kartu fisik. Begitu juga dengan Nokiamon, yang menawarkan pengalaman serupa dengan perbedaan mendasar bahwa Nokiamon dirilis secara digital dan terintegrasi dengan teknologi Blockchain dalam jaringan Solana.

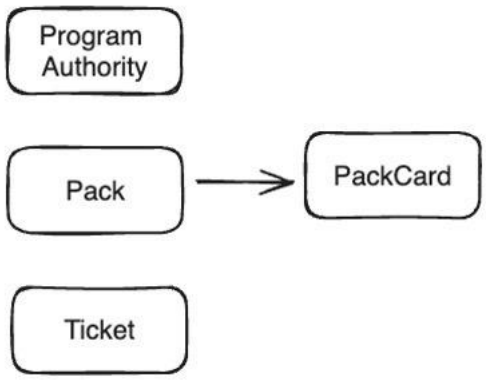
Dalam penelitian ini, peneliti terlibat dalam tim pengembangan Nokiamon dalam bidang Web dan Smart Contract. Oleh karena itu, penelitian ini didasarkan pada salah satu ide yang diusulkan untuk mewujudkan fitur dalam ekosistem Nokiamon, yaitu NFT Pack. Fitur ini memungkinkan pengguna untuk membeli satu NFT yang kemudian dapat dibuka dengan mengakses Smart Contract yang telah dibuat. Program akan melakukan pengundian hadiah, di mana Smart

Contract akan menghasilkan 1 hingga 3 hadiah NFT Nokiamon. Konsep ini mirip dengan contoh di atas ketika membeli paket kartu Pokemon, di mana setelah dibuka, pengguna bisa mendapatkan berbagai kartu secara acak.

Dengan keinginan untuk menciptakan fitur NFT Pack seperti yang dijelaskan sebelumnya, peneliti melakukan penelitian ini dengan tujuan mewujudkan pengalaman membeli NFT dalam bentuk paket kartu, sehingga pengguna dapat merasakan pengalaman yang serupa dengan ketika mereka membeli paket kartu dalam dunia nyata.

2) Merancang Smart Contract

Sebelum membuat semua persyaratan yang ada, hal pertama yang harus dilakukan oleh peneliti adalah membuat rancangan penyimpanan data yang nantinya akan disimpan di Smart Contract. Ini penting dilakukan di awal karena penyimpanan yang ada pada on-chain telah dibuat tidak dapat diubah dengan mudahnya. Oleh karena itu, penting untuk memastikan semua data yang akan disimpan sudah memenuhi persyaratan fungsional yang akan dibuat nantinya, agar tidak terjadi kesalahan di tengah jalan ketika program sudah siap. Karena ini akan sedikit menyulitkan pembaharuan selanjutnya.



Gbr. 5 Visualisasi Diagram Rancangan Data On-Chain

Diatas adalah gambar dari Rancangan Penyimpanan data yang akan dibuat, beberapa berdiri sendiri dan beberapa akan saling berhubungan seperti Pack dengan PackCard yang terhubung mirip relasi di basis data.

Mekanisme penyimpanan data di Blockchain Solana agak berbeda dari beberapa yang lain. Dimana Blockchain Solana tidak menyediakan akses penyimpanan secara langsung di Smart Contractnya. Kita ambil contoh di Blockchain Ethereum dimana Smart Contractnya dapat langsung menambahkan data secara langsung. Ini mirip dengan analogi Object Class dalam bahasa pemrograman yang Stateful.

Tetapi berbeda dengan Solana, penyimpanan harus dilakukan di luar dari Smart Contract itu sendiri, karena dalam Solana, Smart Contract bersifat Stateless. Artinya state apapun yang ada di Smart Contract akan hancur atau hilang disaat sudah tidak dijalankan lagi. Smart Contract hanya akan dipanggil ketika ada sebuah transaksi yang mengarah ke Smart Contract tersebut, dan ketika selesai maka semuanya akan hilang. Oleh karena itu tidak memungkinkan kita menyimpan datanya dalam bentuk state kedalam Smart Contractnya.

Jadi pada langkah ini, kita akan menulis semua perancangan data yang akan kita gunakan. Di sini peneliti menyediakan satu file khusus untuk membuat skema atau Struct untuk dipakai kedepannya. Berikut adalah salah satu cuplikan isinya.

state.rs

```
use anchor_lang::prelude::*;

#[account]
pub struct NokipackAuthority {
    pub guard: Pubkey, // 32
    pub trash: Pubkey, // 32
    pub pack_count: u32, // 4
    pub is_open: bool, // 1
    pub nft_symbol: String, // 4 + 40
    pub nft_collection: Pubkey, // 32
}
// space = 8 + 32 + 32 + 4 + 1 + (4 + 40) + 32 = 153
```

Gbr. 6 Struct untuk Program Authority

state.rs

```
#[account]
pub struct Pack {
    pub index: u32, // 4
    pub name: String, // 4 + 40
    pub image: String, // 4 + 255
    pub allowed_amount_to_redeem: u32, // 4
    pub supply: u128, // 16
    pub max_supply: u128, // 16
    pub price_mrp: u64, // 8
    pub price_mop: u64, // 8
    pub nft_name: String, // 4 + 40
    pub nft_uri: String, // 4 + 255
    pub card_count: u32, // 4
    pub is_open: bool, // 1
}
// space = 8 + 4 + (4 + 40) + (4 + 255) + 4 + 16 + 16 + 8 + 8 + (4 + 40) + (4 + 255) + 4 + 1 = 675

#[account]
pub struct PackCard {
    pub pack: Pubkey, // 32
    pub index: u32, // 4
    pub supply: u128, // 16
    pub max_supply: u128, // 16
    pub mint: Pubkey, // 32
    pub metadata: Pubkey, // 32
    pub edition: Pubkey, // 32
    pub token: Pubkey, // 32
    pub is_open: bool, // 1
    pub rarity: u32, // 4
}
// space = 8 + 32 + 4 + 16 + 16 + 32 + 32 + 32 + 32 + 1 + 4 = 209
```

Gbr. 7 Struct untuk Pack & Pack Card

state.rs

```
#[account]
#[derive(Default)]
pub struct Ticket {
    pub pack: Pubkey, // 32
    pub user: Pubkey, // 32
    pub nft: Pubkey, // 32
    pub supply: u128, // 16
    pub total_cards: u32, // 4
    pub total_cards_to_redeem: u32, // 4
    pub is_exhausted: bool, // 1
    pub is_redeemed: bool, // 1
    pub total_cards_redeemed: u32, // 4
    pub cards_to_redeem: Vec<u8>, // 4 + 400
    pub cards_redeemed: Vec<u8>, // 4 + 3400
}
// space = 8 + 32 + 32 + 32 + 16 + 1 + 1 + 4 + (4 + 400) + (4 + 3400)
```

Gbr. 8 Struct untuk Ticket

Selanjutnya adalah fitur fungsional, karena pada dasarnya program ini terbagi menjadi dua bagian dasar yaitu manajemen data serta proses intinya nanti, untuk proses manajemen data hanya untuk melakukan seperti menambah atau mengubah datanya bahkan sampai menghapus datanya,

dalam daftar fungsionalitas yang sudah di rancangkan sebelumnya, berikut mungkin beberapa yang bisa diberikan sebagai kode contoh, karena kode yang asli sangat panjang, oleh karena itu hanya beberapa yang akan peneliti tampilkan.

initialize_pack_authority.rs

```
use {
  anchor_lang::prelude::*,
  crate::{
    constant::{
      PDA_SPACE_NOKIPACK_AUTHORITY,
      SEED_NOKIPACK_AUTHORITY_PREFIX,
    },
    state::NokipackAuthority,
  },
};

pub fn initialize_pack_authority(
  ctx: Context<InitializePackAuthorityContext>,
  nft_symbol: String,
  nft_collection: Pubkey,
) -> Result<()> {
  let reward_program_authority = &mut ctx.accounts.nokipack_authority;
  reward_program_authority.guard = ctx.accounts.user.key();
  reward_program_authority.trash = ctx.accounts.trash.key();
  reward_program_authority.pack_count = 0;
  reward_program_authority.is_open = true;
  reward_program_authority.nft_symbol = nft_symbol;
  reward_program_authority.nft_collection = nft_collection;

  Ok(())
}
```

Gbr. 9 Fungsi intruksi initialize_pack_authority

Berikut diatas Ini adalah instruksi pertama yang bertujuan untuk "Inisialisasi Program Authority". Ketika instruksi ini dijalankan, tujuannya adalah untuk membuat data dari "Program Authority" yang telah dirancang sebelumnya. Ini merupakan fungsi manajemen data pada umumnya, di mana hanya menulis data On-chain yang diambil dari parameter ketika pengguna menjalankan instruksinya. Data On-chain ditulis dari parameter pengguna tersebut, intruksi yang berjalan juga harus kita definisikan accounts apa saja yang akan dibutuhkan Ketika runtime nanti, berikut adalah structnya.

initialize_pack_authority.rs

```
#[derive(Accounts)]
pub struct InitializePackAuthorityContext<'info> {
  /// CHECK: We're about to create this with Anchor
  #[account(
    init,
    payer = user,
    space = PDA_SPACE_NOKIPACK_AUTHORITY,
    seeds = [
      SEED_NOKIPACK_AUTHORITY_PREFIX.as_bytes(),
      crate::id().as_ref(),
    ],
    bump,
  )]
  pub nokipack_authority: Account<'info, NokipackAuthority>,

  #[account(mut)]
  pub user: Signer<'info>,

  /// CHECK: We're about to check this with Anchor
  pub trash: AccountInfo<'info>,

  pub system_program: Program<'info, System>,
  pub rent: Sysvar<'info, Rent>,
}
```

Gbr. 10 Struct Accounts intruksi initialize_pack_authority

Contoh kedua selanjutnya adalah berikut, instruksi ini berfungsi untuk memperbarui data yang ada di On-chain. Hampir mirip seperti sebelumnya, hanya saja ketika menjalankan instruksi ini kita tidak menggunakan dekorator "init" di Account dari nokipack_authority. Dengan begitu, Account nokipack_authority akan dianggap sebagai Account yang sudah ada dan akan diedit.

update_pack_authority.rs

```
use {
  anchor_lang::prelude::*,
  crate::{
    constant::{
      SEED_NOKIPACK_AUTHORITY_PREFIX,
    },
    state::NokipackAuthority,
    utils::assert_account_key,
  },
};

pub fn update_pack_authority(
  ctx: Context<UpdatePackAuthorityContext>,
  nft_symbol: String,
  nft_collection: Pubkey,
  is_open: bool,
) -> Result<()> {
  let reward_program_authority = &mut ctx.accounts.nokipack_authority;

  // CHEKCS
  // check: user must be the guard
  assert_account_key(
    &ctx.accounts.user.to_account_info(),
    &reward_program_authority.guard,
    "user must be the guard",
  )?;

  // update
  reward_program_authority.is_open = is_open;
  reward_program_authority.nft_symbol = nft_symbol;
  reward_program_authority.nft_collection = nft_collection;

  Ok(())
}
```

Gbr. 11 Fungsi intruksi update_pack_authority

update_pack_authority.rs

```
#[derive(Accounts)]
pub struct UpdatePackAuthorityContext<'info> {
  #[account(
    seeds = [
      SEED_NOKIPACK_AUTHORITY_PREFIX.as_bytes(),
      crate::id().as_ref(),
    ],
    bump,
  )]
  #[account(mut)]
  pub nokipack_authority: Account<'info, NokipackAuthority>,

  #[account(mut)]
  pub user: Signer<'info>,

  pub system_program: Program<'info, System>,
  pub rent: Sysvar<'info, Rent>,
}
```

Gbr. 12 Struct Accounts intruksi update_pack_authority

Jadi, itulah beberapa kode yang digunakan untuk manajemen data di program. Selanjutnya, peneliti akan membahas tentang fungsi utama dari program ini, yaitu proses mulai dari Minting NFT Pack, lalu proses Request Redeem NFT Pack, dan Unpacking atau Redeem NFT Pack, sampai dengan Claim Pack Card NFT Pack pada bagian selanjutnya, sekaligus membahas tentang keamanannya.

Selanjutnya adalah beberapa pembahasan tentang keamanan, seperti berikut, yang pertama adalah masalahh hak akses.

update_pack_authority.rs

```rust
pub fn update_pack_authority(
    ctx: Context<UpdatePackAuthorityContext>,
    ...
) -> Result<()> {
    // always load program authority account to access config
    let program_authority = &mut ctx.accounts.nokipack_authority;

    // CHEKCS
    // check: user must be the guard
    assert_account_key(
        &ctx.accounts.user.to_account_info(),
        &program_authority.guard,
        "user must be the guard",
    )?;

    // main logic
    ...

}
```

Gbr. 13 Kode Contoh Cek Hak Akses

Pada setiap instruksi, pastikan semua pengguna yang mengakses harus ditangani dengan benar. Sebagai contoh, pada fungsi untuk menambahkan pak di kode yang ada di atas, penggunaan fungsi bawaan assert bisa sangat membantu. Dengan menggunakan assert, kita dapat memastikan bahwa pengguna yang mengakses adalah pengguna yang diizinkan, dan memastikan bahwa setiap instruksi hanya dapat dijalankan oleh pengguna yang ditentukan. Hal ini penting untuk menghindari orang lain yang tidak dikehendaki untuk mengakses instruksi tersebut. Sebagai contoh, fungsi instruksi admin seharusnya hanya dapat diakses oleh admin yang telah ditentukan di data "Program Authority". Jika pengguna bukanlah admin, maka instruksi akan ditolak, seperti yang ditunjukkan dalam kode di atas.

Dengan demikian, pembatasan hak akses ini menjadi langkah penting dalam memastikan keamanan dari fungsionalitas yang telah dibuat sebelumnya.

Selanjutnya, kita akan membahas tentang proses minting Compressed NFT. Proses ini merupakan langkah untuk membuat NFT Pack dan mengalokasikannya kepada pengguna. Proses ini juga dapat dikembangkan dengan tambahan fitur pembelian, di mana pengguna akan membeli NFT Pack dengan harga yang ditentukan.

Code Contoh Mint Compressed NFT

```rust
mpl_bubblegum::cpi::mint_to_collection_v1(
  CpiContext::new(
    ctx.accounts.bubblegum_program.to_account_info(),
    mpl_bubblegum::cpi::accounts::MintToCollectionV1 {
      tree_authority: ctx.accounts.tree_authority.to_account_info(),
      leaf_owner: to_user.to_account_info(), // to_user
      leaf_delegate: marketplace_authority.to_account_info(), // marketplace_authority
      merkle_tree: ctx.accounts.merkle_tree.to_account_info(),
      payer: user.to_account_info(), // user payer
      tree_delegate: creator.to_account_info(), // creator
      collection_authority: creator.to_account_info(), // creator
      collection_authority_record_pda: ctx.accounts.bubblegum_program.to_account_info(),
      collection_mint: ctx.accounts.collection_mint.to_account_info(),
      collection_metadata: ctx.accounts.collection_metadata.to_account_info(),
      edition_account: ctx.accounts.edition_account.to_account_info(),
      bubblegum_signer: ctx.accounts.bubblegum_signer.to_account_info(),
      log_wrapper: ctx.accounts.log_wrapper.to_account_info(),
      compression_program: ctx.accounts.compression_program.to_account_info(),
      token_metadata_program: ctx.accounts.token_metadata_program.to_account_info(),
      system_program: ctx.accounts.system_program.to_account_info(),
    },
  ),
  mpl_bubblegum::state::metaplex_adapter::MetadataArgs {
    collection: Some(mpl_bubblegum::state::metaplex_adapter::Collection {
      verified: false,
      key: nokipack_authority.nft_collection
    }),
    uri: pack.nft_uri.to_string(),
    creators,
    name: pack.nft_name.to_string(),
    symbol: nokipack_authority.nft_symbol.to_string(),
    primary_sale_happened: false,
    seller_fee_basis_points: 500,
    is_mutable: true,
    edition_nonce: None,
    token_program_version: mpl_bubblegum::state::metaplex_adapter::TokenProgramVersion::Original,
    token_standard: Some(mpl_bubblegum::state::metaplex_adapter::TokenStandard::NonFungible),
    uses: None,
  }
)?;
```

Gbr. 14 Kode Mint Compressed NFT

Fungsi ini akan melakukan minting NFT seperti biasanya, namun dengan menggunakan Compressed NFT. NFT Pack yang diminting ke pengguna akan digunakan sebagai tanda bahwa pengguna tersebut sudah memiliki NFT Pack, yang nantinya dapat ditukarkan menjadi hadiah dalam proses selanjutnya.

Mempersiapkan Metadata Untuk Validasi

```rust
// CHECK: metadata hash must be same
let new_metadata_collection = mpl_bubblegum::state::metaplex_adapter::Collection {
  key: nokipack_authority.nft_collection,
  verified: true,
};
let new_metadata_creators = &mut Vec::new();
for creator in metadata_creators.clone() {
  new_metadata_creators.push(mpl_bubblegum::state::metaplex_adapter::Creator {
    address: creator.address,
    share: creator.share,
    verified: creator.verified,
  });
}
let new_metadata_build = mpl_bubblegum::state::metaplex_adapter::MetadataArgs {
  name: metadata.name.clone(),
  symbol: metadata.symbol.clone(),
  uri: metadata.uri.clone(),
  edition_nonce: None,
  is_mutable: true,
  primary_sale_happened: false,
  seller_fee_basis_points: metadata.seller_fee_basis_points.clone(),
  uses: None,
  collection: Some(new_metadata_collection),
  creators: new_metadata_creators.to_owned(),
  token_program_version: mpl_bubblegum::state::metaplex_adapter::TokenProgramVersion::Original,
  token_standard: Some(mpl_bubblegum::state::metaplex_adapter::TokenStandard::NonFungible),
};
```

Gbr. 15 Kode membuat Hash Dari metadata sekarang

Kalkulasi Hash Sekarang Dengan Hash State Compression

```rust
// compute hash
let new_metadata_args_hash = keccak::hashv(&[new_metadata_build.try_to_vec()?.as_slice()]);
let new_data_hash = keccak::hashv(&[
  &new_metadata_args_hash.to_bytes(),
  &new_metadata_build.seller_fee_basis_points.to_le_bytes(),
]).to_bytes();

// CHECK: hash
if data_hash != new_data_hash {
  return Err(mpl_bubblegum::error::BubblegumError::DataHashMismatch.into());
}
```

Gbr. 16 Kode mengkalkulasi Hash dan mencocokan

Selanjutnya, terdapat proses Redeem Pack. Ini adalah fungsi yang dijalankan ketika pengguna telah melewati proses validasi sebelumnya. Di dalam fungsi ini, tujuannya adalah memastikan bahwa pengguna mendapatkan hadiah secara acak sesuai dengan Pack yang dipilih atau dibuka. Dengan demikian, proses ini memastikan pengalaman yang adil bagi pengguna dalam mendapatkan hadiah dari NFT Pack yang mereka buka.

Kalkulasi Rnadom Pack Card

```
// get slot hash
let recent_slothashes_info = &ctx.accounts.recent_slothashes.to_account_info();
let data = recent_slothashes_info.try_borrow_data()?;
let most_recent_slothash = array_ref![data, 8, 8];
let raw_random_value: u16 = get_random_value(
  most_recent_slothash,
  &ctx.accounts.clock,
  &ticket.to_account_info(),
  i,
)?;
let mut random_value = ((raw_random_value as f64 / u16::MAX as f64) * possibilites_pack_cards.len() as f64).floor() as
u32;

// safe random checker
let min_val = 0 as u32;
let max_val = (possibilites_pack_cards.len() - 1) as u32;
if random_value < min_val {
  random_value = min_val;
}
if random_value > max_val {
  random_value = max_val;
}

msg!("resutl random pack card index : {:?}", random_value);
```

Gbr. 17 Kode memilih kartu acak

Kode di atas yang digunakan untuk menghasilkan pengacakan angka di Solana terlihat kompleks karena Solana memiliki fitur simulasi transaksi yang mempengaruhi proses pengacakan angka. Pengacakan dilakukan secara tidak langsung untuk menghindari penyalahgunaan pengguna yang dapat melihat hasil simulasi dan mengubah perilakunya berdasarkan hasil tersebut sebelum transaksi dieksekusi di blockchain.

Dalam proses pengacakan angka, beberapa fungsi seperti slothash, clock, dan prooving account digunakan sebagai kiat untuk melakukan hashing agar hasilnya lebih acak. Setelah itu, hasil hashing dikonversi ke dalam bentuk bit dan diambil sebagai index untuk menentukan hadiah kartu mana yang akan diberikan kepada pengguna.

Fungsi Random Value

```
pub fn get_random_value(
  recent_slothash: &[u8],
  clock: &Clock,
  proving_process_account: &AccountInfo,
  index: u32,
) -> Result<u16> {
  // Hash make new random u16
  let mut hasher = DefaultHasher::new();
  hasher.write(recent_slothash); // recent slothash
  hasher.write_u64(clock.slot); // slot
  hasher.write_i64(clock.unix_timestamp); // timestamp

  // make more different hash for each instruction in the same slot
  hasher.write(proving_process_account.try_borrow_data()?.as_ref());
  hasher.write(&index.to_le_bytes());

  let mut random_value: [u8; 2] = [0u8; 2];
  random_value.copy_from_slice(&hasher.finish().to_le_bytes()[..2]);
  Ok(u16::from_le_bytes(random_value))
}
```

Gbr. 18 Kode fungsi acak angka

B. *Pembahasan*

1) Pengujian Perbandingan Biaya Tradisional NFT dan Compressed NFT

Pada tahap ini, peneliti akan melakukan pengujian dari implementasi sistem yang sudah dibangun dari langkah sebelumnya, menghasilkan beberapa pengujian yang digunakan untuk menjawab tujuan dari penelitian ini. Berikut adalah pengujiannya:

TABEL III
PERBANDINGAN NFT DENGAN COMPRESSED NFT

Jumlah	Tradisional *NFT*	Compressed NFT	Rasio
1 juta *NFT*	12 ribu SOL	5 SOL	2,400x
1 miliar *NFT*	12 juta SOL	500 SOL	24,000x

Berikut adalah pengujian terhadap Compressed NFT. Di bab 3 sebelumnya, peneliti merancang pengujian untuk membandingkan NFT normal dengan Compressed NFT. Mari kita lihat terlebih dahulu apa yang telah dikeluarkan oleh Metaplex selaku komunitas yang mengembangkan teknologi standarisasi NFT pada Solana.

Dari perhitungan yang dikeluarkan oleh Metaplex, dapat dilihat perbandingan yang luar biasa dari biaya untuk mencetak atau minting NFT dengan Compressed NFT yang menggunakan teknologi State Compression. Perbedaan harga yang sangat signifikan ini tentu dapat mengubah ekosistem NFT pada Solana jika diterapkan secara menyeluruh. Pada saat artikel penelitian ini ditulis, NFT normal masih mendominasi pasar NFT di Solana karena Compressed NFT baru stabil dirilis beberapa bulan sebelumnya. Tentu saja, pasar tidak akan langsung berubah secara keseluruhan untuk menggunakan Compressed NFT ini, tetapi sudah banyak proyek yang muncul dalam penggunaan Compressed NFT ini dan juga mempromosikan teknologi ini sendiri.

Sebagai gantinya, dalam Compressed NFT karena menggunakan teknologi State Compression, ini artinya minting Compressed NFT tidak dapat dilakukan secara langsung. Kita harus membuat yang disebut Merkle Tree, yang merupakan penyimpanan dari data hash yang berbentuk pohon. Merkle Tree ini harus dibuat terlebih dahulu untuk menjadi induk para Compressed NFT nantinya. Karena Merkle Tree adalah Account Solana atau penyimpanan data dalam Solana, ukurannya harus ditentukan di awal dan bersifat Fixed. Oleh karena itu, sebelum membuat Merkle Tree, harus dihitung berapa jumlah Compressed NFT yang akan bersarang di dalamnya.

Sebagai pengujian di sini, peneliti mencoba untuk membuat 1 juta Compressed NFT. Berikut adalah spesifikasi Merkle Tree yang akan dibuat:

Gbr. 19 Perhitungan dari "https://compressed.app"

Diatas adalah salah satu spesifikasi pembuatan Merkle Tree untuk 1 juta NFT. Mengapa menghasilkan angka di sekitar 7 SOL? Itu karena seperti namanya, Merkle Tree berbentuk pohon yang bercabang 2 daun. Artinya, untuk dapat menampung sebanyak 1 juta daun, kita perlu menghitung 2 pangkat berapa yang hasilnya sama atau mendekati di atas 1 juta, yaitu 2 pangkat 20 menghasilkan 1.048.576, ini mendekati dan di atas dari 1 juta.

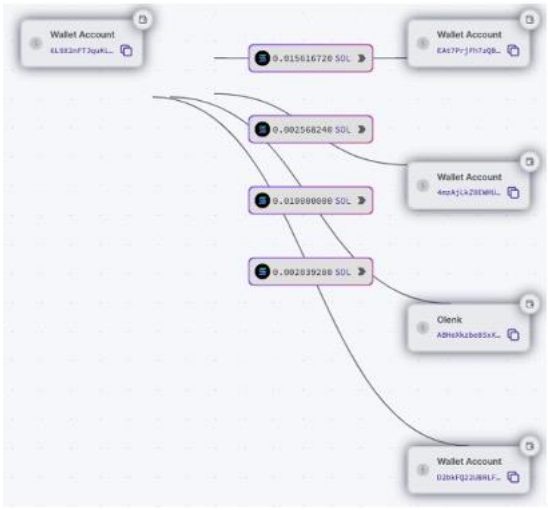
Selanjutnya, tidak lupa beberapa spesifikasi Merkle Tree lainnya, lalu kita hitung menggunakan fungsi yang telah disediakan oleh Metaplex, untuk mendapatkan biaya estimasi untuk penyimpanan di Solana nanti, kita perlu menghitung berapa lamports yang dibutuhkan untuk rent datanya.

kalkulasi Ukuran dan Harga Untuk Merkle Tree

```
const depthSizePair = {
  maxDepth: 20,
  maxBufferSize: 64,
  canopyDepth: 14,
}
const space = getConcurrentMerkleTreeAccountSize(
  depthSizePair.maxDepth,
  depthSizePair.maxBufferSize,
  depthSizePair.canopyDepth
)
const lamports = await connection.getMinimumBalanceForRentExemption(space)
consola.log('space:', space) // output: space: 1092792
consola.log(
  'lamports:', lamports,
  'sol:', lamports / LAMPORTS_PER_SOL
) // output: lamports: 7606723200 sol: 7.6067232
```

Gbr. 20 Visualisasi Perhitungan ukuran dari Merkle Tree

Didapat harga sekitar 7.6 SOL, yang mana sekali pembuatan Merkle Tree ini dapat menjadi induk untuk 1.048.576 Compressed NFT. Ini bisa dibilang cukup efisien dan murah dibanding dengan harga pembuatan satu NFT normal seperti berikut.



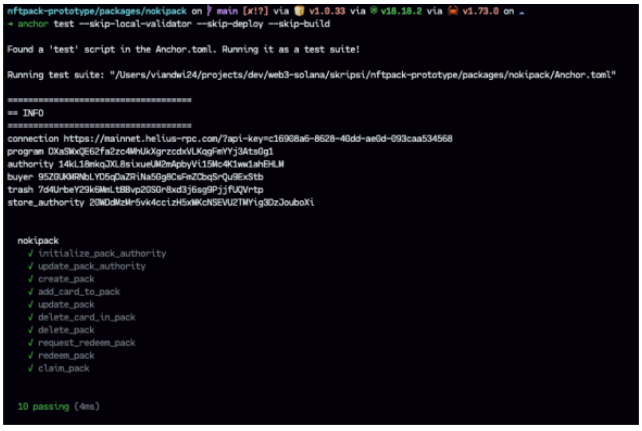
Gbr. 21 Visualisasi percobaan dari minting NFT

Dari percobaan peneliti melakukan minting 1 NFT itu sekitar 0.02 - 0.03 SOL. Jika dibulatkan, anggap saja 0.02 SOL. Bayangkan jika minting 1 juta NFT, maka akan menjadi 20.000 SOL. Ini sungguh harga yang sangat fantastis jika kita bandingkan dengan Compressed NFT.

Jadi jika kita ambil nilai tertinggi misalnya dari 1 juta adalah 58 SOL, sekarang kita hitung juga jika rata – rata transaksi fee yang ada pada Solana adalah 0.0005 SOL, jika kita kali sebanyak 1 juta maka total fee nya adalah 500 SOL, kita tambahkan dengan 58 SOL hasilnya adalah 558 SOL, ini adalah harga yang kita bayar untuk total keseluruhan proses pembuatan Compressed NFT. Jika kita bandingkan 12ribu SOL dengan 558 SOL, ini artinya ada sekitar pengurangan 95.35% biaya, ini sungguh biaya yang sangat fantastis jika langsung dibandingan dengan teknologi ini.

2) Pengujian Fungsionalitas Smart Contract

Berikut adalah dokumentasi dalam hasil pengujian dengan Unit Test yang telah dilakukan terhadap 10 fungsionalitas dasar dari Smart Contract yang telah dibuat pada tahapan implementasi.



Gbr. 22 Dokumentasi Unit Test semua Unit

Berikut adalah tabel hasil pengujian unit yang sudah dilakukan diatas, akan ditampilkan beberapa Langkah juga yang dilakukan selama pengujian berlangsung.

TABEL IIII
TABEL PENGUJIAN FUNGSIONALITAS

ID	Description	Pre Condition	*Test* Steps	Expected Result	Actual Result	Status
FT001	Inisialisasi Program *Authority*	Program Baru Saja Dideploy	1. Menjalankan Instruksi ”Initialize Program *Authority*” 2.. Memastikan Account Program *Authority* Terbuat	Transaksi Berhasil Account Berhasil Terbuat	Transaksi Berhasil Account Berhasil Terbuat	Passed
FT002	Update Program Authority	*Authority* Program Telah Dibuat	1. Menjalankan Instruksi ”*Update* Program *Authority*” 2. Memastikan Account Program *Authority* Telah Diperbarui	Transaksi Berhasil Account Berhasil Diperbarui	Transaksi Berhasil Account Berhasil Diperbarui	Passed
FT003	Menambahkan Pak Set	*Authority* Program Telah Dibuat	1. Menjalankan Instruksi ”Add *Pack*” 2. Memastikan Account *Pack* Telah Dibuat	Transaksi Berhasil Account Berhasil Dibuat	Transaksi Berhasil Account Berhasil Dibuat	Passed
FT004	Menambahkan Kartu kedalam Pak Set	Pak Set Telah Dibuat	1. *Minting NFT Master Edition* 2. Menjalankan Instruksi ”Add *Pack*” 3. Memastikan Account *Pack Card* Telah Dibuat	Transaksi Berhasil Account Berhasil Dibuat	Transaksi Berhasil Account Berhasil Dibuat	Passed
FT005	Menghapus Pak Set	Pak Set Telah Dihapus	1. Menjalankan Instruksi ”Delete *Pack*” 2. Memastikan Account *Pack* Telah Dihapus	Transaksi Berhasil Account Berhasil Dihapus	Transaksi Berhasil Account Berhasil Dihapus	Passed
FT006	Menghapus Kartu didalam Pak	Pak *Card* Telah Dibuat	1. Menjalankan Instruksi ”Delete *Pack Card*” 2. Memastikan Account *Pack Card* Telah Dihapus	Transaksi Berhasil Account Berhasil Dihapus	Transaksi Berhasil Account Berhasil Dihapus	Passed
FT007	*Update* Pak Set	Pak Set Telah Diperbarui	1. Menjalankan Instruksi ”*Update Pack*” 2. Memastikan Account *Pack* Telah Diperbarui	Transaksi Berhasil Account Berhasil Diperbarui	Transaksi Berhasil Account Berhasil Diperbarui	Passed
FT008	Membuka Pak *NFT*	Pak Set Telah Terbuat dan Mempunyai Pak *NFT*	1. Menjalankan Instruksi ”*Request Redeem Pack*” 2. Memastikan Account Ticket Telah Dibuat 3. Memastikan *NFT* Pak Telah di burn	Transaksi Berhasil Account Berhasil Dibuat *NFT* Berhasil Diburn	Transaksi Berhasil Account Berhasil Dibuat *NFT* Berhasil Diburn	Passed
FT009	Mendapatkan Hasil Hadiah Acak	Pak Set Telah Diperbarui	1. Menjalankan Instruksi ”*Redeem Pack*” 2. Memastikan Account Ticket Telah Mendapat Hasil Acak	Transaksi Berhasil Account Berhasil Diperbarui	Transaksi Berhasil Account Berhasil Diperbarui	Passed
FT010	Mengambil Hadiah *NFT*	Tiket telah diredeem dan hadiah telah ditetapkan	1. Menjalankan Instruksi ”*Claim Pack*” 2. Memastikan Account Ticket Telah Diset Hangus 3. Memastikan Hadiah *NFT* Sudah didapat	Transaksi Berhasil Account Berhasil Diperbarui *NFT* Berhasil Didapat	Transaksi Berhasil Account Berhasil Diperbarui *NFT* Berhasil Didapat	Passed

IV. KESIMPULAN

Implementasi State Compression pada NFT Pack dalam blockchain Solana berhasil mengurangi biaya transaksi yang signifikan, mencatat penghematan biaya sebesar 95%. Hal ini tidak hanya menunjukkan peningkatan efisiensi ekonomi tetapi juga potensi untuk skalabilitas yang lebih besar. Penelitian ini berhasil mengimplementasikan teknologi state compression pada NFT Pack dalam blockchain Solana, menunjukkan pengurangan biaya transaksi yang signifikan, serta meningkatkan efisiensi dan skalabilitas dalam ekosistem NFT. Dengan adanya State Compression yang diterapkan pada NFT menghasilkan teknologi baru yang bernama Compressed NFT, dari hasil pengujian yang dilakukan telah berhasil menurunkan biaya yang lumayan signifikan dengan perbandingan yang sangat jauh dari tradisional NFT biasa, dari pengujian yang telah dilakukan telah terjadi penurunan biaya hampir 95,3%, dari yang berkisar 0,012 SOL menjadi 0,00058 SOL.

Pengujian fungsionalitas dasar dari smart contract menunjukkan bahwa sistem yang dikembangkan beroperasi sesuai dengan spesifikasi dan aman dari perspektif keamanan data serta transaksi, dengan memanfaatkan validasi metadata NFT dan mekanisme akses kontrol. Dari pengujian yang telah dilakukan dan telah dirilis ke jaringan Mainnet dan telah digunakan oleh banyak orang, membuktikan bahwa sampai penelitian ini ditulis semuanya masih aman dan semua fungsi telah bekerja dengan baik, ditunjukan pada semua hasil unit testing yang ada, lalu analitik Smart Contract untuk beberapa waktu kebelakang membuktikan semuanya masih bekerja dan berfungsi stabil seperti yang tertera pada bab sebelumnya.

Integrasi fitur NFT Pack berbasis state compression ke dalam website Nokiamon memberikan pengalaman baru

kepada pengguna dalam koleksi dan perdagangan NFT, mendekatkan pengalaman digital dengan pengalaman membeli pak kartu fisik. Sampai saat ini seperti yang dijelaskan di pengujian, implementasi dalam studi kasus Web Nokiamon telah bekerja dan telah digunakan oleh puluhan user dan masih berfungsi. Ini juga bisa dilihat pada tampilan pada Web Nokiamon di halaman Leaderboard, yang menampilkan sejumlah user yang berarti beberapa user merasa puas untuk tetap menggunakan fitur dari implementasi sistem yang ada pada penelitian ini, dibuktikan dengan aktifnya user dalam Leaderboard.

REFERENSI

[1] S. Albeshr and H. Nobanee, "Blockchain Applications in Banking Industry: A Mini-Review," *SSRN Electronic Journal*, 2020, doi: 10.2139/ssrn.3539152.

[2] T. Zhang and Z. Huang, "Blockchain and central bank digital currency," ICT Express, vol. 8, no. 2, 2022, doi: 10.1016/j.icte.2021.09.014.

[3] D. Ghelani, "What is Non-fungible token (NFT)? A short discussion about NFT Terms used in NFT," Authorea, vol. 4, 2022, doi: 10.22541/au.166490992.24247550/v1.

[4] Solana, "State Compression." [Online]. Available: https://docs.solana.com/learn/state-compression

[5] A. Y. Jarry Xiao, Noah Gundotra, Austin Adams, "Compressing Digital Assets with Concurrent Merkle Trees," no. 1, 2022.

[6] A. N. Hasibuan and T. Dirgahayu, "Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna," Jurnal Informatika Universitas Islam Indonesia, vol. 2, no. 1, 2020.

[7] M. Xu, X. Chen, and G. Kou, "A systematic review of blockchain," Financial Innovation, vol. 5, no. 1. 2019. doi: 10.1186/s40854-019-0147-z.

[8] D. Sheridan, J. Harris, F. Wear, J. Cowell, E. Wong, and A. Yazdinejad, "Web3 Challenges and Opportunities for the Market." arXiv, 2022. doi: 10.48550/ARXIV.2209.02446.

[9] K. Carlson, "The Nakamoto Blockchain." Nov. 2018. doi: 10.13140/RG.2.2.17991.34723.

[10] S. Bhujel and Y. Rahulamathavan, "A Survey: Security, Transparency, and Scalability Issues of NFT's and Its Marketplaces," Sensors, vol. 22, no. 22, 2022, doi: 10.3390/s22228833.

[11] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system, October 2008," Cited on, 2008.

[12] A. Yakovenko, Solana: A new architecture for a high performance blockchain v0.8.13. 2019.

[13] S. Edgard and J. S. Suroso, "DESIGN OF NFT SMART CONTRACT SYSTEM USING ETHEREUM NETWORK BLOCKHCAIN TECHNOLOGY," Journal of Theoretical and Applied Information Technology, vol. 101, no. 13, 2023.