Code Signing pada Perangkat *Mobile* dan Pengembangannya pada *BlackBerry*

Aris Feryanto / 13507110[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*aris_feryanto@yahoo.com*

***Abstract*—Seiring dengan peningkatan *bandwidth* dan kemudahan akses, perangkat *mobile* dan juga perangkat Internet lainnya menjadi lebih terbuka terhadap celah-celah keamanan Internet. Salah satunya adalah melalui aplikasi yang didistribusikan dengan menggunakan media Internet. Pengguna perangkat *mobile* bisa tanpa sengaja mengunduh dan menjalankan aplikasi jahat atau bervirus yang mengakibatkan banyak dampak buruk. Untuk mengatasi masalah ini, diperlukan satu mekanisme untuk menjamin keamanan dari aplikasi yang diunduh. Salah satu solusinya adalah dengan melakukan *code signing* pada aplikasi yang didistribusikan. *Code signing* menjamin integritas dan memverifikasi sumber aplikasi, sehingga memudahkan pemeriksaan kerusakan dan pelacakan aplikasi. Makalah ini akan membahas mengenai *code signing* pada perangkat *mobile*, bagaimana cara kerjanya, dan percobaan mengembangkan sebuah simulasi *code signing* pada perangkat *BlackBerry*.**

Index Terms*—*Code signing*, verifikasi aplikasi simulasi *BlackBerry code signing

I. PENDAHULUAN

Sebagai seorang pengembang aplikasi yang baik, tentunya kita mengharapkan aplikasi yang kita kembangkan bisa digunakan dan disenangi oleh banyak orang. Namun, ketika kita menyediakan aplikasi untuk diunduh dari Internet, ada kemungkinan terjadi perubahan-perubahan atau modifikasi yang tidak diinginkan terhadap arsip aplikasi. Perubahan-perubahan ini bisa saja terjadi karena hal-hal sebagai berikut.

1. Gangguan jaringan komunikasi.
   Sebagai dasar dari protokol komunikasi dalam Internet, *Transmission Control Protocol/Internet Protocol* (TCP/IP), memiliki kelemahan, yaitu dapat diganggu pada saat transmisi data melalui *node-node* perantara. Sebuah sesi TCP/IP dapat diganggu dengan beberapa cara, yaitu *eavesdropping*, *tampering*, dan *impersonation* [3]. *Eavesdropping* adalah pencurian informasi yang sensitif, seperti nomor kartu kredit atau identitas pribadi lain. *Tampering* adalah pengubahan informasi yang dikirimkan ke penerima. Sedangkan *impersonation* adalah pemberian informasi dari atau kepada seseorang yang berpura-pura menjadi orang lain.
2. Kesengajaan pihak lain untuk memodifikasi.
   Perubahan pada isi aplikasi yang didistribusikan di Internet juga bisa disebabkan adanya orang yang secara sengaja mengunduh aplikasi, mengubah kodenya, dan kemudian mendistribusikan kembali aplikasi tersebut dengan nama yang sama, seperti proses *mirroring* pada arsip-arsip di server FTP.

Beberapa masalah yang timbul dan menjadi isu kemudian adalah sebagai berikut.

1. Bagaimana pengembang dapat memastikan integritas dari aplikasi yang mereka buat.
2. Bagaimana pengguna dapat memastikan bahwa aplikasi berasal dari sumber yang terpercaya.
3. Bagaimana pengembang sistem operasi (baik sistem operasi komputer, *smartphone*, atau perangkat lain) dapat menyediakan *platform* yang aman bagi pengembang aplikasi.

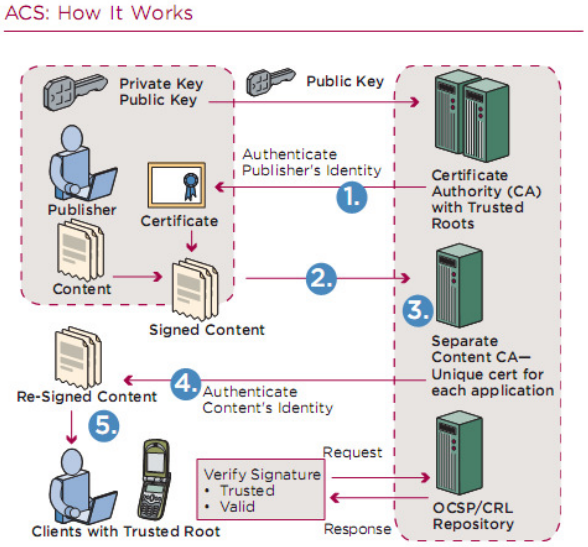
II. CODE SIGNING

Code signing adalah proses dimana sebuah kode ditandatangani secara dijital oleh pembuat kode dalam rangka untuk menjamin autentikasi dan integritas kode yang kuat bagi pemakai kode [2]. Kode yang dimaksud di sini adalah kode aplikasi (*executeable file*). Cara kerja *code signing* adalah dengan memberikan tanda tangan dijital menggunakan sistem kunci privat dan publik, mirip seperti pada SSL atau SSH.

Dengan *code signing*, pengembang menggunakan sebuah kunci untuk menandatangani aplikasi yang dibuat. Kunci ini bersifat unik untuk seorang pengembang atau sebuah grup atau bahkan untuk tiap aplikasi atau objek. Cara umum untuk melakukan tanda tangan dijital adalah dengan menggunakan sertifikat dari *certificate authority* (CA) yang terpercaya.

Code signing sangat berguna dalam lingkungan yang terdistribusi, dimana sumber dari sebuah bagian kode tidak langsung diketahui dengan jelas. Sebagai contoh, aplikasi *Java 2 Micro Edition* (J2ME) yang telah banyak didukung oleh perangkat-perangkat *mobile* yang ada

sekarang. Aplikasi J2ME banyak beredar di Internet dan bebas diunduh langsung melalui perangkat *mobile*. *Code signing* akan membantu pengguna untuk memilih aplikasi mana yang aman untuk dipakai.



Gambar 1. Cara kerja ACS

Code signing* menggunakan *certificate authority

Salah satu cara yang cukup banyak digunakan untuk melakukan *code signing* adalah dengan menggunakan sertifikat dari sebuah *certificate authority*. Dalam makalah ini dicontohkan cara kerja *Authenticated Content Signing* (ACS, lebih umum dari *code signing*, mencakup semua konten dijital) yang disediakan oleh *VeriSign*. Dari gambar 1, langkah-langkah cara kerjanya adalah sebagai berikut.

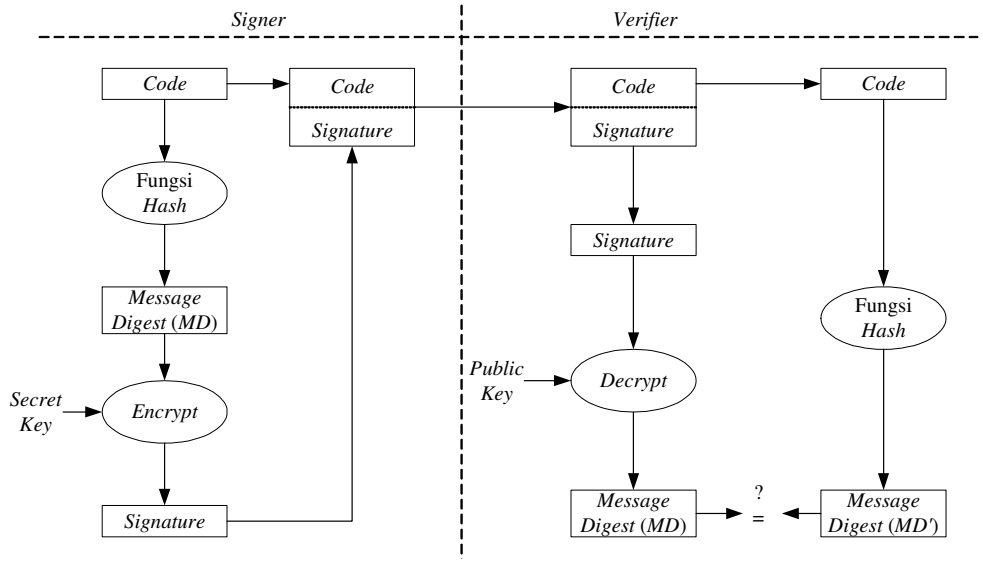
1. Pembuat konten dijital mendaftar dan mendapatkan sebuah sertifikat *Publisher* ID yang merupakan identitas dan legitimasi *vendor* sebagai pengembang aplikasi atau konten.
2. Pembuat konten kemudian menggunakan *Publisher* ID untuk menandatangani kode dan mengunggahnya ke *VeriSign* menggunakan antarmuka yang berbasis *browser* dan aman.
3. *VeriSign* memvalidasi tanda tangan pembuat konten dan membuat sebuah sertifikat *Content* ID yang mengandung informasi pembuat konten dan informasi aplikasi.
4. *VeriSign* menandatangani ulang konten yang diunggah menggunakan sertifikat *Content* ID tadi dan mengizinkan pembuat konten untuk mengunduh konten yang telah ditandatangan ulang tersebut.
5. Konten yang telah ditandatangan ulang atau diautentikasi ini sekarang siap didistribusikan secara aman melalui Internet.

***Code signing* menggunakan kunci sendiri**

Cara lain untuk melakukan *code signing* adalah dimana pengembang bisa menggunakan kunci yang dibuat sendiri. Pengguna kemudian harus mengambil kunci publik dari pengembang untuk memverifikasi bahwa sebuah aplikasi berasal dari pengembang tersebut. Langkah-langkahnya sebenarnya sama seperti tanda tangan dijital biasa, seperti pada gambar 2.

Namun, cara ini memiliki beberapa kelemahan, diantaranya:

1. Pengguna perangkat *mobile* harus mendapatkan kunci publik dan melakukan verifikasi secara manual.
2. Kurangnya tingkat kepercayaan karena kunci publik yang didapatkan bisa saja salah.
3. Tidak tersedianya aplikasi untuk melakukan pengecekan *content* yang ditandatangi dengan menggunakan kunci sendiri.



Gambar 2. Code signing menggunakan kunci sendiri

III. CODE SIGNING PADA PERANGKAT MOBILE

Vendor-vendor sistem operasi atau *platform mobile* pada umumnya sudah menyediakan fasilitas untuk melakukan *code signing* aplikasi yang dibangun di atas sistem operasi atau *platform*-nya.

1. iPhone OS
   Apple sebagai pengembang iPhone OS telah menyiapkan mekanisme untuk menjamin aplikasi yang didistribusikan adalah aplikasi yang benar dan terpercaya. Apple menyediakan App Store yang merupakan bagian dari aplikasi *iTunes* yang menjadi tempat bagi pengguna untuk mengunduh aplikasi-aplikasi untuk iPhone. Aplikasi yang masuk ke App Store telah diperiksa terlebih dahulu dan diberi *code signing*.
2. BlackBerry OS
   Pada BlackBerry OS yang dikembangkan oleh *Research In Motion* (RIM), aplikasi yang harus diberi *code signing* adalah aplikasi-aplikasi yang menggunakan *Application Programming Interface* (API) khusus, misalnya API yang berkaitan dengan invokasi aplikasi lain dalam BlackBerry atau API yang berkaitan dengan kriptografi. *Code signing* pada BlackBerry berbayar, tidak gratis. *Code signing* pada BlackBerry tidak melibatkan *certificate authority*, tapi pengelolaan kuncinya langsung dilakukan oleh pihak RIM sendiri. Tujuan dari *code signing* di sini lebih kepada menjaga keamanan data pengguna BlackBerry dan menjaga agar akses ke kode yang kritis tidak dapat dilakukan sembarangan, sehingga tidak mengganggu keamanan aplikasi BlackBerry.
3. Symbian OS
   Hampir mirip seperti aplikasi pada BlackBerry OS, aplikasi pada Symbian OS juga harus diberi *code signing* jika ingin menggunakan fitur-fitur tertentu. Sama seperti BlackBerry, *code signing* pada Symbian juga berbayar, namun Symbian memanfaatkan *certificate authority*, yakni VeriSign untuk membantu penanganan sertifikat dijitalnya.
4. Brew *platform*
   Binary Runtime Environment for Wireless atau Brew adalah sebuah *platform* yang berjalan di atas *chipset* Qualcomm yang biasanya terdapat pada ponsel

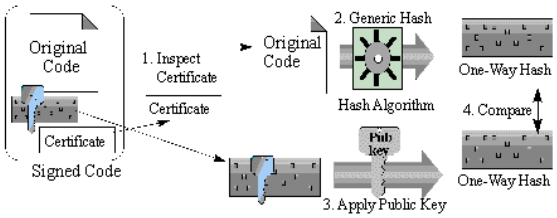
   CDMA. Aplikasi yang dibangun di atas Brew harus melalui suatu proses pengujian yang dinamakan *True Brew Test* (TBT) agar dapat didistribusikan. Selain itu, semua aplikasi Brew yang akan didistribusikan juga harus diberi *code signing*. *Code signing* pada Brew menggunakan jasa *certificate authority VeriSign*, sama seperti aplikasi Symbian OS.
5. J2ME *platform*
   Platform ini adalah *platform* yang paling umum terdapat pada sebagian besar perangkat *mobile*. *Platform* J2ME telah memiliki sistem *code signing* yang memanfaatkan sertifikat dijital yang dipercayakan kepada *certificate authority* (CA). Beberapa CA yang cukup terkenal untuk pemberian *code signing* pada J2ME adalah VeriSign dan Thawte. Dengan melakukan *code signing*, pembuat aplikasi J2ME dapat mengakses fitur-fitur, seperti akses baca dan tulis file, kamera, koneksi internet. Namun, aplikasi J2ME tidak diwajibkan harus memiliki sertifikat untuk dapat menjalankan fitur-fitur tersebut. Aplikasi tanpa sertifikat tetap dapat dipasang dan dijalankan, hanya saja terdapat notifikasi untuk mengizinkan akses tiap kali aplikasi mengakses fitur-fitur tertentu.

Berdasarkan uraian di atas, dapat dilihat terdapat perbedaan-perbedaan dalam aplikasi *code signing* pada berbagai perangkat *mobile*. Untuk perbandingan yang lebih jelas, dapat dilihat pada tabel di bawah ini.

	iPhone OS	BlackBerry OS	Symbian OS	Brew platform	J2ME platform
Tujuan	Verifikasi keamanan aplikasi yang berjalan	Membatasi penggunaan API tertentu demi keamanan perangkat	Membatasi penggunaan fitur-fitur tertentu	Memverifikasi aplikasi yang didistribusikan	Mengamankan akses ke fitur-fitur tertentu
Sertifikasi	Menggunakan mekanisme sendiri	Menggunakan mekanisme sendiri	Menggunakan CA VeriSign	Menggunakan CA VeriSign	Menggunakan beberapa CA, seperti VeriSign dan Thawte
Semua aplikasi	Ya	Tidak	Tidak	Ya	Tidak
Berbayar	Ya	Ya	Ya	Ya	Ya
Dapat dijalankan tanpa *signing*	Tidak	Tidak, bila menggunakan API tertentu	Tidak, bila menggunakan fitur tertentu	Tidak	Ya, namun selalu muncul notifikasi untuk akses fitur tertentu

IV. IMPLEMENTASI CODE SIGNING PADA BLACKBERRY

Dari penjabaran pada bab sebelumnya, dapat dilihat bahwa pada *BlackBerry*, belum ada mekanisme *code signing* yang ditujukan untuk memverfikasi aplikasi. *Code signing* pada *BlackBerry* hanya dilakukan sekedar untuk membatasi penggunaan API khusus. Oleh karena itu, penulis melihat adanya keperluan untuk mengembangkan aplikasi yang mampu melakukan verifikasi aplikasi.



Gambar 3 Proses dalam code signing yang dirancang

Proses verifikasi kode yang telah diberi *code signing* dapat dilihat seperti pada gambar 3. Sebelum diverifikasi, aplikasi akan diberi *code signing*. *Code signing* sebenarnya dapat dilakukan di luar lingkungan pengembangan *BlackBerry*, namun kali ini penulis akan membuat *code signing generator* juga dalam aplikasi *BlackBerry*. Adapun langkah-langkah dan algoritma *hash* dan kriptografi yang digunakan dalam aplikasi ini adalah sebagai berikut.

1. Pertama-tama, arsip kode aplikasi yang akan diberi *code signing* dibaca dan dibuat nilai *hash*-nya dengan menggunakan algoritma MD5.
2. Selanjutnya nilai *hash* ini akan dienkripsi dengan menggunakan algoritma RSA menggunakan kunci privat dari pihak pengembang aplikasi yang diberi *code signing*.
3. Nilai *hash* yang telah dienkripsi inilah yang disebut sebagai *signature* dan disimpan pada file terpisah untuk keperluan verifikasi nantinya.

Sebaliknya, verifikasi dilakukan dengan langkah-langkah sebagai berikut.

1. File *signature* dibaca dan didekripsi menggunakan kunci publik yang didapatkan dari pengembang aplikasi yang akan diverifikasi. Hasil dekripsi ini adalah sebuah nilai *hash*.
2. File kode aplikasi dibaca dan dihitung nilai *hash*-nya dengan algoritma MD5 yang sama dengan algoritma yang digunakan saat memberi *code signing*.
3. Nilai *hash* hasil dekripsi dan nilai *hash* hasil perhitungan dari file aplikasi dibandingkan. Bila sama, maka aplikasi yang diverifikasi adalah aplikasi yang sah dari pengembangnya. Bila tidak sama, mungkin file aplikasi sudah pernah diubah atau juga mungkin terdapat kesalahan pada kunci publik yang digunakan.

Berikut ini adalah potongan kode yang digunakan untuk melakukan verifikasi file aplikasi.

```
private void verifyAction() {
        int index = _list.getSelectedIndex();
    FileExplorerDemoFileHolder fileholder = (FileExplorerDemoFileHolder) _list.get(_list, index);

    if (fileholder != null)
    {
        String filename = fileholder.getPath() + fileholder.getFileName();

        if (Dialog.ask(Dialog.D_OK_CANCEL, "Pastikan file signature telah terdapat " +
                    "pada direktori yang sama.") == Dialog.D_OK)
        {
            FileConnection fc = null;
            FileConnection fcSig = null;

            try
            {
                fc = (FileConnection)Connector.open("file:///" + filename);
                InputStream fis = fc.openInputStream();
                int retval;

                // Membuat instance MD5 digest
                MD5Digest digest = new MD5Digest();
                DigestOutputStream digestStream = new DigestOutputStream( digest, null );
                // Menulis isi file untuk mendapatkan digest
                while ((retval = fis.read()) != -1)
                {
                    digestStream.write( retval );
                }

                // Membaca signature dari file
                byte[] signature = new byte[32];

                fcSig = (FileConnection)Connector.open("file:///" + filename + ".sig");
                InputStream sigIs = fcSig.openInputStream();
```

Menghitung *hash*

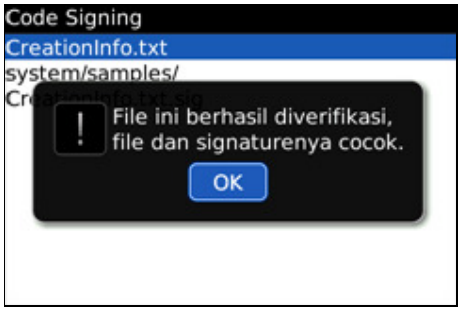
```
                int i = 0;
                while ((retval = sigIs.read()) != -1)
                {
                        signature[i++] = (byte) retval;
                }

                fcSig.close();

                // Membangkitkan hash dengan melakukan dekripsi dari signature yang dibaca
                byte[] hash = new byte[32];
                sampleRSADecryption(16, d, p, q, signature, hash, i - 1);

                // Jika hasil dekripsi sama dengan hash yang dihitung dari file,
                // maka file berhasil diverifikasi
                if (digest.getDigest() == hash)
                {
                        Dialog.alert("File ini berhasil diverifikasi, file dan signaturenya cocok.");
                }
                else
                {
                        Dialog.alert("File ini gagal diverifikasi, " +
                            "mungkin telah ada perubahan pada file.");
                }
            }
            catch (Exception ex)
            {
                Dialog.alert("Gagal memverifikasi : " + filename);
            }
        }
    }
}
```

Mendekripsi *signature*

Membandingkan

Beberapa *screenshot* aplikasi ini adalah sebagai berikut.

Gambar 4 Membuat code signing



Gambar 5 Verifikasi code signing

V. MASALAH PADA CODE SIGNING

Seperti berbagai masalah keamanan yang lain, *code signing* ternyata juga dapat diserang. Beberapakelemahan dari sistem *code signing* adalah sebagai berikut.

1. Pengguna dapat dijebak untuk menjalankan kode aplikasi yang belum diberi *code signing*. Pada beberapa *platform mobile* yang mengizinkan aplikasi untuk tetap dapat dijalankan tanpa diberi *code signing*, pengembang aplikasi yang jahat bisa menjebak penggunanya dengan memberikan penjelasan yang menipu. Misalnya, dengan menjelaskan adanya fitur yang sangat menarik bila pengguna memilih sebuah menu. Sehingga pengguna menjalankan aplikasi, meskipun tanpa menggunakan *code signing*.

2. Sistem tetap aman hanya selama kunci privat yang digunakan tidak diketahui pihak lain selain pengembang.

3. Perlu diperhatikan bahwa *code signing* pada beberapa *platform*, seperti *iPhone*, *BlackBerry*, dan J2ME tidak menjamin pengguna dari serangan berbahaya atau kesalahan (*bug*) pada program. *Code signing* hanya memberikan jaminan bahwa sebuah program atau aplikasi benar-benar berasal dari sebuah pengembang tertentu dan tidak diubah oleh pihak lain.

VI. KESIMPULAN DAN SARAN

Beberapa kesimpulan yang dapat ditarik dari pembahasan mengenai *code signing* dan pembuatan aplikasinya pada *BlackBerry* adalah:

1. Tujuan dasar dari *code signing* pada perangkat *mobile*

adalah menjaga integritas aplikasi, terutama aplikasi yang dapat diunduh secara bebas dari Internet.
2. Implementasi *code signing* dapat dilakukan dengan menggunakan jasa *certificat authority* (CA) ataupun dengan menggunakan kunci sendiri.
3. Implementasi *code signing* pada perangkat *mobile* sangat bervariasi, tergantung dari pengembang sistem operasi atau *platform*.
4. Penulis berhasil membuat simulasi *code signing* pada perangkat *BlackBerry* yang mungkin dapat dikembangkan lagi.
5. *Code signing* memiliki beberapa kelemahan, seperti: pengguna yang dapat dijebak, keamanan bergantung pada kerahasiaan kunci, dan tidak menjamin program bebas dari kesalahan pemrograman atau *bug*.

Penerapan *code signing* pada beberapa aplikasi *mobile* sudah baik. Hanya saja, penulis melihat adanya celah dimana pada beberapa *platform*, aplikasi tanpa *code signing* masih dapat dijalankan. Menurut penulis, hal ini harus dihindari dan sebaiknya aplikasi tanpa *code signing* langsung diblok atau tidak dapat dijalankan.

DAFTAR PUSTAKA

[1] Mobile Code Security, AD Rubin, DE Geer Jr - IEEE Internet Computing, 1998, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.301&rep=rep1&type=pdf

[2] Mobile Code Security, S Loureiro, R Molva, Y Roudier - Proceedings of ISYPAR, 2000 – Citeseer, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.15.833&rep=rep1&type=pdf

[3] Using Public Key Cryptography in Mobile Phones, Discretyx, http://www.discretix.com/PDF/Using%20Public%20Key%20Cryptography%20in%20Mobile%20Phones.pdf

[4] Code Signing Resources, Verisign http://www.verisign.com/code-signing/information-center/resources/index.html

[5] VeriSign Authenticated Content Signing, http://www.verisign.com/stellent/groups/public/documents/data_sheet/003201.pdf

[6] BlackBerry Developer, http://www.blackberry.com/developers/

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, 29 April 2010

Aris Feryanto
13507110