Implementasi *Blockchain* untuk Distribusi Kunci Publik Terdesentralisasi

Devin Alvaro
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
devin.alvaro@gmail.com

***Abstract* — Penyebaran kunci publik saat ini membutuhkan *certificate authority*, otoritas sentral yang menjamin *authenticity* dan *integrity* dari kunci publik. Akan tetapi, otoritas sentral seperti demikian merupakan sebuah kelemahan dari segi keamanan, dikarenakan adanya aspek *trust* yang terlibat di dalamnya. Di sisi lain, *blockchain* merupakan teknologi yang sedang meningkat popularitasnya karena penggunaannya pada *cryptocurrency*. Dengan *blockchain*, sebuah sistem terdentralisasi dapat mendistribusikan data secara aman dan benar tanpa dapat dimanipulasi oleh penyerang atau pihak ketiga. Meskipun sering dikaitkan dengan *cryptocurrency*, konsep *blockchain* itu sendiri dapat diimplementasi untuk berbagai kebutuhan lain, seperti distribusi kunci publik. Pada makalah ini, penulis mengeksplorasi dan melakukan eksperimen dengan mengimplementasi *blockchain* untuk distribusi kunci publik. Setelahnya, penulis memberikan analisis perbandingan distribusi kunci publik dengan *blockchain* dan dengan *certificate authority*, serta *feasibility* penggunaannya di dunia nyata.**

Keywords — public key, public key infrastructure, digital certificate, certificate authority, blockchain, hash, proof-of-work

I. Pendahuluan

Keamanan digital banyak melibatkan kriptografi kunci publik seperti *RSA* dan *ECC (Ecliptic-Curve Cryptography)*. Salah satu kegunaan kriptografi kunci publik adalah sebagai metode yang aman untuk bertukar kunci kriptografi simetris seperti *AES (Advanced Encryption System)* pada saluran yang tidak aman, misalnya Internet.

Dalam kriptografi kunci publik, penerima pesan membangkitkan kunci publik dan privat, kemudian menyebarkan kunci publiknya ke pengirim pesan. Setelah itu, pengirim pesan mengenkripsi pesan dengan kunci publik tersebut kemudian mengirimnya ke penerima pesan. Dengan kriptografi kunci publik, pesan terenkripsi tersebut hanya dapat didekripsi dengan kunci privat, sehingga hanya penerima pesan asli yang dapat mendekripsi pesan karena kunci privat hanya dimiliki olehnya (tidak disebar).

Akan tetapi, penyebaran kunci publik seperti demikian memiliki beberapa kelemahan. Pertama, penerima kunci publik tidak dapat memastikan apakah kunci publik tersebut memang benar berasal dari pengirim kunci publik yang sebenarnya (aspek *authenticity*). Kedua, penerima kunci publik tidak dapat memastikan apakah kunci publik tidak diubah oleh penyerang atau pihak ketiga (aspek *integrity*).

Saat ini, sebagai solusi permasalahan tersebut, penyebaran kunci publik membutuhkan otoritas sentral berupa *certificate authority* yang menerbitkan sertifikat digital, yaitu sertifikat yang menyatakan dan menjamin kebenaran suatu kunci publik. Dengan sertifikat digital, penerima kunci publik dapat memastikan aspek *authenticity* dan *integrity* dari kunci publik yang diterimanya. Akan tetapi, sistem seperti demikian memiliki kelemahan dari segi keamanan yaitu dibutuhkannya otoritas sentral yang terpercaya, dalam hal ini *certificate authority*, sehingga ada aspek *trust* yang terlibat.

Sebagai solusi dari permasalahan tersebut, penulis mengajukan penggunaan *blockchain* untuk distribusi kunci publik terdesentralisasi. *Blockchain* adalah sistem terdistribusi *peer-to-peer* untuk mencatat seluruh *data/record* (disebut *block*) pada suatu jaringan secara aman. Secara singkat, *blockchain* merupakan kumpulan *block* yang berisi index, data, dan *hash* dari *block* sebelumnya (*previous hash*). Keamanan *blockchain* terletak pada mekanisme *proof-of-work* dan sebuah konsensus yang membuat sebuah *node* hanya menerima terusan *blockchain* dari *node* lain apabila *blockchain* tersebut tidak di-*compromise*. Sifatnya yang terdesentralisasi menyebabkan aspek *trust* tidak lagi diperlukan dalam mendistribusikan data pada jaringan. Penjelasan lebih dalam mengenai *blockchain* akan dijelaskan pada bab selanjutnya.

Popularitas *blockchain* sedang meningkat karena penggunaannya untuk Bitcoin dan Ethereum, dua *cryptocurrency* terpopuler saat ini. Meskipun sering dikaitkan dengan *cryptocurrency*, konsep *blockchain* itu sendiri dapat diaplikasikan dalam berbagai jenis sistem lain untuk penyebaran data terdesentralisasi yang aman.

Pengaplikasian *blockchain* ini dapat diterapkan terutama pada sistem yang melibatkan otoritas sentral, seperti penyebaran kunci publik dengan *certificate authority*.

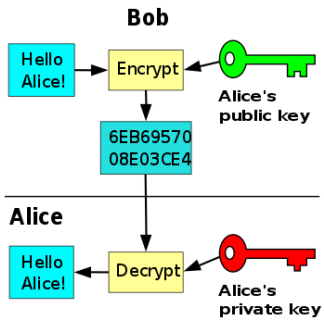
II. Dasar Teori

I. *Public Key*

Dalam kriptografi kunci publik, digunakan sepasang kunci, yaitu kunci publik dan kunci privat. Kunci publik bersifat publik karena dapat disebar ke siapapun, sedangkan kunci privat bersifat rahasia karena hanya diketahui oleh pembangkit kedua kunci tersebut.

Untuk bertukar pesan yang terenkripsi dengan kriptografi kunci publik, pertama-tama pengirim membangkitkan kunci publik dan kunci privat. Kemudian, kunci publik dikirim ke pengirim pesan, di mana pengirimannya tidak bersifat rahasia sehingga kunci publik dapat diketahui pihak manapun. Setelah menerima kunci publik, pengirim pesan mengenkripsi pesannya dengan kunci publik tersebut. Setelah terenkripsi, pesan hanya dapat didekripsi dengan kunci privat. Pesan yang terenkripsi tersebut kemudian dikirim ke penerima pesan yang kemudian mendekripsinya dengan kunci privat yang hanya diketahui olehnya.

Tidak seperti kriptografi simetris, kriptografi kunci publik bersifat asimetris, yang berarti kunci untuk mengenkripsi pesan tidak sama dengan kunci untuk mendekripsinya. Pada prakteknya, kriptografi kunci publik kebanyakan digunakan untuk bertukar kunci kriptografi simetris secara aman. Hal ini dikarenakan algoritma kriptografi kunci publik biasanya mahal secara komputasional, sehingga hanya praktis untuk mengenkripsi pesan pendek, seperti kunci kriptografi simetris. Contoh algoritma kriptografi kunci publik adalah *RSA* dan *ECC*.



Gambar 1. Skema kriptografi kunci publik.
https://upload.wikimedia.org/wikipedia/commons/thumb/f/f9/Public_key_encryption.svg/375px-Public_key_encryption.svg.png

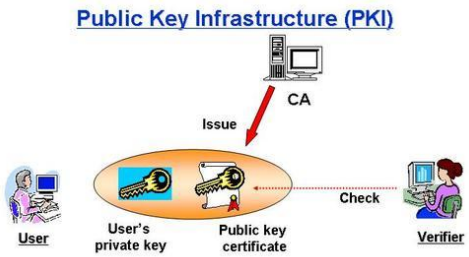
II. *Digital Certificate*, *Certificate Authority*, dan *Public Key Infrastructure*

Pada bagian sebelumnya telah disebutkan bahwa internet menggunakan kriptografi kunci publik untuk mengamankan komunikasi pada jaringan, terutama dengan protokol *HTTPS (HTTP Secure)*. Tidak seperti protokol *HTTP* yang tidak mengenkripsi pesan, protokol *HTTPS* mengenkripsi pesan dengan algoritma kriptografi simetris. Pertukaran kunci simetris ini disebut *TLS handshake*, di mana salah satu pihak mengirimkan kunci publiknya, kemudian pihak satunya mengenkripsi kunci simetris dengan kunci publik tersebut, lalu pihak sebelumnya mendekripsi kunci simetris tersebut dengan kunci privatnya.

Namun, terdapat masalah yang muncul pada mekanisme ini, yaitu bagaimana penerima kunci publik dapat memastikan bahwa kunci publik dikirim dari pengirim yang benar (aspek *authenticity)* dan isinya benar tidak diubah (aspek *integrity*). Maka dari itu, digunakan sertifikat digital, yaitu sertifikat berisi kunci publik dan identitas pengirim kunci publik. Dengan adanya sertifikat digital, penerima kunci publik dapat memeriksa kebenaran kunci publik dengan mencari sertifikat yang berkaitan dengan kunci publik tersebut pada repositorinya.

Sertifikat digital diterbitkan oleh *certificate authority* (*CA*), yaitu badan yang dipercaya untuk memeriksa kebenaran kunci publik dan menerbitkan sertifikat digital untuk menjaminnya. Sistem dengan *CA* ini berbasis kepercayaan, di mana pihak-pihak pada jaringan harus percaya bahwa *CA* jujur dan kompeten dalam menjamin kebenaran sertifikat digital. Biasanya, sistem operasi dan *web browser* menyematkan beberapa sertifikat digital yang benar-benar terpercaya secara *default*.

Kumpulan aturan dan prosedur yang melibatkan sertifikat digital dan *certificate authority* dalam distribusi kunci publik ini secara kesatuan disebut sebagai *public key infrastructure*.

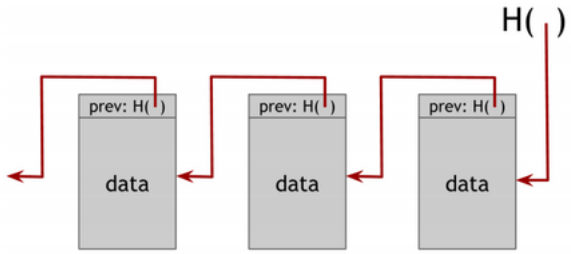


Gambar 2. Skema *public key infrastructure* dengan *certificate authority* dan *digital certificate*.
https://dypmusfs8hvdw.cloudfront.net/data3/HT/JQ/MY-7613701/pki-500x500.jpg

III. *Blockchain*

Blockchain adalah teknologi yang sedang meningkat popularitasnya karena digunakan pada *cryptocurrency* seperti Bitcoin dan Ethereum. Pada dasarnya, *blockchain* adalah struktur data seperti *linked list* berisi kumpulan *block* berisi data/*record* (disebut *transaction*) dan *hash* dari *block* sebelumnya (*previous hash*). Karena *block-block* seperti membentuk rantai (*chain*), maka struktur data ini disebut *blockchain*. Perlu diperhatikan bahwa meskipun *blockchain* sering berkaitan dengan *cryptocurrency*, *transaction* di sini adalah *database*

transaction, bukanlah transaksi ekonomi.



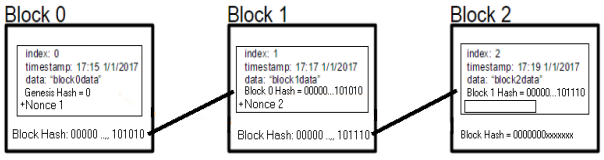
Gambar 2. Gambaran struktur data *blockchain*. https://qph.fs.quoracdn.net/main-qimg-75c5b83ba56f0deff282a64b60f1962a

Blockchain juga dapat dilihat sebagai sebuah basis data terdistribusi, di mana data tereplikasi secara penuh pada setiap *node* pada jaringan (setiap *node* memiliki *full copy*). Setiap *node* pada sistem terdistribusi memiliki hak yang sama dengan setiap *node* lainnya, yaitu hak untuk menerima rantai *blockchain* ter-*up-to-date* dan menambahkan data ke rantai *blockchain* apabila memenuhi syarat tertentu. Maka dari itu, dapat dikatakan *blockchain* merupakan sistem terdesentralisasi tanpa keterlibatan otoritas sentral.

Properti utama dari *blockchain* adalah sifatnya yang *append-only*, yaitu *blockchain* hanya dapat ditambahkan data baru, sedangkan data-data yang telah ditambahkan tidak dapat diubah maupun dihapus. Dengan properti ini, *blockchain* menjamin keamanannya sebagai basis data terdistribusi dan terdesentralisasi, yakni apabila sebuah *node* telah berhasil menambahkan data ke *blockchain*, data tersebut akan selalu ada di seluruh *blockchain* di setiap *node*, tanpa bisa diubah maupun dihapus. Selanjutnya, akan dijelaskan bagaimana *blockchain* menjaga properti *append-only* ini, secara spesifiknya dengan menggunakan *proof-of-work* dan *longest-chain consensus*.

Pertama-tama akan dijelaskan mekanisme penambahan *block* ke *blockhain*. Ketika menambahkan *block* ke *blockchain*, *hash* dari *block* tersebut harus memenuhi sebuah syarat, yaitu *n* karakter pertama dari *hash*-nya haruslah karakter tertentu. Misalnya *hash* "0000727ea4009f7c463475a9a1eb87ef", di mana pada *hash* tersebut 4 karakter pertamanya adalah angka 0. Nilai *n* ini ditentukan oleh *difficulty* dari *blockchain*, semakin tinggi nilai *n*, tentu semakin sulit *hash* dicari.

Karena sebuah *block* memuat data spesifik, belum tentu *hash* dari *block* langsung memenuhi syarat tersebut, oleh karena itu ditambahkanlah *nonce*, yaitu sebuah angka yang dapat membuat *hash* dari *block* memenuhi syarat yang disebutkan di atas.



Gambar 3. Gambaran *nonce* pada *block* untuk membentuk *hash* yang memenuhi syarat *proof-of-work*. http://www.amarketplaceofideas.com/wp-content/uploads/2017/04/BlockChain-Schematic.png

Syarat *hash* seperti demikianlah yang disebut *proof-of-work*. Karena *hash* merupakan *one-way function*, maka satu-satunya cara mendapatkan *hash* yang memenuhi *proof-of-work* adalah dengan melakukan *brute force* untuk mencari *nonce* yang membuat *hash* dari *block* memenuhi *proof-of-work*. Operasi ini mahal secara komputasional, terutama pada *difficulty* yang tinggi. Maka dari itu, *proof-of-work* adalah cara bagi *blockchain* untuk mempersulit penambahan dan pengubahan *block* ke *blockchain*.

Selanjutnya, akan dijelaskan bagaimana setiap *node* pada jaringan *blockchain* menyelaraskan *blockchain*-nya. Pada intinya, *blockchain* di jaringan yang terpanjang dianggap paling *authoritative*. Tentunya, diperiksa dahulu validitas setiap *blockchain* dengan memeriksa *hash*-nya seperti telah dijelaskan di atas. Ketika sebuah *node* akan menambahkan *block*, *node* tersebut terlebih dahulu mencari *blockchain* terpanjang pada jaringan untuk meng-*update* *blockchain*-nya. Setelahnya, barulah *block* ditambah ke *blockchain*. Sebuah *node* harus melakukan hal ini karena *blockchain* terpanjang adalah yang paling *authoritative*, sehingga *node* tersebut perlu membuat *blockchain*-nya yang terpanjang juga agar dapat diterima oleh *node* lain pada jaringan.

Setelah penjelasan mengenai *proof-of-work* dan *longest-chain consensus* pada *blockchain*, selanjutnya akan dianalisis keamanan *blockchain* dari sisi penyerang.

Apabila seorang penyerang mengubah sebuah *block* B_1, *hash* dari *block* tersebut pun berubah, sehingga ia harus mengubah isi *previous hash* dari *block* B_2. Dengan demikian, *hash* dari *block* B_2 juga akan berubah, maka *previous hash* dari *block* B_3 pun harus diubah, dan seterusnya hingga *block* terakhir. Perlu diingat bahwa mengubah isi dari *block* akan mengubah *hash*-nya, maka si penyerang pun harus mencari *nonce-nonce* baru yang memenuhi *proof-of-work* untuk seluruh *block* yang diubah tersebut, yang mana sangat sulit sekali secara komputational. Terlebih lagi, agar dianggap valid sebuah *blockchain* harus diterima oleh mayoritas *node*. Maka dari itu, seorang penyerang harus menyerang mayoritas *node* pada jaringan *blockchain* secara bersamaan, bersama dengan pencarian *proof-of-work*-nya. Karena hal ini sangat sulit dilakukan, inilah yang membuat *blockchain* begitu aman sebagai *append-only distributed ledger*, yaitu bila sebuah data telah masuk ke dalam jaringan *blockchain*, sulit sekali untuk diubah maupun dihapus.

III. Eksperimen

Eksperimen pada makalah ini merupakan implementasi dari *blockchain* sederhana yang bertujuan sebagai *proof-of-concept* bahwa *blockchain* dapat digunakan untuk distribusi kunci publik. Pertama-tama, akan ditunjukkan implementasi *blockchain* secara umum lalu dilanjutkan dengan implementasi penggunaannya untuk distribusi kunci publik. Perlu diperhatikan bahwa kode implementasi yang ditunjukkan pada makalah ini hanya sebagian kode yang menurut penulis terpenting. Kode implementasi selengkapnya dapat dilihat pada repositori

project yang dibuat oleh penulis berikut: https://github.com/devinalvaro/chain.

I. Implementasi *Blockchain* secara Umum

Berikut adalah implementasi struktur data *blockchain*:

```
Blockchain:
  chain               []Block
  currentTransactions []Transaction
  difficulty          int
  genesisHash         string
```

Seperti telah dijelaskan pada bagian sebelumnya, chain adalah kumpulan dari *block*, currentTransactions adalah kumpulan transaksi yang belum di-*confirm*, difficulty adalah tingkat kesulitan *proof-of-work*, dan genesisHash adalah *dummy hash* pertama dari *blockchain*, karena setiap *block* termasuk *block* pertama memerlukan *previous hash*.

Ketika sebuah *node* bergabung pada jaringan *blockchain*, *node* tersebut mencari seluruh *node* pada jaringan kemudian mengumumkan *address*-nya pada semua *node* tersebut. *Node-node* yang menerima *address* ini akan menambahkan *address* tersebut ke sebuah set berisi *address* semua *node* pada jaringan. Beginilah cara setiap *node* pada jaringan terhubung.

Selanjutnya adalah struktur data dari *block*:

```
Block:
  index        int
  prevHash     string
  nonce        int
  transactions []Transaction
  timestamp    Timestamp
```

Struktur data *block* memiliki index, yaitu indeks *block* pada *blockchain* (dimulai dari 0), prevHash yaitu *hash* dari *block* sebelumnya (*previous hash*), *nonce* yaitu angka yang digunakan untuk membuat *hash* dari *block* memenuhi syarat *proof-of-work*, transactions adalah transaksi-transaksi yang di-*confirm* oleh *block* tersebut, dan timestamp.

Struktur data selanjutnya adalah *transaction*. Seperti telah disebutkan pada bagian sebelumnya, *transaction* adalah *database transaction* dari *blockchain* itu sendiri, dan datanya dapat berupa apa saja. Pada implementasi *blockchain* di makalah ini sendiri, terdapat dua tipe transaction; (1). *transaction* bertipe *digital certificate*: berisi informasi kunci publik seperti sertifikat digital dan (2). *transaction* bertipe *authentication address*: berisi informasi autentikasi *address*. Kedua tipe *transaction* ini akan dijelaskan kemudian.

Selanjutnya akan ditunjukkan fungsi-fungsi penting dari implementasi *blockchain* ini.

Salah satu fungsi terpenting dari *blockchain* adalah validasi *chain* yang bertujuan untuk menjaga kemanan *blockchain* dari penyerang yang mengubah/menghapus data atau menambah data tanpa *proof-of-work*. Berikut adalah *pseudocode* validasi *blockchain* beserta penjelasannya:

```
func isChainValid(chain)
  for i := 1; i < len(chain); i++
    if not isHashValid(chain[i].hash())
      return false
    if chain[i - 1].hash() != chain[i].prevHash
      return false

  return true

func isHashValid(hash)
  return hash[:difficulty] == "0" * difficulty
```

Cara kerja validasi *blockchain* adalah dengan mengiterasi setiap *block*, kemudian memeriksa beberapa hal berikut, yaitu; (1). apakah *hash* dari *block* tersebut valid, yakni memenuhi syarat *proof-of-work* dan (2). apakah *hash* dari *block* sebelumnya sama dengan *prevHash block* sekarang. Seperti telah dijelaskan sebelumnya, kedua pengecekan ini memerankan peranan penting dalam menjaga keamanan *blockchain*.

Di bawah fungsi `isChainValid()` adalah fungsi `isHashValid()` yang memeriksa apakah sejumlah karakter pertama dari *hash* merupakan karakter tertentu (agar memenuhi syarat *proof-of-work)*. Pada implementasi ini, karakter tertentu itu adalah '0' dan jumlah karakter pertama yang dibutuhkan sama dengan '0' ditentukan oleh *difficulty*. Dari fungsi ini, terlihat mengapa mencari *proof-of-work* sangat sulit terutama untuk *difficulty* yang lebih tinggi. Kesulitan ini dikarenakan semakin kecil kemungkinan mendapatkan *hash* yang memenuhi syarat tersebut dengan melakukan *brute force* pada *nonce*.

Selanjutnya adalah fungsi `mineBlock()`, yaitu fungsi untuk *block mining*, yaitu mencari *nonce* yang membuat *hash* dari *block* memenuhi syarat *proof-of-work* pada fungsi `isHashValid()`. Cara kerja fungsi ini semata-mata melakukan *brute force* untuk mencari *nonce* yang sesuai. Setelah *block mining* berhasil, *block* akan ditambahkan ke *blockchain* dan *transactions* yang belum di-*confirm* akan di-*confirm* dengan memasukkannya pada *block* tersebut.

```
func mineBlock(blockchain, block)
  block.nonce := 0
  while not isHashValid(block.hash())
    block.nonce += 1

  block.transactions =
blockchain.currentTransactions
  blockchain.currentTransactions = nil

  blockchain.add(block)
```

Fungsi berikutnya adalah `synchronizeChain()`, yaitu fungsi yang digunakan sebuah *node* untuk menyelaraskan *blockchain*-nya dengan *blockchain* seluruh *node* pada jaringan. Fungsi ini dipanggil tepat sebelum dilakukan *block mining* pada saat penambahan *block* baru.

```
func synchronizeChain(blockchain)
  maxLength := len(blokchain.chain)
  longestChain := nil

  for node := range nodes
    chain := fetchNodeChain(node.address)

    if len(chain) > maxLength &&
```

```
        isChainValid(chain, bc.Difficulty)
      maxLength = len(chain)
      longestChain = chain

  if longestChain != nil
    bc.Chain = longestChain
```

Sebelumnya telah disebutkan bahwa konsensus dari jaringan *blockchain* adalah memilih *chain* terpanjang yang valid. Inilah yang terjadi pada potongan kode di atas, di mana *node* n_x yang memanggil fungsi melakukan *fetching* terhadap *blockchain* dari semua *node* yang ada di jaringan, kemudian mencari yang terpanjang di antaranya. Apabila *chain* yang terpanjang bukan merupakan milik *node* sekarang, maka *chain* dari *node* sekarang akan ditimpa dengan *chain* terpanjang tersebut. Sebaliknya, bila *chain* dari *node* n_x adalah yang terpanjang, maka tidak dilakukan apa-apa. *Node* n_x tidak perlu menginformasikan *node* lainnya bahwa *chain*-nya adalah yang terpanjang karena biarlah *node* lain menyelaraskan *chain*-nya pada saat *node* tersebut akan menambahkan *block* baru saja.

II. Penggunaan Implementasi *Blockchain* untuk Distribusi Kunci Publik

Setelah dijabarkan mengenai implementasi sederhana *blockchain* secara umum, selanjutnya akan dijelaskan penggunaan implementasi ini untuk distribusi kunci publik. Karena data pada *blockchain* terletak pada *transaction*, maka kunci publik dan informasi lain yang berhubungan diletakkan pada *transaction* menyerupai sertifikat digital. Berikut adalah struktur data *transaction* bertipe *digital certificate* untuk distribusi kunci publik pada *blockchain*:

```
Transaction:
  // type: digital certificate
  address   string
  publicKey string
```

address pada *transaction* merupakan URL pemilik kunci publik, dan publicKey adalah kunci publik itu sendiri. Tentu saja, masih terdapat banyak atribut lain pada sertifikat digital sebenarnya, akan tetapi pada makalah ini hanya disematkan 2 atribut terpenting tersebut untuk simplisitas.

Dengan sifat *blockchain* yang menjaga data dari penghapusan atau pengubahan oleh pihak penyerang, maka penyebaran *certificate* dengan *blockchain* seperti demikian telah memecahkan masalah *integrity* dari distribusi kunci publik.

Masalah selanjutnya adalah *authenticity*, yaitu pemeriksaan apakah kunci publik benar berasal dari sumber yang sebenarnya. Pada implementasi *blockchain* biasa seperti Bitcoin, ketika sebuah *node* bergabung ke jaringan *blockchain*, *node* tersebut membangkitkan sepasang kunci privat dan kunci publik, kemudian menyebarkan kunci publiknya melalui *public key infrastructure* terpisah. Ketika sebuah *node* tersebut mengirimkan transaksi ke *node* lain, *node* tersebut juga melakukan *signing* terhadap transaksi tersebut dengan kunci privatnya. Ketika *node* lain menerima transaksi tersebut, transaksi tersebut diperiksa kebenarannya dengan kunci publik *node* pengirim. Kunci publik setiap *node* disebar dengan *digital certificate* yang dibuat *certificate authority* seperti pada *public key infrastructure* biasa. Metode seperti ini adalah metode verifikasi dengan *digital signature*.

Akan tetapi cara seperti demikian tidak dapat berlaku pada topik makalah ini, karena justru tujuan implementasi *blockchain* pada makalah ini adalah menyebarkan kunci publik secara terdesentralisasi tanpa *certificate authority*. Sayangnya, penulis tidak menemukan pemecahan masalah ini yang tetap membuat *blockchain* murni terdesentralisasi.

Metode yang diajukan oleh penulis adalah seperti demikian. Selain *transaction* berupa sertifikat digital, di dalam *blockchain* juga terdapat *transaction* bertipe *address authentication* seperti demikian: (perlu diingat bahwa *transaction* dalam *blockchain* dapat berupa data apapun)

```
Transaction:
  // type: address authentication
  address          string
  addressPublicKey string
```

address merupakan *address* dari *node*, sedangkan addressPublicKey akan dijelaskan setelah ini. Perlu diperhatikan bahwa addressPublicKey pada *transaction* bertipe *address authentication* berbeda dengan *publicKey* pada *transaction* bertipe *digital certificate*.

Idenya adalah ketika sebuah *node* bergabung ke jaringan *blockchain*, *node* tersebut membangkitkan sepasang kunci privat (*addressPrivateKey*) dan kunci publik (*addressPublicKey*). Kemudian, *node* tersebut mengirimkan *address* beserta *addressPublicKey* ke sebuah otoritas sentral yang penulis sebut *authentication authority* (*AA*). Di sini, *AA* bertugas memverifikasi kebenaran *address* dari *node* tersebut. Apabila telah terverifikasi, *AA* akan menambahkan *transaction* bertipe *address authentication* ke *blockchain*. Hanya *AA* yang dapat menambahkan *address authentication* ke *blockchain*. Dengan *address authentication* tersebut, setiap *node* dapat memiliki informasi *address* dan *addressPublickey* semua *node* lain pada jaringan karena kedua informasi tersebut terdapat dalam *blockchain*.

Cara kerja autentikasi *address* menjadi sebagai berikut. Ketika sebuah *node* biasa (bukan *authentication authority*) menambahkan *transaction* bertipe *digital certificate* ke *blockchain*, *node* tersebut menambahkan *signature*, yaitu *hash* dari *transaction* yang dienkripsi dengan *addressPrivateKey*-nya. Berikut adalah gambaran struktur data *transaction* bertipe *digital certificate* yang telah ditambah *signature*:

```
Transaction:
  // type: digital certificate
  address   string
  publicKey string
  signature string
```

Kemudian, node yang memeriksa *transaction* memeriksa validitasnya dengan fungsi seperti berikut:

```
func isTransactionValid(node, transaction)
  decryptSign := decrypt(transaction.signature,
node.addressPublicKey)

  if transaction.hash() != decryptSign
    return false
  else
    return true
```

Seperti terlihat, pertama-tama *node* yang memeriksa *transaction* mendekripsi *signature* dengan *addressPublicKey* milik *node* pengirim *transaction*. Selanjutnya, *signature* terdekripsi tersebut diperiksa dengan *hash* dari *transaction*. Apabila sama, maka *transaction* tersebut valid, bila tidak maka tidak valid. Mekanisme ini sama dengan verifikasi dengan *digital signature*, sehingga dapat dipastikan *address* yang terdapat di *transaction* bertipe *digital certificate* ini benar berasal dari *node* yang mengirimnya.

Sebagai contoh, organisasi Mozilla berniat bergabung ke *blockchain* untuk mendistribusikan *public key* untuk situs webnya. Pertama-tama, Mozilla membangkitkan sepasang kunci publik (*addressPublicKey*) dan kunci privat (*addressPrivateKey*), kemudian mendaftarkan *address*-nya, https://mozilla.org ke *authentication authority* (*AA*). Kemudian, *AA* memverifikasi bahwa *address* yang didaftarkan sudah benar, kemudian menambahkan informasi *address* beserta *addressPublicKey* milik Mozilla ke dalam *blockchain*. Kemudian, pengunjung situs web https://mozilla.org dapat mencari *transaction* bertipe *digital certificate* yang memuat *publicKey* situs web tersebut pada *blockchain*, kemudian memastikan autentikasi *transaction* tersebut dengan *addressPublicKey* milik Mozilla yang diisukan oleh *AA* pada *blockchain*.

Dengan demikian, masalah kedua keamanan distribusi kunci publik, *authentication*, telah terpecahkan. Meskipun pada akhirnya diperlukan otoritas sentral berupa *authentication authority*, sehingga *blockchain* tidak murni terdesentralisasi. Analisis mengenai solusi ini akan dijabarkan pada bagian selanjutnya.

IV. Analisis

Pada bagian ini, akan dianalisis perbandingan distribusi kunci publik dengan *certificate authority* dan dengan *blockchain* serta *feasibility* penggunaannya di dunia nyata.

Pada bab Pendahuluan, disebutkan bahwa permasalahan utama distribusi kunci publik adalah; (1). *authentication*, yaitu bagaimana penerima kunci publik memastikan bahwa kunci publik yang diterimanya berasal dari pengirim yang sebenarnya dan (2). *integrity*, yaitu bagaimana penerima kunci publik memastikan bahwa kunci publik yang diterimanya tidak diubah atau dihapus pada proses pengiriman oleh pihak ketiga.

Dengan distribusi kunci publik dengan *blockchain*, masalah kedua, *integrity*, terpecahkan karena *blockchain* adalah struktur data/basis data yang bersifat *append-only*, yakni data hanya bisa ditambahkan ke *blockchain* dan data yang telah ditambah ke *blockchain* tidak dapat diubah maupun dihapus. Penjelasan mengenai sifat ini telah dijelaskan pada bab Dasar Teori.

Yang menjadi masalah adalah *authenticity*, karena pada implementasi *blockchain* biasa (misalnya Bitcoin), digunakan mekanisme *signing* menggunakan kunci privat dan kunci publik, sedangkan kunci publiknya disebar seperti biasa menggunakan *certificate authority*. Tentu saja, tidak mungkin implementasi pada makalah ini menggunakan bantuan *certificate authority*, karena tujuan utama topik makalah ini adalah menghilangkan campur tangan otoritas sentral dalam *public key infrastructure*. Akan tetapi, pada akhirnya penulis tidak menemukan cara *authentication* pada *blockchain* yang murni terdesentralisasi tanpa otoritas sentral. Cara terbaik yang diusulkan oleh penulis adalah menggunakan *authentication authority* seperti yang dijelaskan pada bab Eksperimen.

Meskipun menggunakan *authentication authority* (*AA*) berarti *blockchain* tetap melibatkan otoritas sentral, namun penggunaan *AA* pada *blockchain* ini memiliki keuntungan signifikan dibanding menggunakan *certificate authority*. Seperti dijelaskan sebelumnya, *AA* bertugas memverifikasi *address* dari *node* yang bergabung ke jaringan *blockchain*, lalu menambahkan informasi kunci publik *address*-nya (*addressPublicKey)* ke *blockchain.* Informasi ini dapat digunakan untuk memverifikasi *address* pengirim *transaction* melalui mekanisme *digital signing*. Dengan demikian, tugas sebuah *AA* tidaklah kurang dan tidaklah lebih, hanya memverifikasi *address* dari *node* yang bergabung ke jaringan *blockchain* dan menambahkan informasinya ke *blockchain*.

Berbeda dengan *certificate authority* yang bertugas memverifikasi setiap *public key* yang ada. Ditambah lagi, berarti menggunakan *authentication authority* ini menghilangkan potensi otoritas sentral memanipulasi verifikasi *public key*, karena *authentication authority* hanya berurusan dengan autentikasi *address* setiap *node* pada *blockchain*. Dengan tugas otoritas sentral yang lebih sedikit ini, dapat dikatakan bahwa sistem menjadi lebih terdesentralisasi dan mengurangi faktor *trust* yang terlibat di dalamnya secara signifikan.

Selain itu, distribusi kunci publik dengan *blockchain* ini juga lebih mudah dan sederhana bagi *public key issuer*. Pada sistem dengan *certificate authority*, setiap akan mengisukan kunci publik, *issuer* harus meminta *certificate authority* untuk menerbitkan sertifikat digitalnya dan memastikan sertifikat digital ini terdistribusi pada jaringan. Dengan sistem *blockchain*, hal tersebut tidak diperlukan lagi. *Node* yang mengisukan kunci publik cukup menambah *transaction* berisi *digital certificate*-nya ke *blockchain*, lalu melakukan *mining* pada *block* selanjutnya atau menunggu *node* lain melakukan *mining* pada *block* selanjutnya. Dengan begitu, otoritas sentral tidak lagi dibutuhkan untuk mendistribusikan kunci publik, melainkan otoritas sentral

di sini, *authentication authority*, cukup bertugas memverifikasi *address* setiap *node* yang ingin bergabung ke jaringan saja.

Ada satu lagi isu yang perlu dibahas, yakni *proof-of-work*. Sebelumnya telah dijelaskan mengapa mencari proof-*of-work* (*mining*) adalah operasi yang mahal secara komputasional namun diperlukan untuk meng-*confirm transaction* yang masuk ke *blockchain* dan menjaga keamanan dari *blockchain*. Karena *mining* adalah operasi yang mahal, harus ada insentif yang diberikan kepada pihak yang melakukannya. Pada Bitcoin atau Ethereum, insentif dari *mining* adalah *reward* berupa uang Bitcoin dan Ethereum, di mana sebuah *node* yang berhasil melakukan *mining* mendapatkan hadiah 1BTC atau 1ETH. Akan tetapi, pada *blockchain* untuk distribusi kunci publik ini tidak ada mata uang seperti demikian, sehingga solusi dari isu ini tidak *obvious*.

Usulan penulis adalah mengaplikasikan sistem kuota, yakni batasan kunci publik yang dapat di-*publish* oleh suatu *node*. Apabila suatu *node* ingin menambah kuotanya, *node* tersebut perlu melakukan *mining* agar kuotanya bertambah. Selain memberikan insentif bagi *node* untuk melakukan *mining*, sistem ini juga mengurangi *spam transaction* pada *blockchain*. Masalah yang muncul pada sistem insentif ini adalah, tidak semua *public key issuer* mampu memiliki *hardware* dan kelengkapan lain yang mumpuni untuk melakukan *mining*, karena *mining* membutuhkan komputasi yang besar. Kemungkinan yang terjadi adalah jumlah pihak yang melakukan *mining* menjadi terbatas. Maka dari itu, perlu diaplikasikan mekanisme bagi setiap pihak agar dapat membeli kuota untuk mengisukan kunci publik ke pihak yang melakukan *mining*.

Meskipun begitu, *proof-of-work* tetap memiliki kelemahan lain, meskipun tidak berkaitan dengan segi keamanan. Karena membutuhkan komputasi yang besar untuk *mining*, maka dibutuhkan juga biaya besar seperti misalnya untuk listrik. Hal ini dapat berimplikasi buruk terutama pada lingkungan, karena banyak listrik yang menjadi terbuang hanya untuk melakukan *mining*. Faktor ini merupakan salah satu faktor utama mengapa teknologi-teknologi *blockchain* seperti Ethereum mulai berpaling ke alternatif-alternatif *proof-of-work*.

Alternatif lain adalah menggunakan konsensus yang berbeda dari *proof-of-work*, seperti *proof-of-stake* atau *proof-of-burn*. Akan tetapi, konsensus selain *proof-of-work* sebagian besar masih sebatas konsep dan belum teruji aplikasinya di dunia nyata. Oleh karena itulah, penulis hanya menjelaskan konsensus *proof-of-work* pada makalah ini. Di masa depan, mungkin akan ada konsensus *blockchain* yang lebih baik untuk implementasi *blockchain* untuk distribusi kunci publik ini.

Berdasarkan analisis-analisis tersebut, diperlukan dua hal penting agar sistem *blockchain* untuk distribusi kunci publik ini dapat *feasible* untuk digunakan di dunia nyata. Yang pertama adalah dibutuhkannya *authentication authority* yang dapat dipercaya untuk memverifikasi *address* dari setiap *node* pada jaringan. Seharusnya lebih mudah mencari pihak yang dapat dipercayakan sebagai *authentication authority*, karena beban tugas dan aspek *trust* yang terlibat jauh lebih sedikit. Yang kedua adalah diperlukannya sistem insentif yang tepat, sehingga ada pihak-pihak yang bersedia melakukan *mining* demi meng-*confirm transactions* dan menjaga keamanan *blockchain*.

Apabila kedua aspek tersebut tertangani dengan baik, seharusnya *blockchain* untuk distribusi kunci publik ini *feasible* digunakan di dunia nyata. Akan tetapi, pada kenyataannya *certificate authority* sudah seperti *status quo* dalam *public key infrastructure*. Hal ini dikarenakan sistem berbasis *certificate authority* ini telah berjalan baik dalam waktu yang lama dan belum ada insentif besar bagi pihak-pihak yang terlibat dalam Internet untuk mengubah sistem ini.

V. KESIMPULAN

Setelah melakukan eksperimen dan analisis, dapat disimpulkan bahwa menggunakan *blockchain* untuk distribusi kunci publik dapat menjadi alternatif dari *public key infrastructure* sekarang yang sangat bergantung pada otoritas sentral *certificate authority*. C*ertificate authority* ini bertujuan memecahkan masalah *authentication* dan *integrity* dari *public key* yang dikirim. Dengan *blockchain*, masalah-masalah tersebut juga dapat terpecahkan.

Properti utama *blockchain* adalah sifatnya yang append-*only*, yaitu data hanya dapat ditambahkan, sedangkan data yang telah ditambahkan tidak dapat diubah maupun dihapus oleh penyerang atau pihak yang tak bertanggung jawab. Sifat ini menjamin *integrity* dari *public key* yang didistribusi melalui *blockchain*.

Pada implementasi *blockchain* yang sudah ada, seperti Bitcoin dan Ethereum, masalah *authentication* dipecahkan menggunakan mekanisme *digital signature*. Akan tetapi, untuk *digital signature* ini tetap dibutuhkan penyebaran kunci publik menggunakan *certificate authority* (*CA*). Karena tujuan utama dari makalah ini adalah menghilangkan kebergantungan pada *CA*, harus dicari solusi lain yang lebih terdesentralisasi. Sayangnya, penulis tidak menemukan solusi yang murni terdesentralisasi. Usulan penulis adalah menggunakan *authentication authority*, yaitu pihak ketiga yang memverifikasi *authentication* dari setiap *node* pada *blockchain*. Walaupun sistem *blockchain* menjadi tidak murni terdesentralisasi, pada bab Analisis telah dijelaskan kelebihan-kelebihan *authentication authority* yang signifikan ketimbang menggunakan *CA*.

Karena *blockchain* menggunakan sistem *proof-of-work*, diperlukan insentif agar *node* melakukan *mining* untuk meng-*confirm transaction* dan menjaga keamanan *blockchain*. Pada bagian analisis, penulis mengusulkan diaplikasikannya sistem kuota sebagai *reward* untuk *node* yang melakukan *mining*. Alternatif lain adalah menggunakan konsensus *blockchain* lain seperti *proof-of-stake*, *proof-of-burn*, dan lainnya. Namun, sebagian besar konsensus tersebut masih sebatas teori dan belum teruji

aplikasinya di dunia nyata. Di masa depan, mungkin akan ada konsensus yang lebih tepat untuk implementasi *blockchain* untuk distribusi kunci publik ini.

Apabila masalah *authentication authority* dan insentif *proof-of-work* tertangani dengan baik, maka penulis menyimpulkan sistem *blockchain* ini *feasible* digunakan di dunia nyata. Meskipun begitu, tantangan tetap terletak pada posisi *certificate authority* yang sudah menjadi *status quo* dalam *public key infrastructure*.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Pak Dr. Ir. Rinaldi Munir, MT. sebagai dosen mata kuliah IF4020 Kriptografi. Penulis juga mengucapkan terima kasih kepada seluruh penulis yang tulisannya dijadikan referensi untuk makalah ini. Terakhir, penulis berterima kasih kepada orang tua dan teman-teman atas dukungannya dalam menyelesaikan makalah ini.

Referensi

[1]. Munir, Rinaldi. 2005. *Diktat Kuliah Kriptografi*. Bandung: Program Studi Teknik Informatika Institut Teknologi Bandung.
[2]. Nakamoto, Satoshi. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*.
[3]. https://en.bitcoin.it/wiki/ terakhir diakses pada 17 Mei 2018.
[4]. https://hackernoon.com/learn-blockchains-by-building-one-117428612f46 terakhir diakses pada 17 Mei 2018.
[5]. https://github.com/devinalvaro/chain terakhir diakses pada 17 Mei 2018.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2018

Devin Alvaro