Proteksi Perangkat Lunak dengan Algoritma Kirptografi Kunci Publik

Anugrah Redja Kusuma, Rizki Yulianto, dan Widhiyo Sudiyono

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : if11072@students.if.itb.ac.id , if11006@students.if.itb.ac.id, if11027@students.if.itb.ac.id

Abstrak

Perkembangan internet membuat distribusi perangkat lunak semakin mudah. Pengguna dapat men-*download* file instalasi langsung dari situs pembuat perangkat lunak di internet. Dengan sistem pendistribusian tersebut muncul permasalahan: bagaimana menjaga keaslian perangkat lunak dan bagaimana membatasi perangkat lunak agar dapat digunakan pihak tertentu saja. Dalam makalah ini permasalahan tersebut dicoba untuk diatasi dengan menerapkan beberapa metode proteksi perangkat lunak berbasis algoritma kriptografi kunci publik.

***Kata kunci**: distribusi perangkat lunak, proteksi perangkat lunak, algoritma kriptografi kunci publik*

1. Pendahuluan

Seiring dengan berkembangnya internet, terjadi banyak perubahan dalam sistem pendistribusian perangkat lunak. Perangkat lunak yang dahulu biasa distribusikan melalui toko-toko penjual perangkat lunak, sekarang dapat diperoleh dengan men-*download* file instalasinya langsung dari situs pembuat perangkat lunak, atau melalui situs-situs yang menyediakan berbagai perangkat lunak yang dapat di-*download*.

Dari sistem pendistribusian seperti di atas muncul beberapa masalah yang harus diperhatikan oleh pembuat perangkat lunak. Diantaranya:

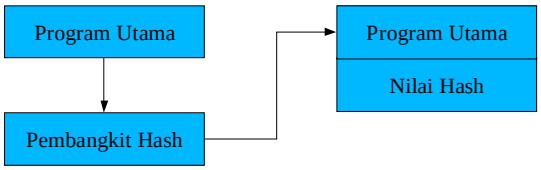
a. Bagaimana menjaga keaslian perangkat lunak dari upaya perubahan oleh pihak luar.
b. Bagaimana membatasi perangkat lunak agar hanya dapat digunakan oleh pihak tertentu, misalnya pihak yang sudah membeli perangkat lunak tersebut.

Makalah ini mencoba menjawab kedua pertanyaan di atas. Metode yang digunakan dititikberatkan pada penggunaan algoritma kriptografi kunci publik.

2. Menjaga Keaslian Perangkat Lunak

Salah satu metode yang dapat digunakan untuk memeriksa keaslian file program adalah dengan menghitung nilai *hash* dari file tersebut dan menyimpan pada tempat tertentu, misalnya ditambahkan (*embed*) di akhir file asli (gambar 1). Kemudian ketika program berjalan, di awal proses ia menghitung kembali nilai *hash* untuk dirinya sendiri, dan membandingkannya dengan nilai

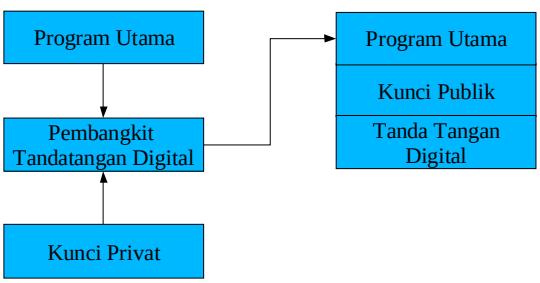
hash yang telah disimpan sebelumnya. Bila kedua nilai sama, maka dapat diasumsikan bahwa file program tersebut masih asli.



Gambar 1. Skema proteksi dengan menggunakan fungsi *hash*

Kelemahan metode ini adalah apabila pihak luar ingin memodifikasi program utama, ia dapat menghitung kembali nilai *hash* dari program termodifikasi dan menimpa nilai *hash* lama dengan nilai *hash* baru.

Dengan sedikit perubahan, metode di atas dapat diadaptasi untuk menggunakan tanda tangan digital (gambar 2). Langkah pertama yang harus dilakukan pembuat perangkat lunak adalah membangkitkan sepasang kunci --kunci publik dan kunci privat-- sebagai parameter untuk algoritma kriptografi kunci publik. Kunci yang dibangkitkan sebaiknya memiliki panjang tertentu untuk meningkatkan faktor keamaanan. Misalnya untuk algoritma RSA disarankan panjang kunci minimal adalah 1024-bit[1)]. Dengan menggunakan kunci privat maka dapat dihitung tanda tangan digital file program. Tanda tangan digital dan kunci publik kemudian ditambahkan di akhir file program.



Gambar 2. Skema proteksi dengan menggunakan tanda tangan digital

Kelemahan metode proteksi perangkat lunak menggunakan fungsi *hash* juga muncul pada metode tanda tangan digital. Pihak luar yang ingin memodifikasi program dapat membangkitkan kunci privat dan publiknya nya sendiri. Kemudian menghitung tanda-tangan digital menggunakan kunci privat miliknya (bukan milik pembuat perangkat lunak), dan mengganti kunci publik dan tanda tangan digital pada program terproteksi dengan nilai yang baru.

Untuk mengatasi kelemahan tersebut, kunci publik dibuat agar tidak dapat dimodifikasi oleh pihak luar. Misalnya dengan menyimpannya, tidak dalam program, melainkan pada *server* di internet. Dengan begitu pihak luar tidak dapat menghitung tanda tangan digital yang sesuai dengan kunci publik karena ia tidak mengetahui kunci privatnya. Sehingga bila terjadi modifikasi file, akan mengakibatkan proses verifikasi gagal.

3. Membatasi Penggunaan Perangkat Lunak

Banyak perangkat lunak yang tersedia saat ini menggunakan kode registrasi untuk membatasi penggunaan perangkat lunak. Ketika perangkat lunak dijalankan tanpa menggunakan kode registrasi yang benar maka ada beberapa fungsionalitas program yang tidak dapat diakses.

Dalam bukunya, Cerven[1)] menyebutkan bahwa ada beberapa tipe proteksi menggunakan kode registrasi , diantaraya:

1. Kode registrasi selalu sama

2. Kode registrasi berubah sesuai informasi yang dimasukkan (perusahaan, nama, dan seterusnya)
3. Kode registrasi berubah sesuai spesifikasi komputer pengguna
4. Kode registrasi diperiksa secara *online*

Tipe-tipe proteksi diatas dapat dibuat dengan memanfaatkan algoritma kriptografi kunci publik, khususnya dalam penggunaannya untuk tanda tangan digital. Yaitu dengan cara menganggap informasi yang dibutuhkan untuk proses registrasi (misalnya nama perusahaan, nama pengguna, spesifikasi komputer pengguna, dan seterusnya) sebagai dokumen yang akan ditandatangani. Nilai tanda tangan digital yang dihasilkan merupakan kode registrasinya.

Kode registrasi tersebut kemudian dapat dikirimkan kepada pengguna menggunakan *email* atau media pengiriman lainnya. Berikut ini adalah contoh kode registrasi sebuah perangkat lunak yang menggunakan algoritma kriptografi RSA-2048 bit:

```
rpSxwsCRTTfOsNkRBib4QjGWgSFZjvfCN1OanTFvwqMyf43o
hC2+dz/wvrwUW/s7FlYg5b3+dwuJS43cZwaXxtvQN3Sy2kcB
tIsiyb4VNWZCX/i3PobYLmGm0f2PIiLnTP6Uude4lMNq9BAH
fqgNNEQ3sibob0WawrIMdlFQHsXjoOqz/TPDnTaLI7hF0em+
kjRQYD8ZbEb7uWeuOvZmndi+rI5/MzVhb9i6J7e7Ts6NVDL1
RZUgu6NwTQIzbfLTdpD3NyjYxdpCXYnhzy3FBD7ASKvXNSM8
2lu6Yl3IKPyQ/3eAQcISXCd3B96uff3EFyL7cPEdgP/u/IvW
8kwQr1==
```

Pada contoh di atas kode registrasi terlihat panjang, sehingga sulit untuk ditulis oleh pengguna. Dengan adanya permasalaah ini, beberapa pembuat perangkat lunak memilih untuk menggunakan kunci yang pendek, sehingga dihasilkan kode registrasi yang lebih singkat. Namun tentunya langkah ini mengurangi tingkat keaamanan perangkat lunak. Berikut ini adalah contoh kode registrasi sebuah perangkat lunak yang dibuat dengan menggunakan algoritma kriptografi Elliptic-Curve 62-bit:

```
JCF8T-2MG8G-Q6BBK-MQKGT-X3GBC
```

4. Kesimpulan

Permasalahan yang ditemui pada pendistribusian perangkat lunak melalui internet adalah: bagaimana menjaga keaslian perangkat lunak dan bagaimana membatasi perangkat lunak agar dapat digunakan pihak tertentu saja.

Permasalahan tersebut dapat diatasi salah satunya dengan menggunakan algoritma kriptografi kunci publik, terutama dalam pemanfaatannya untuk tanda tangan digital.

Untuk menjaga keaslian perangkat lunak dapat dihitung tanda tangan digital dari file program. Nilai yang dihasilkan dapat ditambahkan pada akhir file program.

Untuk membatasi penggunaan perangkat lunak dapat digunakan kode registrasi. Informasi-informasi yang dibutuhkan dalam proses registrasi dapat dianggap sebagai dokumen yang akan ditandatangani. Nilai tanda tangan yang dihasilkan merupakan kode registrasi perangkat lunak.

[1] Cerven, Parfol, *Crackproof Your Software*, No Scratch Press, San Fransisco, 2002