Studi Penggunaan SHA Pada Identifikasi E-KTP Indonesia

Archie Anugrah / 13508001
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18001@students.if.itb.ac.id

***Abstrak*—Pada makalah ini dijabarkan hasil studi mengenai kemungkinan penerapan SHA-1 untuk menjadi identifikator (nomor KTP) menggantikan format nomor KTP saat ini. Ada banyak hal yang ditemukan sepanjang studi, tetapi dengan dukungan infrastruktur yang tepat, hal tersebut mungkin untuk dilakukan dan akan memudahkan pengembangan sistem lebih lanjut nantinya.**

***Kata Kunci*—SHA-1, E-KTP, Nomor KTP.**

I. PENDAHULUAN

A. Pendahuluan

Akhir-akhir ini, isu mengenai digitalisasi Kartu Tanda Penduduk sedang marak seiring dengan proyek E-KTP yang sedang diusung pemerintah. Proyek E-KTP sendiri bukan hanya bertujuan untuk mendigitalisasi data-data digital belaka, tetapi juga bertujuan untuk menyatukan sistem pendataan berbagai daerah, memudahkan pencarian data orang, memudahkan birokrasi, dan banyak manfaat baik lainnya.

Walaupun proyek ini sudah mencapai tahap pengujicobaan di beberapa wilayah, ternyata proyek ini masih memiliki banyak kendala, diantaranya adalah sulitnya menyatukan sistem-sistem kerja di daerah-daerah, selain problema-problema lainnya.

Pada makalah ini, penulis akan menyorot satu masalah sederhana, yaitu mengenai penentuan nomor KTP dan segala implikasi yang disebabkannya. Sementara ini, nomor KTP memiliki suatu format kode tertentu yang dibubuhkan nomor urut diujungnya. Pada pengembangan sistem digital nantinya, saat *field* yang disediakan sudah habis, akan terjadi permasalahan integrasi no urut baru dan penetapan standard baru. Oleh karena itu, penulis menyarankan penggunaan algoritma hash dalam penomoran nomor KTP karena selain memberikan keunikan, hash juga memiliki standard baku tertentu yang sudah diakui secara internasional.

B. Kartu Tanda Penduduk

Kartu Tanda Penduduk adalah kartu pengenal yg harus dimiliki setiap orang (warga negara) yg memuat nama, nomor, jenis kelamin, umur dan tempat lahir, pekerjaan, dan alamat yg jelas [definisi Kamus Besar Bahasa Indonesia]. Inti dari Kartu Tanda Penduduk ini sendiri ada pada fungsi pengenal yang dimilikinya. Hal itu berarti setiap orang harus dapat diacu dengan unik, dan hal itu dicapai dengan mendapatkan beberapa data.

Selain properti keunikan, "Kartu ini wajib dimiliki bagi Warga Negara Indonesia (WNI) dan Warga Negara Asing (WNA) yang memiliki Izin Tinggal Tetap (ITAP) yang sudah berumur 17 tahun atau sudah pernah kawin atau telah kawin. Anak dari orang tua WNA yang memiliki ITAP dan sudah berumur 17 tahun juga wajib memilki KTP" [Wikipedia]. Hal ini mengartikan bahwa, tidak semua manusia yang tinggal di suatu negara memiliki Kartu Tanda Penduduk. Hanya orang-orang yang dianggap sah saja yang memilikinya, dan mungkin itulah yang menjadi pertimbangan dalam menggunakan nomor urut sebagai penambah *field* akhir di nomor Kartu Tanda Penduduk Indonesia.

Gambar 1.Kartu Tanda Penduduk Indonesia

C. Hash

Hash adalah fungsi matematis yang memetakan suatu data ke representasi lain yang unik. Representasi ini biasanya dinyatakan sebagai angka ataupun *array of byte*. Ada banyak macam hash. Hash yang paling umum digunakan adalah *hash* dari keluarga SHA. Dari keluarga SHA sendiri, *hash* yang paling populer adalah SHA-1. *Hash* ini populer karena kemampuan proses SHA-1 yang cukup cepat, penggunaan algoritma yang tidak terlalu kompleks, dan *library-library* yang rata-rata sudah mendukung fungsi ini.

Fungsi *hash* sangat penting peranannya karena fungsi hash dapat memberikan standard representasi yang unik. Kekuatan hash terletak pada standard keluaran dan algoritma.

II. STUDI

A. Field yang dibutuhkan

Untuk memilih *field* yang dibutuhkan, ada baiknya kita melihat field-field yang ada pada Kartu Tanda Penduduk Indonesia. Field yang tercantum adalah Nomor Induk Kependudukan (N.I.K.), nama lengkap, tempat dan tanggal lahir, jenis kelamin, agama, status perkawinan, golongan darah, alamat, pekerjaan, kewarganegaraan, foto, masa berlaku, tempat dan tanggal dikeluarkan Kartu Tanda Penduduk, tandatangan pemegang Kartu Tanda Penduduk, serta nama dan nomor induk pegawai pejabat yang menandatanganinya. Dari semua field ini, tidak semua *field* harus kita proses, yang kita butuhkan hanyalah field-field yang menjamin kekhasan saja. Field-field yang dibutuhkan kiranya hanyalah **Nama Lengkap, tempat dan tanggal lahir, jenis kelamin, golongan darah**. Tempat Kartu Tanda Penduduk tidak dimasukkan karena, data ini akan membuat nomor menjadi dependen terhadap tempat pengurusan, bukan terhadap individu lagi. Sedangkan alamat, agama, foto dan tanda tangan sendiri mungkin diganti pada perpanjangan berikutnya yang akan membuat hash dapat sering berubah-ubah setiap periode, sehingga *field-field* ini tidak dipilih.

Dari *field-field* yang telah ditentukan, mari kita pikirkan lagi, apakah *field-field* tersebut benar cukup untuk merepresentasikan seorang individu dengan unik ? Mari kita bayangkan, apakah mungkin ada seseorang yang memiliki nama lengkap, tempat tanggal lahir, jenis kelamin, dan golongan darah yang sama ? Jawabannya mungkin. Misalkan ada seseorang yang melahirkan anak kembar identik dan keduanya diberikan nama Andi, tentu saja semua isi dari *field* tersebut bernilai sama (kecuali tanggal lahir dibuat dalam presisi yang lebih tinggi). Berarti dari sisi pelabelan, kita membutuhkan suatu atribut lain untuk memastikan bahwa hal seperti itu tidak terjadi. Tetapi untuk sementara ini, atribut semacam itu masih belum ada, sehingga kemungkinan untuk terjadinya celah sistem ini masih ada dan hanya bisa diperbaiki dengan mengubah data yang ada (misalnya kedetilan waktu lahir dibuat sampai hitungan sekon).

B. Algoritma Hash Yang Digunakan

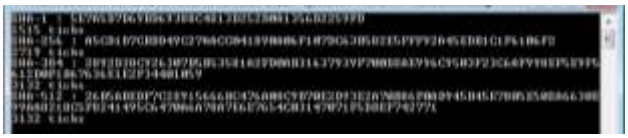
Sebagaimana judul makalah ini, hash yang akan digunakan merupakan hash dari keluarga SHA (*Secure Hash Algorithm*), hanya saja pertanyaannya adalah hash versi berapa ? Ada banyak pilihan keluarga SHA, yaitu SHA-0, SHA-1, SHA-256/224, SHA-512/384.

Dari masalah popularitas, saat ini fungsi SHA yang paling umum digunakan adalah SHA-1. Fungsi ini dianggap cukup cepat dan reliable untuk kebutuhan saat ini.

Dari masalah panjang bit sendiri, keluarga SHA-0 dan SHA-1 adalah fungsi hash dengan keluaran yang paling pendek dibandingkan yang lain, yaitu sepanjang 20 byte atau dengan kata lain sebanyak 2^{20} kombinasi. 20 byte ini, kalau ditransformasikan dalam heksa desimal menjadi 40 karakter yang sudah cukup panjang untuk ditaruh di nomor Kartu Tanda Penduduk. Sehingga, dari sisi panjang hasil *hash*, lebih baik memilih fungsi *hash* SHA-0 atau SHA-1.

Dari masalah tingkat keamanan, fungsi *hash* yang paling aman adalah SHA-256/224 dan SHA-512/384. Kedua tipe SHA ini masih belum dapat ditemukan kolisinya, berbeda dengan para pendahulunya. Semakin baru algoritma *hash* tersebut, semakin aman pula keluarannya.

Dari semua hal tersebut, penulis menyimpulkan bahwa SHA yang paling tepat untuk digunakan adalah SHA-1. Hal ini disebabkan karena, walaupun sudah ada kemungkinan terjadinya kolisi, tetapi hal itu kecil sekali kemungkinannya (bahkan pembuktiannya sendiri masih berupa serangan hipotesis). Selain itu, yang paling penting fungsi ini secara umum memiliki performansi yang paling baik dan panjang hasil hash yang paling masuk akal untuk kebutuhan.



Gambar 2.Hasil uji coba SHA

C. Teknik pembentukan Hash

Dalam pembentukan hash, digunakan *string* tunggal yang berisi semua data *field-field* yang digabungkan. Setelah string ini terbentuk, data di-hash menggunakan SHA-1 dan dikonversi menjadi string representasi heksadesimalnya. Hal ini memiliki impact yang kurang baik karena hasil *hash* akan memiliki huruf A-F. Hal ini dapat diperbaiki dengan menggunakan representasi selain heksa, contohnya ASCII (*American Standard Code for Information Interchange*). Hanya saja ada masalah lain yang muncul jika menggunakan representasi seperti itu, yaitu dimungkinkannya kemunculan karakter-karakter aneh ataupun karakter yang tidak dapat dicetak di layar. Oleh sebab itu, representasi dalam heksadesimal merupakan representasi yang paling *feasible* untuk saat ini.

D. Pengujicobaan performansi perangkat lunak

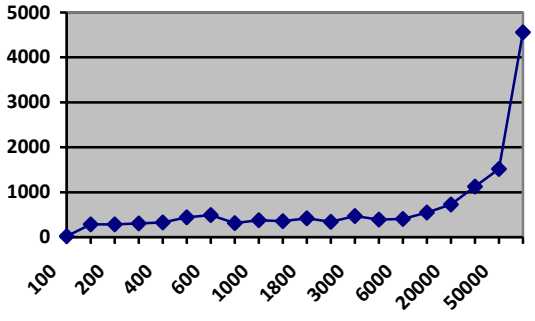
Dalam pengujicobaan ini, digunakan bahasa C# yang didukung oleh *.Net Framework* 4.0. Untuk pemanfaatan algoritma SHA-1 sendiri, digunakan algoritma yang dicantumkan pada http://dotnetpulse.blogspot.com/2007/12/sha1-hash-calculation-in-c.html.

```
public static string CalculateSHA1(string
text)
        {
            byte[] buffer =
Encoding.Default.GetBytes(text);
            SHA1CryptoServiceProvider
cryptoTransformSHA1 =  new
SHA1CryptoServiceProvider();
            string hash =
BitConverter.ToString(

cryptoTransformSHA1.ComputeHash(buffer)).Repl
ace("-", "");

            return hash;
        }
```

Gambar 3. Testing Pembentukan Hash

Setelah yakin dengan pembentukan *hash*, kita lakukan pengujian seperti apakah kinerja fungsi ini saat harus memroses banyak data. Pengujian ini menggunakan basis data MySQL. Pengujian ini bertujuan untuk melihat lama waktu pemrosesan data besar. Untuk pengujian digunakan data dummy terlebih dahulu. Pengujian ini dilakukan di komputer dengan *Chipset Intel Core2Duo* dengan kecepatan 2GHz. Berikut hasil dari eksperimen yang didapat oleh penulis

- Jumlah Data : 100
  Lama Eksekusi : 285
- Jumlah Data :200
  Lama Eksekusi :297
- Jumlah Data : 300
  Lama Eksekusi : 306
- Jumlah Data : 400
  Lama Eksekusi :328
- Jumlah Data :500
  Lama Eksekusi :441
- Jumlah Data :600
  Lama Eksekusi :490
- Jumlah Data :800
  Lama Eksekusi :313
- Jumlah Data :1000
  Lama Eksekusi :380
- Jumlah Data :1200
  Lama Eksekusi :359
- Jumlah Data :1800
  Lama Eksekusi :417
- Jumlah Data :2400
  Lama Eksekusi :340
- Jumlah Data :3000
  Lama Eksekusi :469
- Jumlah Data :4000
  Lama Eksekusi :394
- Jumlah Data :6000
  Lama Eksekusi :405
- Jumlah Data :10000
  Lama Eksekusi : 551
- Jumlah Data :20000
  Lama Eksekusi :724
- Jumlah Data :30000
  Lama Eksekusi :1128
- Jumlah Data :50000
  Lama Eksekusi :1517
- Jumlah Data :200000
  Lama Eksekusi :4557



Bagan 1Kurva lama eksekusi.

Dari pengujian, didapatkan kesimpulan bahwa pada kisaran data ribuan unit, lama pemrosesan data cenderung stabil. Setelah masuk ke kisaran puluhan ribu, lama pemrosesan mulai meningkat secara eksponensial.

Dari kurva bisa kita lihat bahwa pada jumlah data 200000 entri, pemrosesan dilakukan sekitar 4,5 detik. Jadi, kira-kira pada awal keberjalanan, untuk memroses 200 juta penduduk Indonesia, kira-kira dibutuhkan waktu 4,5 X 1000 = 4500 detik = 75 menit. Angka ini adalah prediksi waktu yang cukup realistis untuk penghitungan data negara. Mengapa realistis ? Hal ini disebabkan nantinya pengolahan data akan dilakukan oleh server dengan performansi yang baik, bukan sekedar komputer desktop seperti kondisi pengujian saat ini.

Mari kita spekulasikan penggunaannya.Tentu saja pada mode operasional, program tidak akan memroses 200 juta (kisaran kasar jumlah penduduk Indonesia) entri setiap waktu. Asumsikan setiap harinya ada 1% penduduk Indonesia yang membutuhkan pemrosesan nomor kartu tanda penduduk, atau sebesar 2 juta orang orang. Waktu yang dibutuhkan untuk pemrosesan 200 ribu orang dari hasil pengujian sebelumnya adalah 4,5 detik. Berarti kita perkirakan bahwa untuk memroses 2 juta data kita membutuhkan 45 detik pada komputer rumahan. Dalam satu hari, kira-kira sistem hanya membutuhkan waktu sekitar 45 detik saja untuk memroses seluruh data yang dientrikan pada hari itu menandakan bahwa entri data ini tidak memerlukan resource yang terlalu besar pada server

dan *feasible* untuk diterapkan.

E. Efek dari penggunaan hash

Pada format kartu tanda penduduk saat ini, dari angka yang tertera pada nomor kartu tanda penduduk kita bisa mendapatkan informasi mengenai asal kecamatan, tanggal lahir, dan nomor registrasi. Hal ini dilakukan supaya nomor kartu tanda penduduk setiap orang dapat dibuat secara unik, sehingga penggunaan hash tidak akan mengubah kegunaan dari nomor kartu tanda penduduk itu sendiri.

Penggunaan hash dapat menghilangkan kandungan informasi tersebut. Selain mengenai hilangnya format, penggunaan hash dengan memergunakan sistem heksadesimal, kita juga terpaksa memergunakan abjad A-F pada nomor kartu tanda penduduk yang terkesan tidak natural.

Tetapi selain efek negatif, tentu saja *hash* juga membawa beberapa pengaruh positif. Pengaruh yang paling penting adalah memungkinkan pembuatan kode identifikator secara unik yang *scalable*. Maksud dari *scalable* disini adalah, jika suatu saat angka penampung nomor kartu tanda penduduk sudah tidak memadai, kita dapat mengganti fungsi *hash* ini dengan fungsi *hash* lain yang dapat menggenerasi rangkaian karakter yang lebih panjang. Efek positif lain yang tidak kalah pentingnya adalah, adanya kemungkinan penggunaan *hash* untuk dicantumkan dalam tanda tangan digital.

F. Studi mengenai pemanfaatan nomor KTP sebagai Hash

Ada satu aplikasi dari *hash* yang dirasa sangat penting dalam hal verifikasi dan validasi, hal itu adalah peletakan tanda tangan digital. Hal ini sangat penting berkaitan dengan aplikasi E-KTP ke depannya. Dengan menggunakan hash standard sebagai identifier, kita dapat menggunakan hash ini untuk menambahkan tanda tangan digital pada E-KTP. Tetapi untuk mengimplementasikan hal ini, kita harus menentukan terlebih dahulu, kunci apakah yang dapat digunakan untuk mengenkripsi tanda tangan digital ini.

Mungkin, untuk kartu tanda penduduk konvensional, tidak mungkin dilekatkan tanda tangan digital. Tetapi, untuk keberjalanan E-KTP nantinya, pemeriksaan keabsahan kartu sangat penting.

Untuk membuat tanda tangan digital ini, dibutuhkan 2 bilangan yang saling prima dan unik untuk dijadikan kunci privat dan kunci publik. Ide penulis adalah menggunakan fungsi enkripsi yang memiliki unsur simetris (kunci privat dan kunci publik dapat dipertukarkan, contohnya RSA). Setelah itu kita membutuhkan angka yang khas untuk masing-masing kartu yang kunci publiknya dapat disimpan didalam sebuah basis data terpusat, sedangkan kunci privatnya diletakkan didalam kartu tersebut. Dengan cara seperti ini, kita dapat memastikan bahwa kartu tersebut memang dimiliki oleh orang yang bersangkutan.

Tapi ada satu hal yang bermasalah disini, yaitu hash yang akan dikunci adalah nomor kartu tanda penduduk, yang dalam kasus ini pasti akan dicantumkan dalam kartu tanda penduduk, sehingga data mengenai hash yang akan dicantumkan menjadi terbuka. Hal ini akan memudahkan penjebolan pengamanan terhadap E-KTP ini dan kurang cocok digunakan. Untuk menyelesaikan masalah ini, penulis menawarkan sebuah solusi, dimana hash yang dipakai untuk menjamin keaslian kartu dibuat lagi dari tanda tangan digital. Byte array yang merepresentasikan citra tanda tangan lah yang di-hash lalu digunakan menjadi tanda tangan digital dengan menggunakan kunci privat dalam kartu tersebut. Hasil tanda tangan digital yang dicantumkan dapat dengan mudah diperiksa oleh sistem dengan menggunakan kunci publik

Dari solusi yang ditawarkan, ada implikasi dimana data dalam E-KTP tidak boleh diakses, karena begitu seseorang telah mendapatkan akses terhadap E-KTP, orang itu dapat menggunakan kunci privat dan citra tanda tangan yang tersimpan didalamnya untuk berperan sebagai pengguna. Hal ini dapat dicapa jiga keamanan verifikasi kartu awal terlaksana dengan baik.

Mengapa pemeriksaan ini penting ? Karena nantinya diharapkan E-KTP dapat terintegrasi dengan segala aspek keihdupan kita, seperti data kepemilikan bahkan sampai ATM. Sekarang apakah pemeriksaan ini dapat dilakukan dengan keadaan sekarang ini ? Mari kita lakukan pengujian lebih lanjut mengenai hal tersebut.

Pertama-tama kita akan menguji performansi pemeriksaan / verifikasi terhadap E-KTP. Pengujian ini digunakan menggunakan C# juga, sama seperti pada bagian sebelumnya. Untuk kodenya sendiri, penulis menggunakan enkripsi RSA hasil buatan sendiri.

Dari hasil pengujian didapatkan bahwa lama pemrosesan 1 data baik tanda tangan maupun verifikasi rata-rata adalah sekitar 355 ms untuk komputer Core 2 Duo 2 Ghz. Waktu proses ini penulis rasa cukup masuk akal untuk waktu proses satu buah kartu.

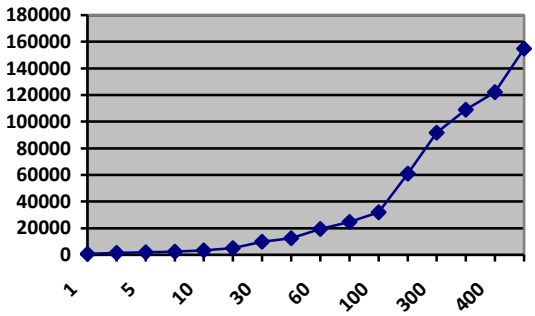
Pada paragraf sebelumnya, sudah diuji mengenai kinerja pengecekan satu buah kartu. Tetapi, sebagaimana implikasi yang ditulis sebelumnya, bahwa *hash* dari citra tanda tangan dapat digunakan untuk melegalisasi dokumen-dokumen digital lainnya yang berhubungan dengan individu tertentu. Untuk memroses hal ini, maka akan dibutuhkan sebuah server khusus yang menyimpan data semua penduduk (server pemerintah). Setiap ada request untuk legalisasi, server akan memroses dan mengembalikan pernyataan apakah permintaan tersebut valid atau tidak. Pada paragraf ini akan diuji kinerja bayangan dari server tersebut, dan dianalisis kemungkinan aplikasinya.

Pada pengujian ini, kembali digunakan bahasa pemrograman C# dengan bantuan basis data MYSQL. Pengujian ini dilakukan di komputer dengan Chipset Intel Core2Duo dengan kecepatan 2GHz.

- Jumlah Data : 1

Lama Eksekusi : 713

- Jumlah Data :3
  Lama Eksekusi :1329
- Jumlah Data : 5
  Lama Eksekusi : 1835
- Jumlah Data : 7
  Lama Eksekusi :2423
- Jumlah Data :10
  Lama Eksekusi :3285
- Jumlah Data :15
  Lama Eksekusi :5008
- Jumlah Data :30
  Lama Eksekusi :9952
- Jumlah Data :40
  Lama Eksekusi :12392
- Jumlah Data :60
  Lama Eksekusi :19324
- Jumlah Data :80
  Lama Eksekusi :24760
- Jumlah Data :100
  Lama Eksekusi :31808
- Jumlah Data :200
  Lama Eksekusi :60874
- Jumlah Data :300
  Lama Eksekusi :91679
- Jumlah Data :350
  Lama Eksekusi :108903
- Jumlah Data :400
  Lama Eksekusi :122196
- Jumlah Data :500
  Lama Eksekusi :154716



Bagan 2.Kurva lama eksekusi

Dapat kita lihat dari bagan bahwa,pada pemrosesan 400 data saja, waktu yang dibutuhkan sudah 150 detik. Sekarang kita lakukan kalkulasi sederhana. Anggap sekitar 1 % dari penduduk Indonesia melakukan transaksi pada waktu yang sama, berarti ada 2 juta data yang harus diproses. Berarti, untuk komputer standard, waktu yang dibutuhkan untuk memroses data dalam sehari kira-kira 155X(2juta/500) = 620000 detik = 172 jam. Dapat kita lihat bahwa kebutuhan resource untuk komputasi tanda tangan digital membutuhkan kekuatan komputasi yang besar. Tetapi, testing dilakukan di lingkungan desktop standard. Jika kita menggunakan server yang memiliki perangkat keras yang baik, tentu saja waktu pemrosesan tidak akan selambat itu.

G. Penanganan kesalahan

Penggunaan *hash* sebagai tanda identifikasi memberikan berbagai permasalahan baru yang harus ditangani. Permasalahan pertama adalah masalah kolisi. SHA-1 memiliki kemungkinan untuk kolisi, walaupun masih dalam tahap hipotesis. Jika terjadi kolisi (data hash kedua data sama, walaupun data mereka berbeda), maka ada beberapa opsi yang bisa digunakan

1. Mengganti isi data.
   Hal ini tidak disarankan karena hal ini dapat merusak konsistensi data.
2. Memberikan padding pada ujung data
   Pada solusi ini, kita bisa memberikan sebuah spasi kosong pada ujung data tempat tanggal lahir. Spasi sendiri digunakan karena spasi tidak tampak secara visual. Tetapi mengapa harus pada field tempat dan tanggal lahir ? Hal ini dilakukan untuk menanggulangi kemungkinan verifikasi nama. Anggap saja pada suatu hari pengguna harus memasukkan nama untuk dicocokkan dengan isi kartu. Walaupun nama yang dimasukkan sudah tepat, fungsi *equal* pada string akan memberikan hasil yang selalu false karena adanya spasi di ujung data yang tidak terlihat oleh mata. Sedangkan, jika kita menggunakan format tempat dan tanggal lahir kartu tanda penduduk saat ini, maka ketika kita memeriksa tempat lahir, kita harus melakukan *split* string yang dipisahkan oleh spasi dan penambahan spasi ini tidak berpengaruh pada manipulasi data.

Masalah lain yang ditimbulkan adalah kebutuhan akan *server* publik yang dapat diandalkan untuk melakukan validasi E-KTP. Server ini harus cukup andal untuk diakses dari seluruh Indonesia pada waktu yang sama. Sebenarnya fungsiyang diberikan *server* ini sederhana, hanya menaruh list kunci dari masing-masing E-KTP, tetapi jika *server* ini down, fungsi-fungsi administratif yang dimiliki E-KTP dapat menjadi bermasalah. Server ini pun harus dapat melakukan manipulasi sertifikasi digital yang cukup berat, sehingga *server* harus sangat andal.

Masalah lain yang mungkin muncul adalah kemungkinan hilangnya data kartu. Hal ini dapat ditangani dengan menyimpan data-data tersebut dalam *server* negara yang tentunya harus sangat aman, atau peretas dapat menggunakan segala akses yang dimiliki seseorang (kunci privat dan data).

III. KESIMPULAN

Pemanfaatan *hash* sebagai alternatif nomor kartu tanda

penduduk dapat menjadi suatu solusi dalam memberikan sebuah identifier yang *scalable* terhadap sistem. Tentu saja penggunaan representasi seperti ini memiliki beberapa efek negatif pula. Tetapi, dengan pengembangan lebih lanjut, penggunaan hash sebagai pengganti identifikator pada kartu tanda penduduk dapat menjadi alternatif besar bagi pengacuan individu pada basis data kependudukan Indonesia.

REFERENCES

[1] Munir, Rinaldi. 2005. Diktat Kuliah IF5054 Kriptografi.Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] http://id.wikipedia.org/wiki/Kartu_Tanda_Penduduk (diakses tanggal 28 April 2011)

[3] http://en.wikipedia.org/wiki/Secure_Hash_Algorithm(diakses tanggal 28 April 2011)

[4] http://kamusbahasaindonesia.org(diakses tanggal 28 April 2011)

[5] http://dotnetpulse.blogspot.com/2007/12/sha1-hash-calculation-in-c.html(diakses tanggal 28 April 2011)

[6] http://en.wikipedia.org/wiki/RSA (diakses tanggal 28 April 2011)

[7] http://en.wikipedia.org/wiki/Digital_signature (diakses tanggal 28 April 2011)

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, 2 Mei 2011

Archie Anugrah
13508001