Sistem Autentikasi Pengunggahan File dengan Algoritma ECDSA

Rakhmatullah Yoga Sutrisna
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
rakhmatullahyoga@gmail.com

***Abstract*—Penggunaan aplikasi berbasis web yang menyediakan fitur pengunggahan file saat ini sudah banyak sekali ditemukan di kalangan masyarakat. Namun tidak banyak aplikasi yang menjamin aspek keamanan pengunggahan file pada situs web aplikasi tersebut. Hal tersebut menimbulkan resiko bahwa file yang diunggah oleh pengguna aplikasi rentan terhadap serangan yang memanipulasi isi data. Makalah ini akan membahas satu alternatif autentikasi file dalam sebuah sistem yang melibatkan proses pengunggahan file sehingga pengguna bisa mendapatkan jaminan keaslian file yang diunggah.**

Kata kunci — pengunggahan file, tandatangan digital, ECDSA, autentikasi.

I. PENDAHULUAN

Pada era informasi saat ini pertukaran data melalui media digital sudah bukan menjadi hal yang asing lagi di kehidupan manusia. Pengguna komputer yang terhubung dalam sebuah jaringan komputer dapat saling berkomunikasi dan berbagi data dengan mudah tanpa harus mengeluarkan usaha atau biaya yang cukup memberatkan penggunanya. Melalui media internet, pengguna juga dapat melakukan penyimpanan berkas digital (*file*) pada *cloud storage*, sehingga pengguna tidak perlu lagi mengkhawatirkan mengenai kapasitas penyimpanan *file* yang terbatas pada komputer *client* mereka masing-masing.

Saat ini sudah tersedia banyak sekali media penyimpanan yang berbasis *cloud computing* yang dapat digunakan orang-orang di berbagai penjuru dunia. Media penyimpanan tersebut dapat diakses melalui web maupun aplikasi *client* yang melakukan sinkronisasi secara berkala terhadap *file* yang disimpan pada komputer *client* pengguna dengan *file* yang disimpan pada *server* media penyimpanan berbasis *cloud*. Pada pengaksesan dengan antarmuka web, *file* akan melalui mekanisme pengunggahan (*upload*) yang akan memgirimkan *file* tersebut melalui protokol internet. Selain *cloud storage*, terdapat banyak sekali aplikasi berbasis web yang salah satu proses bisnisnya melibatkan mekanisme pengunggahan *file* melalui protokol internet.

Sayangnya, protokol internet saat ini memiliki resiko pada sisi keamanan komunikasinya. Hal ini menimbulkan celah bagi pihak-pihak yang tidak berwenang atas *file* yang diunggah untuk dapat memanipulasi *file* tersebut. Selain itu penyerangan terhadap *file* yang diungah seringkali tidak diketahui oleh pengguna yang mengunggah *file*-nya Apabila *file* yang diunggah diserang oleh pihak yang tidak berwenang tersebut, bukan tidak mungkin pengguna akan kehilangan esensi data yang diunggah pada *cloud storage* atau aplikasi berbasis web tersebut.

Makalah ini akan membahas salah satu alternatif autentikasi dengan memanfaatkan *Elliptic Curve Digital Signature Algorithm* (ECDSA) yang dapat diterapkan dalam sebuah sistem pengunggahan *file*. Alternatif tersebut memang tidak meningkatkan aspek keamanan dalam mekanisme pengunggahan *file*, namun akan membuat sistem pengunggahan dapat mengetahui apakah *file* yang diunggah adalah *file* yang otentik berasal dari pengunggah atau file tersebut telah dimanipulasi oleh pihak-pihak yang tidak bertanggungjawab. Dengan adanya alternatif autentikasi pengunggahan *file* pada aplikasi berbasis web, diharapkan pengguna dapat melakukan proses transfer data digital tanpa khawatir *file* tersebut hilang atau termanipulasi.

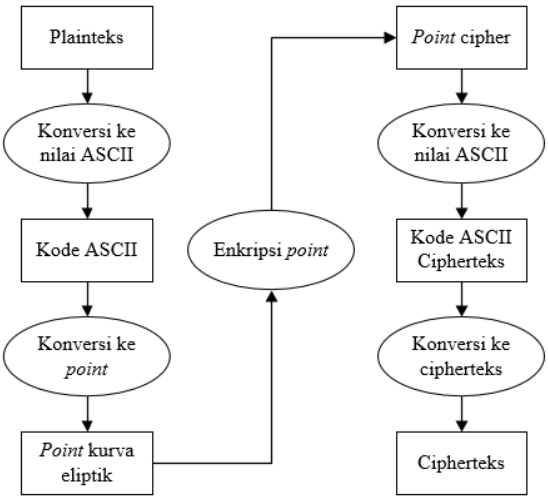
II. DASAR TEORI

A. Kriptografi Kurva Eliptik

Kriptografi Kurva Eliptik (ECC) adalah metode pada kriptogafi yang menggunakan *Eliptic Curve* pada tahap enkripsi dan dekripsinya. Algoritma ini dikembangkan oleh Neal Koblitz dan Victor S. Miller tahun 1985 [1]. ECC adalah salah satu algoritma kriptografi yang berbasis kunci publik. Keunggulan ECC dibandingkan dengan metode lainnya adalah ECC dapat memberikan tingkat keamanan yang sama dengan metode lainnya dengan ukuran bit yang lebih kecil.

Pada saat pembangkitan kunci, ECC membutuhkan parameter a,b,p,G,n yang disebut "*Domain Parameters*". G adalah pembangkit titik (*point*), dan n adalah orde dari G. Pihak A membangkitkan bilangan acak sehingga nA < n sebagai kunci privat dan menghitung kunci publik PA = G+G+G… +G nA kali. B membangkitkan bilangan acak nB < n sebagai kunci privat dan menghitung kunci publik PB = G+G+G…+nB kali.

Enkripsi dilakukan dengan cara mengirim 2 chiperteks point, yaitu {kG, Pm + k PB)} dan dekripsi dilakukan dengan cara Pm + kPB - nB(kG) = Pm + k(nB)G - nB(kG) = Pm. Enkripsi dan dekripsi pada ECC hanya dapat dilakukan pada Point, oleh karena itu pesan yang ingin dienkripsi haruslah di-*encode* menjadi *point* terlebih dahulu, dan pesan yang akan didekripsi haruslah di-*decode* dari *point*. Proses enkripsi, termasuk tahap *encode* dan *decode* teks dijelaskan pada Gambar 1 berikut.



Gambar 1. Proses enkripsi ECC

Ketika melakukan *encoding* menggunakan metode Koblitz, langkah-langkah yang perlu dilakukan adalah sebagai berikut.

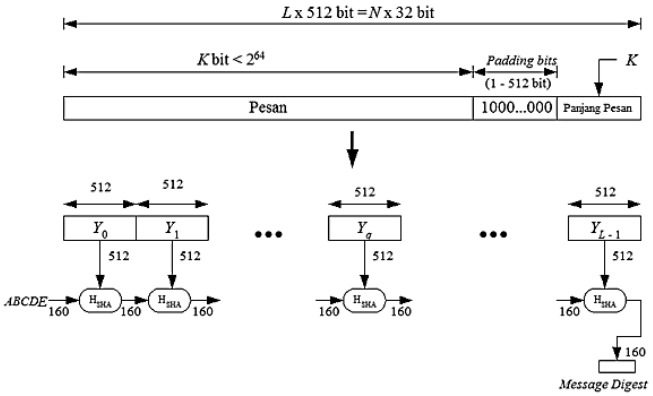
1) Pilih kurva eliptik Ep(a,b)
2) E memiliki *point* sejumlah N
3) Pesan terdiri atas angka 0-9 dan huruf A,B,C,…,X,Y,Z yang dikodekan sebagai 10, 11, …, 35.
4) Konversi pesan menjadi angka antara 0 – 35.
5) Pilih *auxiliary base parameter* k
6) Untuk setiap angka mk, ambil x=mk+1 dan selesaikan y
7) Sebuah nilai y akan ditemukan sebelum x=mk+k-1. Ambil point (x,y).

B. Fungsi *Hash* SHA1

Fungsi *hash* adalah fungsi yang menerima masukan berupa *string* dengan panjang sembarang, dan mengubahnya menjadi *string* yang panjangnya tetap [2]. Fungsi *hash* memiliki persamaan h=H(M) dimana h adalah hasil *hash*, dan M adalah pesan yang akan di-*hash*. Fungsi *hash* hanya bersifat 1 arah. Pesan *hash* tidak dapat dikembalikan lagi ke pesan semula.

Pada makalah ini, jenis *hash* yang dipakai adalah *Secure Hash Algorithm (SHA)*, yaitu SHA1. SHA dibuat oleh NIST dan digunakan bersama dengan DSS (*Digital Signature Standard*). SHA adalah pengembangan dari MD4 yang dibuat oleh Ronald L. Rivest dari MIT. Langkah untuk melakukan *hash* adalah seperti berikut.

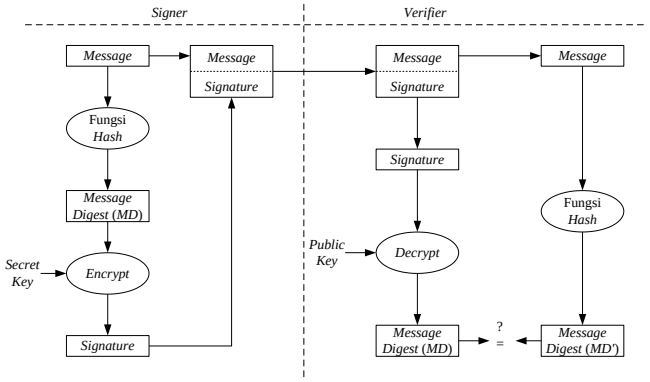
1) Penambahan bit-bit pengganjal (*padding bits*)
2) Penambahan nilai panjang pesan semula
3) Inisialisasi nilai MD
4) Pengolahan pesan dalam blok berukuran 512 bit.



Gambar 2. Skema pembuatan *message digest* dengan SHA-1

C. *Elliptic Curve Digital Signature Algorithm*

Elliptic Curve Digital Signature Algorithm (ECDSA) adalah analogi kurva eliptik dari algoritma tandatangan digital [2]. Algoritma ini merupakan pengembangan algoritma tandatangan digital secara umum yang menggunakan algoritma ECC dalam proses pembangkitan tandatangan digital serta verifikasinya. Alur umum tandatangan digital dapat dilihat pada Gambar 3 berikut.



Gambar 3. Alur umum algoritma tandatangan digital [3]

Pada algoritma ECDSA, langkah-langkah pemberian tandatangan digital dijabarkan sebagai berikut.

1) Pengirim menghitung nilai hash dari pesan M yang akan dikirim.
2) Pengirim melakukan enkripsi terhadap nilai hash h menggunakan kunci privat ECC
3) Pengirim mengirimkan pesan M + hasil enkripsi ECC dari hasil *hash* pesan ke penerima.

Setelah pengirim melakukan langkah-langkah pembangkitan tandatangan digital dan mengirim pesan tersebut ke penerima, langkah selanjutnya yang dilakukan oleh penerima pesan adalah memverifikasi tanda tangan. Berikut langkah-langkah verifikasi tandatangan digital.

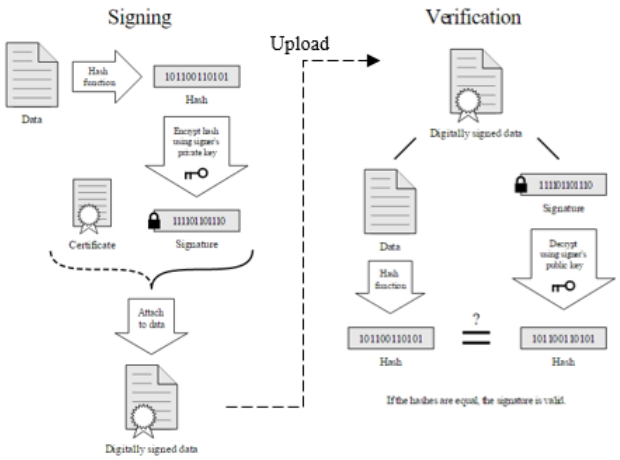
1) Penerima memisahkan antara pesan dan tanda tangan.
2) Penerima melakukan dekripsi tanda tangan menggunakan kunci public pengirim.

3) Penerima melakukan hash terhadap pesan yang diterima.
4) Hasil dari nomor 2 dan 3 dibandingkan. Apabila hasilnya sama, maka data yang dikirim valid.

III. IMPLEMENTASI SISTEM

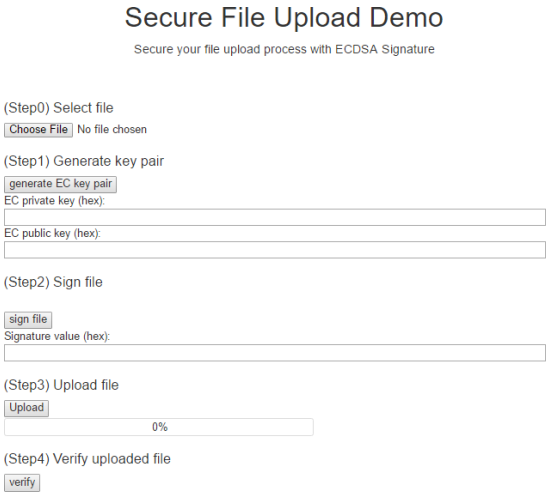
Sistem pengunggahan *file* pada umumnya hanya menerima data berupa *file* yang diunggah ke *server* aplikasi. Beberapa web pada sisi *client* melakukan proses enkripsi terhadap file sebelum diunggah ke sisi *server*. Hal tersebut memang akan mengamankan file tersebut agar tidak dapat dibaca oleh penyerang, namun penyerang situs web tetap dapat memanipulasi *file* yang sedang dikirimkan tersebut. Hal ini menyebabkan *server* akan menerima file tanpa melihat keaslian dari *file* yang diterima, sehingga apabila terjadi serangan pada *file* saat melalui proses pengunggahan ke *server*, pengguna tidak dapat mengetahui apakah *file* yang diunggahnya bebas dari manipulasi oleh pihak penyerang atau tidak. Hal ini tentu sangat merugikan bagi pengguna apabila *file* yang diunggahnya adalah *file* penting yang akan diakses kembali pada waktu yang akan datang. Untuk dapat memastikan apakah *file* yang diunggah oleh pengguna benar-benar merupakan *file* dari pengunggah, sistem harus menyediakan fitur autentikasi terhadap file yang diunggah.

Pada makalah ini diajukan mekanisme autentikasi file yang memanfaatkan tandatangan digital dengan algoritma ECDSA. Ide dasar pada mekanisme ini adalah client membaca sebuah file sebagai sebuah *binary string* serta membangkitkan sepasang kunci ECDSA, lalu membangkitkan sebuah *string signature* dari *file* yang dibaca tersebut dengan menggunakan kunci privat dari pasangan kunci ECDSA yang telah dibangkitkan. Setelah *signature* dibangkitkan, *client* akan mengirimkan paket data yang berisi *file*, kunci publik ECDSA yang digunakan untuk verifikasi, serta *signature* yang telah dibangkitkan pada proses sebelumnya. Pada sisi server, setelah menerima data yang dikirim oleh client, server akan melakukan proses verifikasi file yang diunggah. Tahap-tahap verifikasi meliputi pembacaan file yang diunggah dan mengkonversinya menjadi *byte array*, kemudian mencocokkan hasil dekripsi *signature* yang dikirim dengan hasil fungsi hash SHA1 dengan masukan berupa *byte array* dari file yang diunggah. Setelah proses verifikasi selesai, selanjutnya server akan mengirimkan respon pada client yang berisi status hasil verifikasi file yang telah diunggah. Dengan mekanisme ini pengguna dapat memastikan file yang telah diunggah dapat disimpan sama persis dengan file yang terdapat pada komputer client pengguna. Diagram alur kerja dari sistem autentikasi pengunggahan file ini dapat dilihat pada Gambar 4.



Gambar 4. Skema sistem autentikasi pengunggahan file

Sistem ini diimplementasikan dalam aplikasi berbasis web. Antarmuka client pada sistem ini diimplementasi dengan menggunakan html form, dan library Javascript untuk tandatangan digital jsrsasign. Tampilan antarmuka client sistem pengunggahan file dapat dilihat pada Gambar 5 berikut. Sedangkan pada sisi server, sistem diimplementasi dengan bahasa pemrograman Java yang berbasis web dengan menggunakan java servlet, dan memanfaatkan library pengolahan file serta API Java Security dan BouncyCastle untuk proses verifikasi file yang diunggah.



Gambar 5. Antarmuka client

Pada *library* enkripsi yang terdapat pada bahasa pemrograman Java (server) serta Javascript (client), terdapat beberapa jenis kurva eliptik yang dapat digunakan. Tabel 1 berikut menjabarkan sifat-sifat dari beberapa parameter yang ada pada *library* tersebut [3]. Kolom *strength* menjelaskan perkiraan jumlah bit keamanan yang diberikan. Kolom ukuran menjelaskan panjang bit parameter kurva eliptik. Kolom modulus menjelaskan perkiraan modulus algoritma. Kolom "Koblitz/Random" menyatakan parameter tersebut terasosiasi dengan kurva Koblitz atau dipilih secara acak.

Tabel 1. Sifat-sifat kurva eliptik parameter domain

Nama	*Strength*	Ukuran	Modulus	Koblitz/Random
secp112r1	56	112	512	Random
secp112r2	56	112	512	Random
secp128r1	64	128	704	Random
secp128r2	64	128	704	Random
secp160k1	80	160	1024	Koblitz
secp160r1	80	160	1024	Random
secp160r2	80	160	1024	Random
secp192k1	96	192	1536	Koblitz
secp192r1	96	192	1536	Random
secp224k1	112	224	2048	Koblitz
secp224r1	112	224	2048	Random
secp256k1	128	256	3072	Koblitz
secp256r1	128	256	3072	Random
secp384r1	192	384	7680	Random
secp521r1	256	521	15360	Random

Pada sistem yang diajukan dalam makalah ini, konfigurasi kurva eliptik yang digunakan adalah kurva eliptik 256 bit secp256r1. Kurva secp256r1 memiliki konfigurasi parameter a,b,p,G,n sebagai berikut.

a = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC

b = 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B

p = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF

= $2^{224}(2^{32} - 1) + 2^{192} + 2^{96} – 1$

G = 03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 (*compressed*)

= 04 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5 (*uncompressed*)

IV. Hasil Percobaan dan Analisis

Pengujian yang dilakukan pada sistem ini meliputi dua hal, yaitu pengujian apakah verifikasi dapat dilakukan pada platform berbeda (Java dan Javascript), serta apakah sistem dapat mengetahui adanya serangan yang terjadi pada proses pengunggahan file. Pengujian terhadap serangan dilakukan dengan cara melakukan *intercept* koneksi client pada saat proses pengunggahan file sedang berlangsung, kemudian dilakukan manipulasi terhadap konten data yang dilewatkan untuk selanjutnya diteruskan menuju server. Pengujian juga dilakukan dengan mengunggah berbagai jenis file dengan ukuran yang bervariasi. Rincian pengujian sistem dalam proses pembangkitan *digital signature* dapat dilihat pada Tabel 2, sedangkan rincian pengujian untuk proses autentikasi (verifikasi) file dapat dilihat pada Tabel 3 berikut.

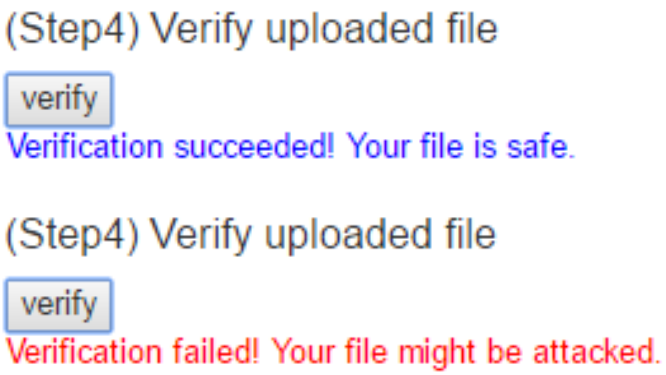
Tabel 2. Pengujian proses pembangkitan signature

No.	Jenis file	Ukuran	Durasi
1.	File teks (.txt)	3 *byte*	60 ms
2.	Foto (.jpg)	181 Kb	132 ms
3.	Audio (.mp3)	10111 Kb	3516 ms
4.	Dokumen (.pdf)	814 Kb	356 ms
5.	Executable (.exe)	646 Kb	323 ms

Tabel 3. Pengujian proses autentikasi file

No.	Jenis file	Ukuran	Durasi	Modifikasi?	Valid?
1.	File teks (.txt)	3 *byte*	1320 ms	Ya	Tidak
2.	Foto (.jpg)	181 Kb	39 ms	Tidak	Ya
3.	Audio (.mp3)	10111 Kb	353 ms	Tidak	Ya
4.	Dokumen (.pdf)	814 Kb	1336 ms	Ya	Tidak
5.	Executable (.exe)	646 Kb	75 ms	Tidak	Ya

Berdasarkan kedua tabel tersebut, sistem terbukti dapat melakukan autentikasi terhadap file yang diunggah oleh pengguna meskipun proses pembangkitan signature dan proses verifikasinya dijalankan secara terpisah pada platform yang berbeda. Antarmuka client setelah proses verifikasi file dilakukan dapat dilihat pada Gambar 6. Selain itu, sistem juga dapat mendeteksi adanya serangan berupa modifikasi paket data yang terjadi selama proses pengunggahan file berlangsung. Jika dicermati pada kedua tabel hasil pengujian, terlihat bahwa durasi proses pembangkitan *signature* sebanding dengan ukuran file yang diproses, sedangkan pada proses verifikasi *signature*, durasi proses tidak hanya bergantung pada ukuran file, namun juga dipengaruhi oleh adanya serangan terhadap file yang diunggah tersebut.



Gambar 6. Antarmuka client setelah proses validasi (atas: valid, bawah: terdeteksi serangan)

V. KESIMPULAN DAN SARAN

Berdasarkan hasil implementasi dan pengujian terhadap sistem autentikasi pengunggahan file yang dibahas pada makalah ini, dapat disimpulkan bahwa tandatangan digital dapat dimanfaatkan juga untuk melakukan autentikasi pada sistem pengunggahan file. Hal ini sangat bermanfaat bagi para pengguna aplikasi web yang menyediakan mekanisme pengunggahan file supaya para pengguna bisa mendapatkan jaminan autentikasi atas file yang diunggah ke server aplikasi tersebut.

Pada penelitian ini sistem dibangun dengan memanfaatkan kombinasi algoritma ECDSA sebagai algoritma tandatangan digital dan algoritma SHA1 sebagai algoritma fungsi hash. Penggunaan kombinasi algoritma tersebut masih memakan waktu yang cukup lama apabila file yang diunggah sudah mencapai ukuran *megabyte*. Oleh karena itu diharapkan pada penelitian selanjutnya dapat ditemukan metode yang lebih efisien yang dapat memproses autentikasi file dengan waktu yang relatif konstan meskipun ukuran file yang diunggah dapat bervariasi.

VI. REFERENSI

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation,* vol. 48, pp. 203-209, 1987.

[2] D. Johnson, A. Menezes dan S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36-63, 2001.

[3] Certicom Corp., "SEC 2: Recommended Elliptic Curve Domain Parameters," 2000. [Online]. Available: http://www.secg.org/SEC2-Ver-1.0.pdf. [Diakses 2016].

[4] R. Munir, "Slide Kuliah Rinaldi Munir: Fungsi Hash," 2016. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Fungsi%20Hash%20(2015).ppt. [Diakses 13 April 2016].

[5] R. Munir, "Slide Kuliah Rinaldi Munir: Tandatangan Digital," 2015. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Tandatangan%20Digital%20(2015).ppt. [Diakses 13 April 2016].