ANALISIS DAN IMPLEMENTASI *ELLIPTIC CURVE CRYPTOGRAPHY* DAN APLIKASINYA PADA SISTEM *FILE SAVE GAME* NINTENDO WII

Shieny Aprilia – NIM : 13505089

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jln. Ganesha 10, Bandung
E-mail : if15089@students.if.itb.ac.id

Abstrak

Makalah ini menjabarkan hasil analisis dan implementasi *Elliptic Curve Cryptography*, yaitu salah satu algoritma kriptografi kunci publik, dan aplikasinya pada sistem file *save game* Nintendo Wii. File *save game* perlu dilindungi dengan menggunakan teknik kriptografi agar suatu file *save game* hanya dapat diakses oleh pihak yang berkepentingan saja. Pada makalah ini akan dijabarkan bagaimana cara kerja algoritma ECC dan contoh implementasinya yang dikembangkan pada *open source library* yang memiliki ECC *library*.

Kata Kunci: *Elliptic Curve Cryptography*, *digital signature*, kunci publik, kunci privat, penandatanganan, verifikasi, file *save game*, Nintendo Wii

1. Pendahuluan

Sampai saat ini, dunia game sudah berkembang sangat pesat. Bermacam-macam teknologi game telah diperbarui. Selain itu, mesin untuk bermain game pun semakin beraneka ragam, beberapa di antaranya yang masih cukup populer sekarang ini adalah Sony Playstation, Microsoft Xbox360, dan Nintendo Wii. Seperti layaknya komputer, setiap mesin game tersebut memiliki media penyimpanan data yang digunakan untuk menyimpan file *save game* yang digunakan untuk menyimpan kemajuan atau nilai yang diperoleh para pemainnya, sehingga pemain dapat melanjutkan suatu game dari tempat ia meninggalkan game tersebut dengan cara me-*load* file *save game* tersebut. Namun fitur penyimpanan file *save game* ini harus menjamin bahwa suatu file milik seorang pemain A hanya dapat diakses oleh pemain A saja. Oleh karena itu, diperlukan suatu sistem keamanan yang dapat mewujudkan hal ini.
Teknik yang digunakan pada sistem keamanan ini biasanya merupakan teknik kriptografi yang mengenkripsi file atau melakukan *digital signature* terhadap file sehingga hanya pemain yang berkepentinganlah yang dapat mengakses file tersebut. Salah satunya adalah teknik yang diaplikasikan pada sistem file *save game* Nintendo Wii, yaitu *Elliptic Curve Cryptography*.

2. *Elliptic Curve Cryptography*

Elliptic Curve Cryptography adalah sebuah algoritma kriptografi kunci publik, yaitu algoritma di mana setiap pihaknya memiliki sepasang kunci privat dan kunci publik. Kunci privat hanya dimiliki oleh segelintir pihak, sedangkan kunci publik disebarluaskan ke semua pihak.
Operasi matematika pada ECC merupakan fungsi kurva eliptik $y^2 = x^3 + ax + b$, di mana $4a^3 + 27b^2 \neq 0$. Setiap nilai a dan b yang berbeda dapat menghasilkan bentuk kurva eliptik yang berbeda, dan semua nilai x dan y (titik kordinat) yang memenuhi persamaan di atas akan memotong kurva eliptik tersebut. Kunci publik pada algoritma ECC adalah sebuah titik pada kurva eliptik dan kunci privatnya adalah sebuah angka random. Kunci publik diperoleh dengan melakukan operasi perkalian terhadap kunci privat dengan titik generator G pada kurva eliptik.

2.1 Masalah Logaritma Diskrit

Keamanan ECC bergantung pada masalah logaritma diskrit yang sulit untuk dipecahkan. Misalnya P dan Q adalah dua titik pada suatu kurva eliptik, sedemikian rupa sehingga kP = Q, di mana k adalah sebuah nilai skalar. Dengan mengetahui nilai P dan Q, tidak mungkin bisa mendapatkan nilai k, jika k adalah nilai yang sangat besar. k adalah logaritma diskrit basis P terhadap Q. Oleh karena itu, operasi utama pada ECC adalah perkalian titik seperti contoh di atas. Pada bagian-bagian selanjutnya akan dijelaskan tiga operasi yang dilakukan pada implementasi algoritma ECC.

2.2 Perkalian Titik

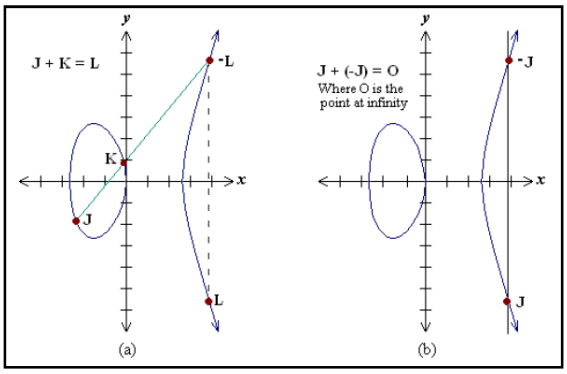
Pada operasi ini, seperti yang telah dijelaskan di atas, sebuah titik P pada kurva eliptik dikalikan dengan bilangan skalar k dengan menggunakan persamaan kurva eliptik untuk mendapatkan titik Q pada kurva eliptik yang sama. Dengan demikian kP = Q.

Perkalian titik didapatkan dengan melakukan dua operasi dasar kurva eliptik sebagai berikut :

a. Operasi pertambahan titik, yaitu menambahkan dua buah titik J dan K untuk mendapatkan titik L. Dengan demikian L = 2J.
b. Operasi penggandaan titik, yaitu menambahkan titik J dengan dirinya sendiri untuk mendapatkan titik L. Dengan demikian L = 2J.

2.3 Pertambahan Titik

Pada operasi ini, dua buah titik J dan K pada sebuah kurva eliptik ditambahkan satu sama lain untuk mendapatkan titik L pada kurva eliptik yang sama.

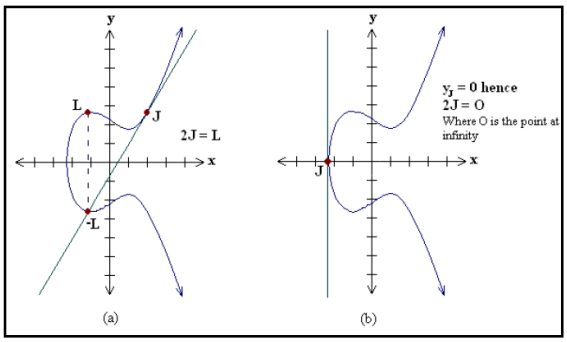


Gambar 1 Ilustrasi Operasi Pertambahan Titik pada Kurva Eliptik

Pada gambar di atas, titik J dan K adalah titik yang berada pada kurva eliptik. Jika K ≠ -J maka sebuah garis yang melewati J dan K akan memotong kurva eliptik dan melewati sebuah titik –L. Pencerminan dari –L terhadap sumbu X akan menghasilkan titik L, yang merupakan hasil dari penambahan J dan K.

2.4 Penggandaan Titik

Penggandaan titik adalah penambahan titik J pada kurva eliptik dengan dirinya sendiri untuk mendapatkan titik L pada kurva eliptik yang sama.



Gambar 2 Ilustrasi Operasi Penggandaan Titik pada Kurva Eliptik

Untuk mendapatkan L dengan menggandakan titik J, sehingga L = 2J, misalkan ada sebuah titik J pada kurva eliptik. Jika ordinat titik J tidak sama dengan nol, maka garis yang tangen pada J akan melalui –L. –L adalah pencerminan L terhadap sumbu X, dengan demikian L = 2J.

2.5 ECDSA – *Elliptic Curve Digital Signature Algorithm*

ECDSA adalah varian dari *Digital Signature Algorithm* (DSA) yang menggunakan teknik *Elliptik Curve Cryptography*. Untuk mengirim pesan yang ditandatangan dari A ke B, kedua belah pihak harus menyetujui semua parameter persamaan yang digunakan untuk menjalankan algoritma. Secara singkat, proses ECDSA adalah sebagai berikut :

1. Pembangkitan tanda tangan
   Untuk menandatangan pesan M dari pengirim A, digunakan kunci privat A yaitu P_A.
   a. Hitung e = hash(M), di mana hash adalah fungsi hash kriptografik, seperti MD5
   b. Pilih bilangan acak k antara 1 sampai dengan n-1
   c. Hitung r = x_1 mod n, di mana (x_1, y_1) = k * G. Jika r = 0, kembali ke langkah b
   d. Hitung s = k^{-1} (e + P_Ar)mod n. Jika s = 0, kembali ke langkah b
   e. Tanda tangan telah dibangkitkan, yaitu pasangan (r, s)
2. Verifikasi tanda tangan
   Untuk melakukan otentikasi tanda tangan A, B menggunakan kunci publik A, yaitu Q_A.

a. Periksa apakah r dan s adalah bilangan bulat antara 1 dan n-1. Jika tidak, maka tanda tangan tidak valid
b. Hitung e = hash(m), di mana hash adalah fungsi yang sama yang digunakan pada pembangkitan tanda tangan
c. Hitung $w = s^{-1} \pmod n$
d. Hitung $u_1 = ew \pmod n$ dan $u_2 = rw \pmod n$
e. Hitung $(x_1, y_1) = u_1G + u_2Q_A$
f. Tanda tangan valid jika dan hanya jika $x_1 = r \bmod n$

3. Implementasi *Elliptic Curve Cryptography*

Pada bagian ini, akan ditampilkan sejumlah contoh implementasi algoritma ECC untuk menandatangan dan melakukan verifikasi data teks dengan menggunakan ECC. Contoh implementasi ini menggunakan OpenSSL, yaitu *open source library* dalam bahasa C yang dilengkapi dengan *library* ECC.

Contoh di bawah ini adalah contoh penggunaan *library* ECC pada OpenSSL untuk melakukan *digital signature* data teks :

```
int main(int argc, char *argv[]) {
        key =
EC_KEY_new_by_curve_name(NID_sect233r1);
        if (key == NULL) {

        fatalerror("EC_KEY_new_by_curve_nam
e");
        }

        ret = BN_hex2bn(&priv_key,argv[1]);
        ret = BN_hex2bn(&pub_key_x,argv[2])
&& ret;
        ret = BN_hex2bn(&pub_key_y,argv[3])
&& ret;

        if(ret == 0) {
                fatalerror("Kunci salah.");
        }

        if(!EC_KEY_set_private_key(key,
priv_key)) {

        fatalerror("EC_KEY_set_private_key"
);
        }

        group = EC_KEY_get0_group(key);
        pub_key = EC_POINT_new(group);

        if(!EC_POINT_set_affine_coordinates
_GFp(group, pub_key, pub_key_x, pub_key_y,
NULL)) {

        fatalerror("EC_PINT_set_affine_coor
dinates_GFp");
        }
        if(!EC_KEY_set_public_key(key,
pub_key)) {

        fatalerror("EC_KEY_set_public_key")
;
        }

        BIO_printf(bio_stderr,"Membaca
argumen kunci:\n");
        writekey(bio_stderr,"PRIV:
",priv_key);
        writekey(bio_stderr,"PUBX:
",pub_key_x);
        writekey(bio_stderr,"PUBY:
",pub_key_y);

        if (!EC_KEY_check_key(key)) {

        fatalerror("EC_KEY_check_key");
        }

        BIO_printf(bio_stderr,"Membuat
digest.\n");
        EVP_MD_CTX_init(&md_ctx);
        EVP_DigestInit(&md_ctx,
EVP_ecdsa());

        while(!feof(stdin)) {
                inlen = BIO_read(bio_stdin,
inbuf, 80);

        if(!EVP_DigestUpdate(&md_ctx,
inbuf, inlen)) {

        fatalerror("EVP_DigestUpdate");
                }
        }
        EVP_DigestFinal(&md_ctx, digest,
&dgst_len);

        //bagian ini yang melakukan
penandatanganan file
        signature = ECDSA_do_sign(digest,
dgst_len, key);

        digest_bn = BN_bin2bn(digest,
dgst_len, NULL);
        BIO_printf(bio_stderr,"SHA1:
%s\n",BN_bn2hex(digest_bn));

        BIO_printf(bio_stdout,"\nTanda
tangan:\n");

        writekey(bio_stdout,"SIGNR:
",signature->r);
        writekey(bio_stdout,"SIGNS:
",signature->s);

        return 0;
}
```

Contoh di bawah ini adalah contoh penggunaan *library* ECC pada OpenSSL untuk verifikasi data teks yang telah ditandatangan dengan ECC:

```
int main(int argc, char *argv[]) {
        key =
EC_KEY_new_by_curve_name(NID_sect233r1);
        if (key == NULL) {

        fatalerror("EC_KEY_new_by_curve_nam
e");
        }

        ret =
BN_hex2bn(&pub_key_x,argv[1]);
```

```
        ret = BN_hex2bn(&pub_key_y,argv[2])
&& ret;
        ret = BN_hex2bn(&signature-
>r,argv[3]) && ret;
        ret = BN_hex2bn(&signature-
>s,argv[4]) && ret;

        if(ret == 0) {
                fatalerror("Kunci salah.");
        }

        group = EC_KEY_get0_group(key);
        pub_key = EC_POINT_new(group);

        if(!EC_POINT_set_affine_coordinates
_GFp(group, pub_key, pub_key_x, pub_key_y,
NULL)) {

        fatalerror("EC_PINT_set_affine_coor
dinates_GFp");
        }
        if(!EC_KEY_set_public_key(key,
pub_key)) {

        fatalerror("EC_KEY_set_public_key")
;
        }

        BIO_printf(bio_stderr,"Membaca
argumen data kunci:\n");
        writekey(bio_stderr,"PUBX:
",pub_key_x);
        writekey(bio_stderr,"PUBY:
",pub_key_y);

        BIO_printf(bio_stderr,"Membaca
tanda tangan:\n");
        writekey(bio_stdout,"SIGNR:
",signature->r);
        writekey(bio_stdout,"SIGNS:
",signature->s);

        if (!EC_KEY_check_key(key)) {

        fatalerror("EC_KEY_check_key");
        }

        BIO_printf(bio_stderr,"Membuat
digest.\n");
        EVP_MD_CTX_init(&md_ctx);
        EVP_DigestInit(&md_ctx,
EVP_ecdsa());

        while(!feof(stdin)) {
                inlen = BIO_read(bio_stdin,
inbuf, 80);

        if(!EVP_DigestUpdate(&md_ctx,
inbuf, inlen)) {

        fatalerror("EVP_DigestUpdate");
                }
        }
        EVP_DigestFinal(&md_ctx, digest,
&dgst_len);

        //bagian ini yang melakukan
verifikasi tanda tangan
        ret = ECDSA_do_verify(digest,
dgst_len, signature, key);
        if (ret == -1){
                printf("Error:
ECDSA_do_verify\n");
        }else if (ret == 0){
                printf("\nTanda tangan tidak
valid.\n");
        }else{
                printf("\nTanda tangan
valid.\n");
        }

        return 0;
}
```

4. Analisis Aplikasi ECC pada File Save Game Nintendo Wii

Seperti yang telah disebutkan di bagian sebelumnya, sistem file *save game* pada Nintendo Wii menggunakan ECC NIST B 233 bit, atau yang biasa disebut sect233r1 pada OpenSSL. Kunci privat yang digunakan berukuran 233 bit dan dipasangkan dengan kunci publik berukuran 2 x 233 bit.

File *save game* Nintendo Wii diakhiri dengan sertifikat untuk menentukan kevalidan suatu file. Sertifikat ini terdiri dari pasangan kunci publik dan tanda tangan. Sertifikat ini disimpan dalam bentuk data 60 bit, di mana 30 bit pertama untuk elemen pertama, dan 30 bit berikutnya untuk elemen kedua. Sertifikat ini dapat dicek kevalidannya dengan menggunakan fungsi `EC_KEY_check_key` pada OpenSSL.

Algoritma ECC adalah algoritma kriptografi yang kuat untuk melakukan penandatanganan file dan pengenkripsian file, karena melibatkan operasi titik, garis, dan kurva, yang sulit untuk dipecahkan baik secara *brute force* maupun dengan menggunakan teknik serangan lainnya. Selain itu, ukuran kunci ECC relatif lebih kecil daripada teknik kriptografi kunci publik yang lainnya, seperti RSA. Kunci berukuran 160 bit pada ECC telah terbukti lebih aman daripada kunci berukuran 1024 bit pada RSA. Namun, algoritma ini tergolong algoritma kriptografi yang kompleksitas komputasinya cukup tinggi karena melibatkan banyak operasi titik kordinat, garis, dan kurva.

5. Kesimpulan

Kesimpulan yang diperoleh dari analisis dan implementasi *Elliptic Curve Cryptography* dan aplikasinya pada file *save game* Nintendo Wii adalah sebagai berikut :

1. Algoritma ECC adalah algoritma kriptografi kunci publik yang aman karena melibatkan operasi titik, garis, dan kurva
2. Algoritma ECC memiliki kunci yang berukuran kecil dan terbukti lebih aman daripada algoritma RSA yang menggunakan kunci yang jauh lebih besar sekalipun
3. Algoritma ECC merupakan algoritma kriptografi yang cukup tinggi kompleksitasnya, karena melibatkan operasi titik, garis, dan kurva

4. Nintendo Wii, salah satu mesin *video game console* yang paling populer saat ini, menggunakan ECC sebagai teknik untuk sistem file *save game*-nya

DAFTAR PUSTAKA

[1] Anoop MS. *Elliptic Curve Cryptography : An Implementation Tutorial*. Tanggal akses : 6 Mei 2009.

[2] Nintendo-Scene Forums. *http://www.nintendo-scene.com*. Tanggal akses : 6 Mei 2009.

[3] OpenSSL HomePage. http://www.openssl.org. Tanggal akses : 21 Mei 2009.

[4] National Security Agency, Central Security Service. http://www.nsa.gov/index.shtml. Tanggal akses : 6 Mei 2009