Pemeriksaan Integritas *File* Yang Di-*upload* ke *Cloud Storage dengan* Schnorr *Digital Signature*

Ivan Andrianto
Teknik Informatika / Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
andrianto.ivan@gmail.com

***Abstrak*—Penggunaan *cloud storage* terus meningkat dalam beberapa tahun terakhir. Dengan *cloud storage*, pengguna dapat menyimpan *file* yang dimilikinya pada *cloud storage* sehingga dapat diakses dengan mudah. Namun terdapat kemungkinan *file tersebut* termodifikasi baik karena kesalahan transmisi data maupun diubah pada saat disimpan di *cloud storage*. Salah satu cara untuk memeriksa integritas dari *file* tersebut adalah dengan menggunakan *digital signature*. Terdapat berbagai skema untuk membuat *digital signature*. Salah satunya adalah dengan menggunakan skema Schnorr.**

Keywords—Digital Signature; Elliptical Curve, Schnorr, cloud storage

I. PENDAHULUAN

Cloud storage merupakan suatu model dimana data disimpan pada beberapa *server remote* yang dapat diakses melalui Internet. Tren penggunaan *cloud storage* semakin meningkat dalam beberapa tahun terakhir. Hal itu disebabkan kemudahan yang ditawarkan *cloud storage* yaitu data dapat diakses dimana saja dan kapan saja melalui berbagai perangkat. *Cloud storage* biasanya dikelola dan dioperasikan oleh cloud storage service provider. Terdapat beberapa provider yang banyak digunakan antara lain Dropbox, Google Drive, OneDrive, dan Box. Pengguna diberi ruang penyimpanan yang cukup lega untuk menyimpan berbagai jenis *file digital*.

Namun terdapat beberapa masalah yang dihadapi saat menggunakan layanan cloud storage. Salah satunya adalah integritas file. Pada saat *file* di-*upload*, terdapat kemungkinan data yang di-*upload* secara tidak sempurna karena suatu hal. Selain itu, pada saat disimpan di *cloud storage*, *file* tersebut dapat berubah, khususnya apabila terdapat banyak orang yang dapat mengakses *file* tersebut. Oleh karena itu, diperlukan suatu cara agar dapat memeriksa integritas file untuk dapat memastikan bahwa isi dari file tidak berubah dengan file sebelum diupload

Untuk mengatasi masalah tersebut, salah satu cara yang dapat digunakan adalah dengan menerapkan *digital signature*. *Digital signature* perlu dibuat sebelum file di-*upload* dan akan diverifikasi setelah file di-*download* kembali. Dengan memanfaaatkan *digital* signature, dapat diketahui apakah *file* telah berubah dari *file* asli sebelum di-*upload* ke layanan *cloud storage*.

Terdapat berbagai skema yang dapat digunakan untuk membentuk *digital signature*. Dua skema yang banyak digunakan adalah ECDSA dan El Gamal. Namun selain kedua skema tersebut juga terdapat skema lain, salah satunya adalah Schnorr. Schnorr merupakan salah satu skema yang cukup handal untuk *digital signature*. Tetapi karena dipatenkan (U.S. Patent 4,995,082), tidak banyak yang menggunakan algoritma tersebut. Paten tersebut telah habis masa berlakunya pada Februari 2008 silam.

II. DASAR TEORI

A. *Digital Signature*

Digital Signature merupakan skema yang digunakan untuk memeriksa keaslian dari dokumen atau pesan digital. *Digital signature* dikembangkan berdasarkan kriptografi kunci publik, atau yang biasa disebut dengan *asymmetric cryptography*. Dengan menggunakan salah satu algoritma kunci publik, seseorang dapat membangkitkan *digital signature* untuk sebuah dokumen atau pesan digital.

Untuk membuat *digital signature*, dimanfaatkan fungsi *hash* satu arah dengan parameter data yang akan diberi *digital signature*. Hasil *hash* tersebut kemudian dienkripsi dengan *private key*. Enkripsi dilakukan terhadap hasil *hash*, bukan terhadap kesluruhan data. Hal itu dilakukan karena fungsi hash dapat menghasilkan output dengan panjang yang tetap. Dengan demikian, enkripsi yang dilakukan terhadap hasil dari *hash* akan lebih cepat dan biasanya memiliki panjang yang lebih pendek. Hasil dari hash sangat bergantung pada data yang di-*hash*. Apabila dilakukan perubahan sedikit terhadap salah satu karakter, hasil yang didapat akan jauh berbeda.

Untuk memeriksa integritas data yang telah diberi *digital signature*, perlu digunakan *public key* dari pihak yang memberikan *signature*. Apabila hasil hash yang diperoleh sama, kemungkinan besar data tersebut tidak berubah. Apabila tidak sama, terdapat dua kemungkinan yaitu data telah berubah atau public key yang digunakan untuk verifikasi tidak berkorespondensi dengan private key yang digunakan untuk membuat *digital signature*.

B. *Elliptical Curve Cryptography*

Elliptical Curve Cryptography adalah teknik untuk kriptografi kunci publik dengan berdasarkan teori *elliptic curve* untuk membentuk kunci secara lebih cepat dengan ukuran yang lebih kecil. ECC menghasilkan kunci berdasarkan properti dari persamaan *elliptic curve.*

Kelebihan dari ECC adalah dapat menghasilkan kunci yang tidak terlalu panjang namun secara keamanan tidak jauh berbeda dengan sistem lain yang menghasilkan kunci sangat panjang. Selain itu komputasinya juga efisien. Beberapa penerapan ECC antara lain untuk enkripsi, *digital signature*, dan *pseudo-random generators*.

C. Schnorr

Schnorr merupakan salah satu skema untuk *elliptic curve signature*. Meskipun algoritma Schnorr tidak sepopuler ECDSA dan ElGamal, sebenarnya algoritma tersebut mempunyai beberapa kelebihan.

Jika dibandingkan dengan ElGamal, Schnorr memiliki beberapa kelebihan yaitu:

- Tidak memerlukan inversi dari mod q, sehingga mengurangi waktu yang diperlukan untuk komputasi
- Schnorr menyertakan point random sebagai bagian dari input fungsi hash, bukan menghitung r = zB + H(m). Meskipun waktu yang diperlukan untuk verifikasi sedikit lebih lambat, tetapi dapat mengurangi space yang diperlukan untuk signature

Pada Schnorr *signature*, perlu digunakan sebuah *elliptical curve* dengan parameter yang telah ditenukan. Kemudian salah satu point g pada elliptical curve digunakan sebagai generator. Selain itu juga diperlukan fungsi hash H. Skema dari Schnorr Signature dapat dibagi menjadi tiga tahap yaitu pembangkitan kunci, signing, dan verifikasi.

1. Pembangkitan kunci
   - Pilih satu nilai random d sebagai private key
   - Public key yang dihasilkan adalah y = gd
2. Pembuatan *signature*
   - M adalah pesan yang akan di-sign
   - Buat suatu nilai random k
   - Hitung nilai r = gk
   - Hitung nilai e = H (M | r)
   - Hitung nilai s =(k - e * d)
   - Signature yang dihasilkan adalah (e, s)
3. Verifikasi
   - Hitung Rv = gs + ye
   - Hitung hasil hash ev = H (M | Rv)
   - Apabila ev = e, maka dapat berhasil diverifikasi

Berikut ini adalah bukti bahwa Rv = r

Rv = gs + ye = g (k - e * d) + gde = gk - gde + gde = gk = r

D. secp256k1

secp256k1 merupakan paramter yang digunakan pada *ellpitical curve* yang didefinisikan pada *Standards for Efficient Cryptography (SEC)* (Certicom Research, http://www.secg.org/sec2-v2.pdf).

secp256k1 didesain untuk membuat komputasi yang dilakukan lebih efisien, tidak seperti kebanyakan implementasi *elliptical curve* yang menggunakan struktur random. Apabila diimplementasi dengan baik, secp256k1 dapat menjadi lebih cepat sebesar lebih dari 30% dibanding kebanyakan kurva lain.

secp256k1 dapat didefinisikan sebagai tuple (*p*,*a*,*b*,*G*,*n*,*h*) dengan properti sebagai berikut

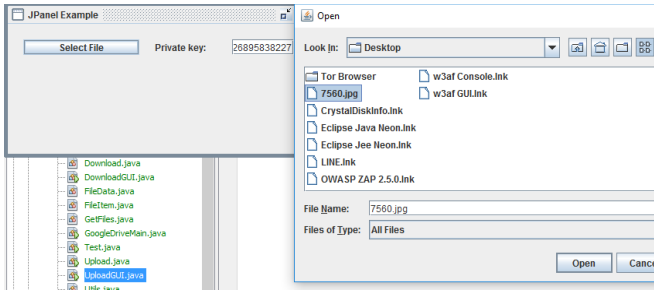
- *p* = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
- *a* = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- *b* = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007
- G (*base point*) = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
- *n* (*order*) = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
- *h* (*cofactor*) = 01

Karena a bernilai 0, maka persamaan kurva menjadi $E: y^2 = x^3+ax+b$

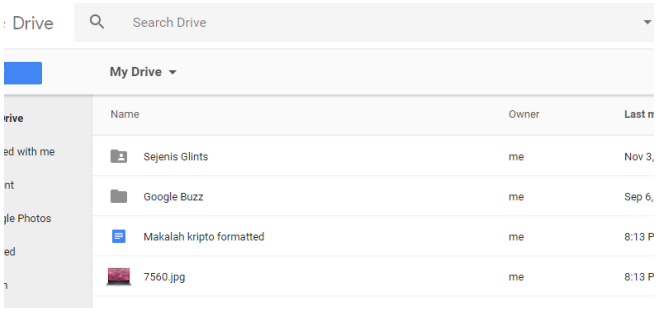
III. RANCANGAN APLIKASI

Aplikasi berikut dibuat khusus untuk Google Drive dengan memanfaatkan API yang disediakan oleh Google. Aplikasi menerapkan Schnorr *digital signature* dengan kurva secp256k1. Aplikasi dapat dibagi menjadi dua bagian yaitu untuk bagian menghasilkan *digital signature* pada saat file belum diupload dan untuk memverifikasi *file* yang telah di-*download* kembali.

Untuk membuat *digital signature*, pengguna dapat memasukkan *private key* pada *field* yang disediakan. Selanjutnya pengguna memilih salah satu *file* yang ada di komputer lokal. Sebelum *file* tersebut di-*upload*, dilakukan perhitungan terhadap *digital signature* menggunakan skema pembangkitan *signature* Schnorr. Pasangan *value digital signature* (e, s) yang dihasilkan akan disimpan di database lokal. Setelah itu, *file* diupload ke Google Drive



Gambar 1 - Tampilan program untuk upload file dan menambahkan digital signature



Gambar 2 -File yang telah teruplod di Google Drive

Untuk melakukan pengecekan terhadap integritas dari file yang telah di-*upload*, program akan menampilkan daftar file pengguna yang ada di Google Drive pengguna tersebut. Pengguna memilih salah satu file, lalu program akan men-*download file* tersebut. Berdasarkan *fileId* dari file tersebut, program akan mencocokkan apakah *digital signature* dari *file* tersebut disimpan di *database*. Apabila ditemukan, pengguna dapat melakukan validasi dengan memasukkan *public key* pada *field* yang disediakan (dipisahkan dengan koma). Verifikasi terhadap *digital signature* akan dilakukan saat pengguna mengklik tombol "Verifikasi".

IV. Eksperimen dan Pembahasan Hasil

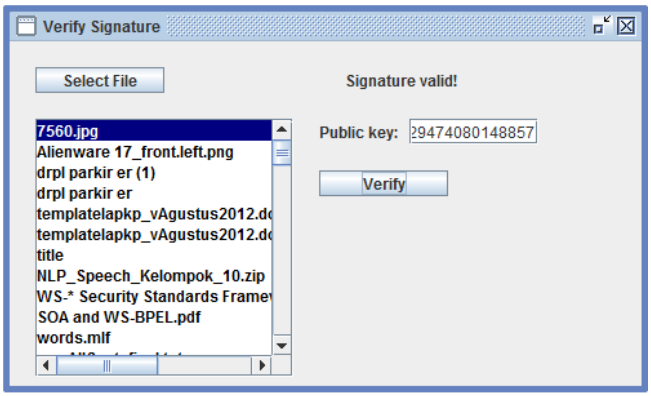
Pada eksperimen, *private key* yang digunakan adalah:
764603425862698961893421874406514494771637429672327919833869283426895838227

Berikut ini beberapa skenario yang dicoba oleh Penulis

1. *File* tidak diubah dan *public key valid*

e	760707883573374275164267632914887303411329615405208225318083831644453582339973
s	682615703179313141714323408958562430088018984913914101887211987328538135791 20
Public key x	967104318691033738355646441171531561112848645672448063528278274476583765437111
Public key y	288135644727778974208678680753491523030876916151385143849074281294740801488557

Apabila *file* tidak diubah dan *public key* yang digunakan *valid*, maka dapat disimpulkan integritas *file* tetap terjaga.

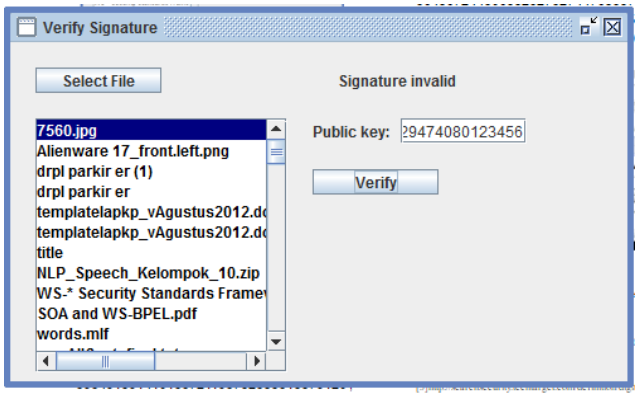


Gambar 3 - Hasil Verifikasi file tidak diubah dan public key valid

2. *File* tidak diubah, *public key* tidak *valid*

e	760707883573374275164267632914887303411329615405208225318083831644453582339973
s	682615703179313141714323408958562430088018984913914101887211987328538135791 20
Public key x	123456789091033738355646441171531561112848645672448063528278274476583765437111
Public key y	288135644727778974208678680753491523030876916151385143849074281294740801234 56

Apabila *file* tidak diubah tetapi *public key* yang digunakan tidak *valid*, maka verifikasi gagal.

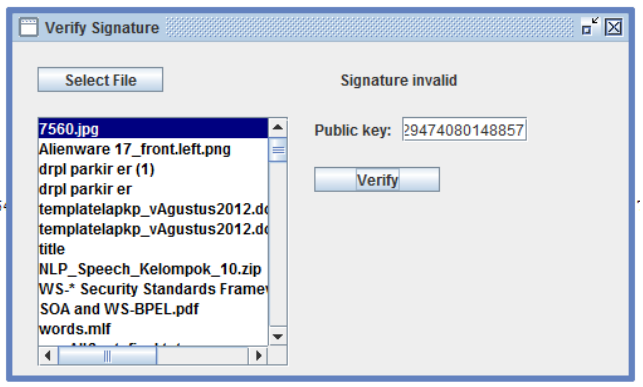


Gambar 4 - Hasil verifikasi file tidak diubah dan public key tidak valid

3. *File* diubah dan *public key valid*

e	760707883573374275164267632914887303411329615405208225318083831644453582339733
s	682615703179313141714323408958562430088018984913914101887211987328538135791200
Public key x	123456789091033738355646441171531561112848645672448063528278274476583765437111
Public key y	288135644727778974208678680753491523030876916151385143849074281294740801234561

Penulis dengan sengaja melakukan perubahan pada *file* gambar yang disimpan di Google Drive dengan mengeditnya menggunakan *online tool*. Apabila *file* berubah tetapi *public key* yang digunakan *valid*, verifikasi gagal. Hal ini sesuai dengan yang diharapkan, sehingga dapat digunakan untuk mengetahui apabila *file* telah mengalami modifikasi.

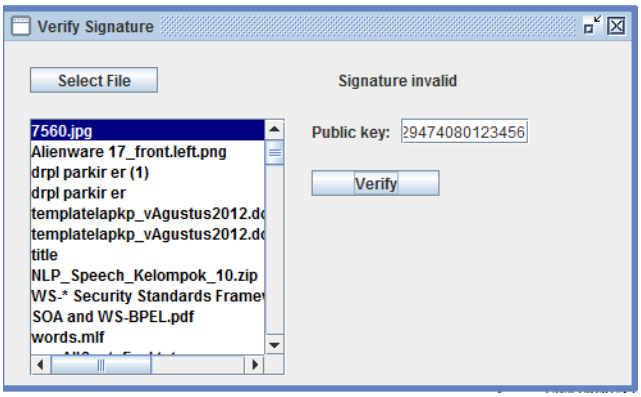


Gambar 5 - Hasil verifikasi file diubah dan public key valid

4. *File* diubah dan *public key* tidak *valid*

e	760707883573374275164267632914887303411329615405208225318083831644453582339733
s	682615703179313141714323408958562430088018984913914101887211987328538135791200
Public key x	123456789091033738355646441171531561112848645672448063528278274476583765437111
Public key y	288135644727778974208678680753491523030876916151385143849074281294740801234561

Apabila *file* diubah dan *public key* yang digunakan tidak *valid*, maka verifikasi gagal.



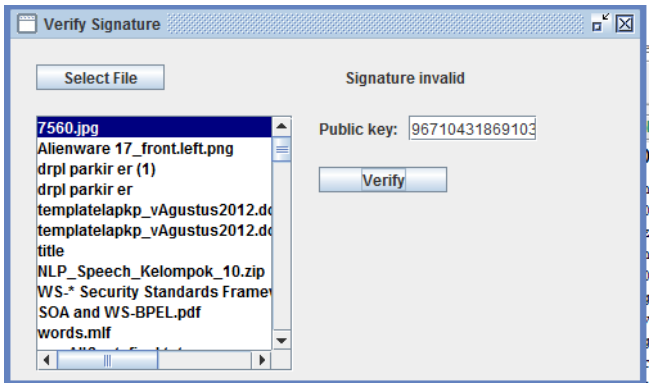
Gambar 6 - Hasil verifikasi file diubah dan public key tidak valid

5. *Digital Signature* Tidak *Valid*

e	13728491750174910328194267632914887303411329615405208225318083831644453582339733
s	13728491750174910328194267632914887303411329615405208225318083831644453582339733
Public key x	123456789091033738355646441171531561112848645672448063528278274476583765437111
Public key y	288135644727778974208678680753491523030876916151385143849074281294740801234561

Pada perobaan ini, Penulis dengan sengaja mengubah *digital signature* yang disimpan pada *database*. Dapat

dilihat bawah hasil yang didapat tidak valid apabila signature yang digunakan tidak valid.



Gambar 7 - Hasil verifikasi file apabila signature tidak valid

V. KESIMPULAN DAN SARAN

Schnorr dapat diterapakan sebagai skema untuk pembangkitan *digital signature*. Komputasinya cukup cepat dan *signature* yang dihasilkan tidak terlalu panjang. Performa juga semakin bertambah baik apabila dikombinasikan dengan parameter kurva yang tepat seperti secp256k1.

Integritas *file* merupakan hal yang penting bagi pengguna *cloud storage*. Oleh karena itu, diperlukan aplikasi *cloud storage client* yang dapat melakukan pembangkitan *signature* dan melakukan melakukan verifikasi integritas *file* berdasarkan *digital signature*. Fungsionalitas dari aplikasi yang dibuat oleh Penulis masih perlu ditambahkan seperti mendukung pengelompokan *file* berdasarkan *folder* dan mendukung layanan *cloud storage* selain Google Drive.

DAFTAR REFERENSI

[1]http://www.derkeiler.com/Newsgroups/sci.crypt/2006-08/msg01621.html. Diakses 17 Desember 2016.

[2]https://en.bitcoin.it/wiki/Secp256k1. Diakses 17 Desember 2016.

[3]http://searchsecurity.techtarget.com/definition/digital-signature. Diakses 17 Desember 2016.

[4]http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography. Diakses 17 Desember 2016.

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, 19 Desember 2016

Ivan Andrianto

NIM.13513039