Analisis Keamanan *Key Agreement Protocol* pada *Elliptic Curves Criptography*

Neni Adiningsih – NIM: 13503007

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail: if13007@students.if.itb.ac.id

Abstrak

Protokol *authenticated key agreement* mempunyai arti penting dalam membangun keamanan dalam komunikasi dua atau lebih entity melalui internet. Pada makalah akan dibahas mengenai *secure protocol* yang digunakan pada *authenticated key agreement* yang berbasis pada Diffie-Hellman *key agreement*, secure protocol yang lebih sederhana daripada protocol sebelumnya dan multiple key agreement protocol yang dapat digunakan untuk menggunakan dua atau lebih kunci secara bersama (*share-key*) pada satu saat proses eksekusi protokol.

Protokol *key agreement* Diffie-Hellman merupakan sebuah metode yang cukup terkenal yaitu mengenai penggunaan kunci secara bersama oleh dua atau lebih entitas pada alat komunikasi publik. Dari segi komputasi, protokol dibutuhkan untuk mengeksekusi fungsi eksponensial dalam suatu kelompok dari kedua pasangan pengirim dan penerima. Nilai exponent yang dipilih saling bebas untuk setiap user. Pada lingkungan dengan *resource* yang terbatas seperti *smartcard* komputasi yang dibutuhkan sangat besar.

Pada makalah ini akan dibahas mengenai satu variant dari Diffie-Hellman, yang berdasarkan pada spesifikasi draft ANSI X9.42 yang dikembangkan oleh ANSI X9F1 working group dan protokol key agreement terotentikasi yang berdasarkan Diffie-Hellman yang bekerja pada *elliptic curve group*. Asumsi yang berlaku pada protokol yang dibahas adalah permasalahan diskrit logaritma *elliptic curve* adalah *secure*.

***Kata kunci**: Diffie-Hellman, protocol, authenticated key agreement, elliptic curves, share-key.*

1. Pendahuluan

Protokol adalah aturan yang berisi rangkaian langkah-langkah, yang melibatkan dua atau lebih orang, yang dibuat untuk menyelesaikan suatu kegiatan.

Protokol kriptografi merupakan protokol yang menggunakan kriptografi. Beberapa kegunaan protokol kriptografi diantaranya:

1. berbagi komponen rahasia untuk menghitung sebuah nilai,
2. membangkitkan rangkaian bilangan acak,
3. meyakinkan identitas orang lainnya (otentikasi), dll

Protokol kriptografi dibangun dengan melibatkan beberapa algoritma kriptografi. Sebagian besar protokol kriptografi dirancang untuk dipakai oleh kelompok yang terdiri dari 2 orang pemakai, tetapi ada juga beberapa protokol yang dirancang untuk dipakai oleh kelompok yang terdiri dari lebih dari dua orang pemakai (misalnya pada aplikasi *teleconferencing*)

Dalam mendeskripsikan protokol kriptografi dalam makalah ini akan digunakan nama-nama sebagai berikut:

Alice	:	Orang pertama
Bob	:	Orang kedua
Carol	:	Orang ketiga dalam protokol 3/3 orang
Dave	:	Orang keempat dalam protokol 4

orang
Eve : Penyadap (eavesdropper)
Trent : Juru penengah (arbirator) yang dipercaya

Nama-nama di atas tidak hanya menyatakan orang tetapi juga merepresentasikan sebuah mesin, misalnya *computer client*, *computer server*, program penyadap atau mesin yang melakukan intersepsi.

1.1 Protokol Komunikasi dengan Sistem Kriptografi Simetri

Protokol 1:
1. Alice dan Bob menyepakati algoritma kriptografi simetri yang akan digunakan
2. Alice dan Bob menyepakati kunci yang akan digunakan
3. Alice menulis pesan plainteks dan mengenkripsinya dengan kunci menjadi cipherteks
4. Alice mengirim pesan cipherteks kepada Bob
5. Bob mendeskripsi pesan cipherteks dengan kunci yang sama dan membaca plainteksnya

Eve mendengar semua percakapan antara Alice dan Bob pada protokol ini. Jika Eve menyadap transmisi pesan pada langkah (4), ia harus mencoba mengkriptanalis cipherteks untuk memperoleh plainteks tanpa mengetahui kunci. Jika ia mendengar pembicaraan pada langkah 1 dan 2, maka ia mengetahui algoritma dan kunci yang digunakan, sehingga ia dapat mendeskripsi cipherteks dengan kunci tersebut.

Protokol kriptografi di atas tidak bagus karena kunci harus tetap rahasia sebelumnya, sepanjang, dan setelah protokol. Langkah 1 dapat dilakukan dalam mode publik, namun langkah 2 harus dilakukan dalam mode rahasia. Sistem kriptografi kunci-publik dapat memecahkan masalah distribusi kunci ini.

1.2 Protokol Komunikasi dengan Sistem Kriptografi Kunci-Publik

Protokol 2:
1. Alice dan Bob menyepakati algoritma kriptografi kunci-publik yang akan digunakan
2. Bob mengirimi Alice kunci publiknya (kunci publik Bob)
3. Alice mengenkripsi pesannya dengan kunci publik Bob kemudian mengirimkannya ke Bob
4. Bob mendeskripsi pesan dari Alice dengan kunci privat miliknya (kunci privat Bob)

Pada umumnya, pengguna di jaringan menyepakati algoritma kriptografi kunci-publik yang digunakan. Setiap pengguna jaringan mempunyai kunci publik dan kunci privat, yang dalam hal ini kunci publik dipublikasikan melalui basisdata yang dapat diakses bersama. Dengan demikian, protokol kriptografi kunci publik menjadi lebih sederhana sebagai berikut:

Protokol 3:
1. Alice mengambil kunci publik Bob dari basisdata kunci publik
2. Alice mengenkripsi pesannya dengan kunci publik Bob kemudian mengirimkannya kepada Bob
3. Bob mendeskripsi pesan dari Alice dengan kunci privat miliknya (kunci privat Bob)

Eve yang mendengar pembicaraan selama protokol ini akan mendapatkan kunci publik Bob, tetapi Eve tidak dapat mendeskripsi cipherteks karena ia tidak mengetahui kunci privat Bob.

Dalam dunia nyata, sistem kriptografi kunci-publik bukanlah pengganti sistem kriptografi simetri. Sistem kriptografi kunci-publik tidak digunakan untuk mengenkripsi pesan, melainkan untuk mengenkripsi kuni pada sistem kriptografi simetri. Dengan sistem kriptografi kunci-publik, maka pertukaran kunci pada sistem kriptografi simetri dapat dilakukan dengan protokol kriptografi kunci-publik sebagai berikut:

Protokol 4:
1. Bob mengirimi Alice kunci publiknya
2. Alice membangkitkan kunci simetri K, mengenkripsikannya dengan kunci publik Bob, lalu mengirimkannya ke Bob,

$$E_B(K)$$

3. Bob mendeskripsi pesan dari Alice dengan menggunakan kunci privatnya untuk mendapatkan kembali kunci simetri K,

$$D_B(E_B(K)) = K$$

4. Baik Alice dan Bob dapat saling berkirim pesan dengan sistem kriptografi simetri dengan menggunakan kunci K.

Dua gabungan sistem kriptografi yang digunakan pada protokol 4 di atas disebut hybrid cryptosystem dan kunci simetri yang dipertukarkan disebut session key. Dengan protokol 4 di atas, kita katakan bahwa sistem kriptografi kunci-publik berhasil memecahkan masalah manajemen kunci yang sangat penting, yaitu pertukaran kunci.

1.3 Protokol untuk Tanda Tangan Digital

1.3.1 Menandatangani Dokumen dengan Sistem Kriptografi Simetri dan Seorang Juru Penengah

Alice ingin menandatangani dokumen digital (pesan atau arsip) dan mengirimkannya ke Bob. Ia meminta Trent sebgai juru penengah (misalnya pengacara) antara Alice dan Bob (diperlukan jika sewaktu-waktu ada pertengkaran antara Alice dan Bob). Trent akan memberikan tanda-tangan berupa sertifikasi terhadap dokumen yang dikirim oleh Alice. Sistem kriptografi yang digunakan adalah simetri. Trent memberikan kunci rahasia K_A kepada Alice dan kunci rahasia K_B kepada Bob (K_A dan K_B berbeda).

Protokol 5:
1. Alice mengenkripsi dokumen dengan KA dan mengirimkannya kepada Trent

2. Trent mendeskripsi dokumen dari Alice dengan KA

3. Trent menambahkan pada dokumen yang sudah dideskripsi sebuah pertanyaan sertifikasi bahwa dia telah menerima dokumen itu dari Alice, kemudian mengenkripsi keseluruhannya dengan KB.

4. Trent mengirim cipherteks yang dihasilkan kepada Bob.

5. Bob mendeskripsi cipherteks dengan KB. Ia membaca dokumen dan sertifikasi dari Trent bahwa Alice yang mengirimkan dokumen tersebut.

Karakteristik pemberian tanda-tangan dengan protokol 5 adalah sebagai berikut:
1. Tanda tangan (*signature*) pasti otentik, karena Trent adalah juru penengah yang dipercaya, Trent mengetahui bahwa dokumen dari Alice. Sertifikasi dari Trent berlaku sebagai bukti bagi Bob.

2. Tanda tangan tidak dapat digunakan lagi untuk dokumen yang lain. Jika Bob menggunakan sertifikasi dari Trent untuk dokumen yang lain, maka kecurangan Bob ini dapat diketahui oleh Trent sebagai berikut:
   a. Trent meminta dokumen tersebut dari Bob
   b. Trent mengenkripsi dokumen tersebut dengan KA dan membandingkannya dengan cipherteks dari Alice
   c. Jika hasil enkripsi dokumen dari Bob tidak sama dengan cipherteks dari Alice, maka Bob telah melakukan kecurangan

3. Dokumen yang sudah ditandatangani tidak dapat diubah. Trent dapat membuktikan bahwa dokumen sudah berubah dengan cara yang sama seperti no.2 di atas.

4. Tanda-tangan tidak dapat disangkal. Jika Alice menyangkal bahwa dia yang mengirim dokumen, sertifikasi dari Trent dapat menyanggah sangkalan Alice.

Protokol 5 di atas tidak praktis karena membutuhkan pihak ketiga (Trent) untuk memberikan sertifikasi keabsahan dokumen dan prosesnya memakan waktu.

1.3.2 Menandatangani Dokumen dengan Sistem Kriptografi Kunci-Publik

Protokol 6:
1. Alice mengenkripsi dokumen dengan kunci privatnya. Ini sekaligus juga berarti Alice telah memberikan tanda-tangan (signature) pada dokumennya

2. Alice mengirim dokumen yang terenkripsi kepada Bob

3. Bob mendeskripsi dokumen dengan kunci publik Alice. Ini sekaligus juga berarti Bob telah memverifikasi tanda-tangan pada dokumen.

Protokol 6 tidak membutuhkan pihak ketiga (Trent) untuk memberikan tandatangan (Trent hanya diperlukan untuk mensertifikasi bahwa kunci publik Alice memang benar milik Alice). Protokol 6 memiliki karakteristik yang sama seperti prda protocol 5.

1.3.3 Menandatangani Dokumen dengan Sistem Kriptografi Kunci-Publik dan Fungsi Hash Satu-Arah

Protokol 7:
1. Alice meringkas dokumennya menjadi message digest dengan fungsi hash satu-arah

2. Arah mengenkripsi messae digest dengan kunci privatnya. Hasil enkripsinya disertakan (embedded) pada dokumen. Ini berarti Alice tealh memberi tanda-tangan digital pada dokumennya.

3. Alice mengirim dokumen yang sudah diberi tanda-tangan digital kepada Bob

4. Bob meringkas dokumen dari Alice menjadi message digest dengan fungsi hash yang sama. Bob mendeskripsi tanda-tangn digital yang disertakan pada dokumen Alice. Jika hasil deskripsinya sama dengan message digest yang dihasilkan, maka tanda-tangan digital tersebut sah.

Jika dokumen yang sama ingin ditandatangani oleh dua orang (Alice dan Bob), maka orang Carol, dibutuhkan pada proses verifikasi. Protokolnya adalah sebagai berikut:

Protokol 8:
1. Alice memebri tanda-tangan digital pada message digest dari dokumen

2. Bob memebri tanda-tangan digital pada message digest dari dokumen

3. Bob mengirimkan tanda-tangan digitalnya kapada Alice

4. Alice mengirim dokumen yang sudah diberi tanda-tangan digital dari Bob kepada Carol

5. Carol memverifikasi tanda-tangan digital Alice dan tanda-tangan digital Bob (carol mengetahui kunci publik Alice dan kunci publik Bob)

1.4 Protokol untuk Tanda Tangan Digital dengan Enkripsi

Protokol ini dapat dianalogikan seperti pengiriman surat yang menggunakan amplop tertutup. Tanda tangan pada surat memberikan bukti kepemilikan, hal ini sama dengan fungsi tanda-tangan digital pada dikumen elektronis. Sedangkan amplop memberikan perlindungan keamanan (privacy), hal ini sama dengan fungsi enkripsi pada dokumen. Tanda tangan digital diberikan dengan menggunakan kunci privat pengirim dan dokumen dienkripsi dengan kunci public penerima.

Protokol 9:
1. Alice menandatangani dokumen atau pesan (M) dengan menggunakan kunci privat (A)
$$S_A(M)$$

2. Alice mengenkripsi dokumen yang sudah ditandatangani dengan kunci publik Bob (B) dan mengirimkannya kepada Bob
$$E_B(S_A(M))$$

3. Bob mendeskripsi cipherteks yang diterima dengan kunci privatnya
$$D_B(E_B(S_A(M))) = S_A(M)$$

4. Bob melakukan verifikasi dengan mendeskripsi hasil pada langkah 3 dengan menggunakan kunci publik Alice dan sekaligus mendapatkan kemb0ali dokumen yang belum dienkripsi.
$$V_A(S_A(M)) = M$$

Menandatangani dokumen sebelum mengenkripsikannya adalah cara yang alamiah. Dalam kehidupan sehari-hari, kita menulis surat, menandatanganinya, dan memasukkannya ke dalam amplop. Bila Alice memasukkan surat ke dalam amplop, kemudian menandatangani amplop, maka keabsahannya diragukan. Jika Bob memperlihatkan surat Alice tersebut kepada Carol, maka Carol mungkin menuduh Bob berbohong tentang isi surat tersebut.

Alice tidak harus menggunakan kunci publik/kunci privat yang sama untuk enkripsi dan tanda tangan. Alice dapat menggunakan dua pasang kunci: sepasang untuk enkripsi dan sepasang untuk pemberian tanda tangan.

Misalkan Bob ingin mengkonfirmasi bahwa dia telah menerima dokumen dari Alice. Maka, Bob mengirimkan konfirmasi ‘tanda terima’ kepada Alice. Protokol pengiriman pesan tanda terima sebagai berikut:

Protokol 10:
1. Alice menandatangani dokumen atau pesan (M) dengan menggunakan kunci privatnya, mengenkripsikannya dengan kunci publik Bob dan mengirimkannya kepada Bob
$$E_K(S_A(M))$$

2. Bob mendeskripsi cipherteks yang diterima dengan kunci privatnya (B), memverifikasi tanda tangan digital dengan kunci publik Alice dan sekaligus mendapatkan kembali dokumen yang belum dienkripsi
$$V_A(D_B(E_B(S_A(M)))) = M$$

3. Bob menandatangani dokumen (M) dengan kunci privatnya mengenkripsikannya dengan kunci public Alice, dan mengirimkannya ke Alice.
$$E_A(S_B(M))$$

4. Alice mendeskripsi dokumen dengan kunci privatnya dan memverifikasi tanda-tangan digital dengan kunci publik Bob.

$$V_B(D_A(E_A(S_B(M)))) = M'$$

Jika M' yang dihasilkan sama dengan dokumen yang dikirim oleh Alice (M), maka Alice tahu bahwa Bob menerima dokumennya dengan benar.

1.5 Petukaran Kunci

Seperti yang sudah disebutkan pada bagian sebelum ini, *session key* adalah kunci simetri yang digunakan untuk mengenkripsi pesan selama berkomunikasi saja. Protocol 4 menyebut bahwa Alice (atau Bob) mengirimkan kunci publiknya terlebih dahulu sebelum mengenkripsi session key. Dalam praktek, kunci public disimpan di dalam basisdata. Hal ini membuat pertukaran kunci menjadi lebih mudah dengan protocol berikut:

Protokol 11:

1. Alice mengambil kunci publik Bob dari basisdata

2. Alice membangkitkan session key K, mengenkripsikannya dengan kunci public (PK) Bob, dan mengirimkannya ke Bob,
$$E_{PK}(K)$$

3. Bob mendeskripsi pesan dari Alice dengan menggunakan kunci rahasianya (SK) untuk mendapatkan kembali session key K,
$$D_{SK}(E_PK(K)) = K$$

4. Baik Alice dan Bob dapat saling berkirim pesan dengan sistem kriptografi simetri dengan menggunakan kunci K.

Pertukaran kunci dan pengiriman psan dapat dilakukan bersamaan. Jadi, Alice dan Bob tidak perlu menyelesaikan protocol pertukaran kunci sebelum bertukar pesan.

Protokol 12:

1. Alice membangkitkan session key K, mengenkripsi pesan M dengan menggunakan K,
$$E_K(M)$$

2. Alice mengambil kunci publik Bob dari basisdata

3. Alice mengenkripsi K dengan kunci publik Bob,
$$E_B(K)$$

4. Alice mengirim pesan terenkripsi bersama-sama dengan kunci terenkripsi kepada Bob,
$$E_K(M), E_B(K)$$

5. Bob mendeskripsi menggunakan kunci privatnya untuk mendapatkan kembali *session key* K,
$$D_B(E_B(K)) = K$$

6. Bob mendeskripsi pesan dengan menggunakan kunci K,
$$D_K(E_K(M)) = M$$

Jika Alice ingin mengirim pesannya tidak hanya kepada Bob, tetapi juga kepada Carol dan Dave maka protokol pertukaran kunci dan pengiriman pesan dilakukan secara *broadcast* dengan protokol berikut:

Protokol 12:

1. Alice membangkitkan session key K, mengenkripsi pesan M dengan menggunakan K,
$$E_K(M)$$

2. Alice mengambil kunci publik Bob, Carel dan Dave dari basisdata

3. Alice mengenkripsi K dengan kunci publik Bob, Carel, dan Dave,
$$E_B(K)$$

4. Alice mengirim pesan terenkripsi bersama-sama dengan kunci terenkripsi kepada Bob, Carel dan Dave,
$$E_K(M), E_B(K) , E_C(K) , E_D(K)$$

5. Hanya Bob, Carol dan Dave yang dapat mendeskripsi kunci K dengan menggunakan kunci privatnya masing-masing.

6. Hanya Bob, Carol dan Dave yang dapat mendeskripsi pesan dengan menggunakan kunci K.

Protokol 12 di atas diimplementasikan pada jaringan *store-and-forward*. Dalam hal ini, server memforwardkan pesan terenkripsi dan kunci terenkripsi dari Alice kepada Bob, carol, dan Dave.

1.6 Otentikasi

1.6.1 Otentikasi dengan menggunakan sandi-lewat dengan fungsi hash satu-arah

Misalkan Alice *log on* ke komputer host (misal *automatic teller machine*). Bagaimana host tahu bahwa yang masuk adalah Alice? Secara tradisional, password digunakan untuk otentikasi. Host tidak perlu menyimpan password, ia hanya perlu menyimpan nilai hash dari password tersebut yaitu dengan fungsi hash satu arah. Protokol otentikasinya adalah sebagai berikut:

Protokol 15:

1. Alice mengirim password ke host
2. Host mengkompresi password dengan fungsi hash satu arah
3. Host mengkompresi hasil dari fungsi hash dengan nilai hash yang disimpan sebelumnya di dalam tabel basisdata.

Kelemahan otentikasi dengan protocol 15 ini adalah rentan terhadap serangan dictionary attack. Misalkan Mallory (seorang penyerang aktif yang sangat dengki) berhasil meng-hack komputer host dan mencuri tabel data *password* yang sudah dikompres dengan fungsi hash satu arah. Selanjutnya Mallory menggunakan kamus yang berisi 1.000.000 password yng sangat umum dipakai orang (nama jalan, tanggal kelahiran, nama anak, dsb). Ia mengkompres seluruh entry di dalam kamus dengan fungsi hash satu-arah dan menyimpan hasilnya. Kemudian ia membandingkan tabel data *password* yang dicuri dari host dengan hasil hash terhadap isi kamus dan melihat kecocokannya.

Untuk membuat dictionary attack lebih sulit, sistem kemanan computer biasanya menambahkan garam (*salt*). *Salt* adalah rangkaian bit yang dibangkitkan secara acak dan disambungkan dengan password. Kemudian *password* yang sudah disambung dengan salt dikompres dengan fungsi hash dan hasilnya disimpan di dalam tabel. Semakin panjang s*alt* semakin bagus. Sistem UNIX menggunakan salt 12-bit.

1.6.2 Otentikasi dengan menggunakan sistem kriptografi kunci publik

Host menyimpan tabel yang berisi kunci publik semua pengguna. Setiap pengguna memiliki kunci rahasia yang bersesuaian dengan kunci publiknya. Protokol otentikasinya adalah sebagai berikut:

Protokol 16:

1. Host mengirimi Alice sebuah string acak
2. Alice mengenkripsi string dengan kunci rahasianya dan mengirimkan kembali ke host beserta *user-id*-nya
3. Host mencari kunci publik Alice berdasarkan user-id yang diberikan dan mendeskripsi cipherteks dari Alice dengan kunci publik tersebut
4. Jika hasil deskripsi sama dengan string yang semula dikirim oleh host, maka host mengizinkan Alice mengakses system.

2. Key Agreement Protocol

Pada protokol *key agreement* dua atau lebih entitas yang terdistribusi menggunakan satu kunci rahasia secara bersama (*share-key*) yang disebut dengan *session key*. Kunci ini dapat digunakan untuk membentuk saluran komunikasi konfidensial antar entitas. Beberapa protokol *key agreement* yang berbasiskan Diffie-Hellman telah di*propose* dalam beberapa tahun ini, tetapi beberapa protokol tersebut masih mempunyai beberapa kekurangan. Jumlah atribut yang diperlukan pada protokol *key agreement* telah teridentifikasi dan banyak dari protokol-protokol tersebut yang dianalisa menggunakan beberapa atribut tersebut.

Beberapa atribut tersebut digunakan pada beberapa protokol di bawah ini, yaitu:

1. Known-key Security
   Protokol *key agreement* pada masing-masing entitas harus menghasilkan kunci rahasia yang unik. Protokol tersebut harus tetap dapat mencapai *goal* di hadapan lawan yang mempelajari *session key* yang lain.
2. (Perfect) forward secrecy
   Jika kunci privat long-term dari satu atau lebih entitas sama (*compromised*) maka kerahasian dari session key sebelumnya yang dibangun oleh entitas-entitas yang *honest* tidak akan terpengaruhi.
3. Key-compromise impersonation
   Asumsikan bahwa kunci long-term pada A *disclose* (diperlihatkan). Oleh sebab itu, lawan dapat mengetahui nilai kunci tersebut dengan berpura-pura menjadi A (*impersonate* A). Akan tetapi dengan kekurangan ini tidak memungkin bagi lawan untuk dapat berpura-pura menjadi entitas selain A terhadap A.

4. Unknown key-share
   Entitas A tidak dapat dipaksa untuk menggunakan kunci secara bersama dengan entitas B tanpa adanya pengetahuan dari A. Misal, ketika A percaya bahwa kunci tersebut di-*share* dengan entitas C ≠ B dan B percaya bahwa kunci telah di-*share* dengan A.

5. Key control
   Tidak satupun protocol yang terlibat harus dapat memilih nilai awal

Tahap pertama pada proses key agreement adalah perhitungan angka rahasia, sebut saja dengan ZZ. ZZ dapat dihasilkan jika pasangan kunci publik/privat dari originator dan recipient digunakan. Nilai ZZ yang dihasilkan kemudian dikonversi menjadi kunci kriptografi simetri. Pada saat *originator* menjalankan pasangan kunci privat/publik yang statis, maka pengantar nilai publik yang random tersebut memastikan bahwa hasil kunci simetri yang dihasilkan akan selalu berbeda untuk setiap *key agreement*.

2.1 Pembangkitan ZZ

Spesifikasi X9.42 mendefinisikan bahwa nilai rahasia ZZ yang di-*share* dihasilkan dari persamaan berikut:

$$ZZ = g \wedge (xb * xa) \bmod p$$

Pada individual parties menggunakan perhitungan sebagai berikut:

$$ZZ = (yb \wedge xa) \bmod p = (ya \wedge xb) \bmod p$$

dimana,
^ mendefinisikan fungsi eksponensial.
ya merupakan kunci publik untuk A

$$ya = g \wedge xa \bmod p$$

yb merupakan kunci publik untuk B

$$yb = g \wedge xb \bmod p$$

xa merupakan kunci privat untuk A
xb merupakan kunci privat untuk B
p, q adalah bilangan prima yang sangat besar

$$g = h^{(p-1)/q} \bmod p$$

dimana, h adalah sembarang integer dengan range

$$1 < h < p-1$$

sehingga $h^{(p-1)/q} \bmod p > 1$, $g \equiv q \bmod p$; contoh: g^q mod p = 1 jika g!=1

j adalah nilai integer yang sangat besar dengan p=qj + 1.

2.2 Pembangkitan Material Kunci

X9.42 meneydiakan sebuah algoritma yang dapat digunakan untuk menghaislkan sejumlah material kunci yang esensial dari ZZ. Algoritma tersebut diturunkan dari algoritma di bawah ini, yaitu dengan menambahkan atau mengurangi parameter opsional.

$$KM = H (ZZ \,||\, OtherInfo)$$

H merupakan fungsi hash untuk message digest pada SHA-1. ZZ merupakan nilai rahasia yang di-*share* yang dihasilkan dari perhitungan pada 2.1. Nilai ZZ harus memenuhi octet p, sehingga harus diawali oleh 0 untuk setiap nilai yang kurang dari 8. misal jika p adalah 1024 bit, maka panjangnya ZZ harus 128 bytes.

OtherInfo merupakan DER encoding dengan struktur seperti di bawah ini:

```
OtherInfo ::= SEQUENCE {
    keyInfo KeySpecificInfo,
    partyAInfo [0] OCTET STRING OPTIONAL,
    suppPubInfo [2] OCTET STRING
}
```

```
KeySpecificInfo ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    counter OCTET STRING SIZE (4..4)
}
```

Definisi pada ASN.1 menggunakan tangging EXPLICIT.

Pada struktur di atas,
algorithm adalah algoritma ASN.1 OID dari algoritma wrapping CEK yang menggunakan KEK. Algoritma ini bukan sebuah AlgorithmIdentifier, tetapi hanya merupakan OBJECT IDENTIFIER. Tidak ada parameter yang digunakan.

counter merupakan angka 32 bit, yang direpresentasikan pada network byte order. Nilai awal ZZ adalah 1, maka byte untuk nilai ini adalah 00 00 00 01 (hex), dan nilai ini akan bertambah satu untuk setiap kali fungsi yagn menghasilkan KEK tersebut dijalankan.

partyAInfo adalah string random yang diberikan oleh sender. Pada CMS, partyAInfo merupakan sebuah parameter pada UserKeyingMaterial field (dienkodekan menjadi sebuah OCTET STRING). partyAInfo MUST harus terdiri dari 512 bit.

suppPubInfo merupakan panjang KEK, dalam bit, direpresntasikan sebagai angka 32 bit number pada network byte order. Misal, untuk 3DES maka nilai KEK adalah 00 00 00 C0.

Untuk menghasilkan sebuah KEK, satu akan menghasilkan satu atau lebih block KM sampai jumlah material yang dihasilkan cukup.
Block KM dikonkatenasi dari kiri ke kanan. Contoh: KM (counter=1) || KM (counter=2)...

Walaupun string yang lebih panjang dari ZZ dihasilkan, kunci yang efektif dari KEK dibatasi oleh ukuran ZZ dan beberapa tingkatan pertimbangan keamanan ditentukan oleh nilai p dan q. Akan tetapi, jika partyAInfo berbeda untuk setiap pesan, maka akan dihasilkan KEK yang berbeda pula untuk setiap pesan. partyAInfo harus digunakan pada mode Static-Static tetapi mungkin akan muncul pada mode Ephemeral-Static.

2.3 KEK Computation

Setiap algoritma kunci enkripsi membutuhkan ukuran kunci yang fix (n). KEK dihasilkan dari mapping n-byte yang paling kiri dari KM pada kunci. 3 DES membutuhkan 192 bit material kunci, oelh karena itu algoritma harus dijalankan dua kali, pertama dengan nilai counter sama dengan 1 (untuk menghasilkan K1', K2' dan 32 bit pertama dari K3'). K1', K2' dan K3' kemudian distandartkan untuk menghasilkan kunci 3 DES yaitu K1, K2 dan K3.

RC2-128 membutuhkan 128 bit material kunci pada saat algoritma dijalankan satu kali dengan nilai couter sama dengan 1 dan bit yang paling kiri dari 128 bit tersebut dikonversi secara langsung menjadi sebuah kunci RC2. Hal ini berlaku juga untuk RC2-40 yang membutuhkan 40 bit material kunci pada saat algoritma dijalankan satu kali dengan nilai counter sama dengan 1 dan bit yang paling dari 40 bit tersebut digunakan sebagai kunci.

2.4 Panjang Kunci untuk Algoritma yang Umum digunakan

Beberapa algoritma kunci enkripsi yang umum mempunyai panjang KEK tertentu. Berkut beberapa algoritma tersebut:

3-key 3DES	192 bits
RC2-128	128 bits
RC2-40	40 bits

Panjang kunci yang paling efektif untuk RC2 adalah sama dengan panjang kunci RC2 yang sebenarnya.

2.5 Validasi Kunci Publik

Algoritma di bawah ini dapat digunakan untuk melakukan validasi pada saat menerima kunci publik y.

1. Lakukan verifikasi terhadap nilai y, yaitu memeriksa range dari nilai tersebut. y harus berada pada interval [2,p-1]. Jika hal ini tidak dipenuhi maka kunci tidak valid.

2. Hitung nilai y^q mod p.
   Jika hasilnya sama dengan 1, maka kunci valid. Begitu sebaliknya, jika tidak maka kunci tidak valid.

Tujuan utama dari validasi kunci publik adalah untuk mencegah small-subgroup attack pada pengirim pasangan kunci. Jika digunakan mode Ephemeral-Static maka pengecekan/validasi yang dilakukan tidak akan berpengaruh. Prosedur di atas digunakan pada pending patents.

2.6 Contoh

2.6.1 Contoh 1

ZZ terdiri dari 20 *bytes*, yaitu:
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

Algoritma *key wrap* yang digunakan adalah 3DES-EDE *wrap*.

Tidak ada partyAInfo yang digunakan.
Oleh karena itu, maka dihasilkan input pada *invocation* pertama pada SHA-1 adalah:

// ZZ
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

// 3DES wrap OID
30 1d 30 13 06 0b 2a 86 48 86 f7 0d 01 09 10 03 06

//Counter
04 04 00 00 00 01

//key length
a2 06 04 04 00 00 00 c0

Maka dihasilkan 20 bytes dari byte di atas yaitu:

a0 96 61 39 23 76 f7 04 4d 90 52 a3 97 88 32 46 b6 7f 5f 1e

Sehingga input pada *invocation* kedua pada SHA-1 adalah:

//ZZ
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

//3DES wrap OID
30 1d 30 13 06 0b 2a 86 48 86 f7 0d 01 09 10 03 06

//Counter
04 04 00 00 00 02

//panjang kunci
a2 06 04 04 00 00 00 c0

Maka dihasilkan 20 bytes dari byte di atas yaitu:
f6 3e b5 fb 5f 56 d9 b6 a8 34 03 91 c2 d3 45 34 93 2e 11 30

sehingga didapatkan nilai,
K1' = a0 96 61 39 23 76 f7 04
K2' = 4d 90 52 a3 97 88 32 46
K3' = b6 7f 5f 1e f6 3e b5 fb

Nilai-nilai K_i' tersebut belum distandartkan.

2.6.2 Contoh 2

ZZ terdiri dari 20 *bytes*, yaitu:
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

Algoritma *key wrap* yang digunakan adalah RC2-128 *key wrap*, sehingga dibutuhkan 128 bits (16 *bytes*) material kunci.

partyAInfo yang digunakan adalah 64 bytes,
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01

Oleh karena itu, maka dihasilkan input pada SHA-1 adalah:

//ZZ
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

//RC2 wrap OID
30 61 30 13 06 0b 2a 86 48 86 f7 0d 01 09 10 03 07

//Counter
04 04 00 00 00 01

//partyAInfo
a0 42 04 40 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01

//panjang kunci
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01 a2 06 04 04 00 00 00 80

Maka dihasilkan 20 *bytes* dari *byte* di atas yaitu:
48 95 0c 46 e0 53 00 75 40 3c ce 72 88 96 04 e0 3e 7b 5d e9

sehingga didapatkan nilai,
K = 48 95 0c 46 e0 53 00 75 40 3c ce 72 88 96 04 e0

2.7 Requirement terhadap Kunci dan Parameter

X9.42 membutuhkan sekumpulan parameter yang merupakan bentukan dari persamaan p=jq + 1.

Dimana, q adalah bilangan prima yang besar dengan panjang m dan j adalah integer dengan nilai j>=2.

Algoritma yang digunakan untuk menghasilkan bilangan prima yang besar tersebut adalah algoritma pada FIPS.

X9.42 membutuhkan kunci privat yang berada pada interval [2, (q-2)]. Nilai x dibangkitkan secara random dan harus berada pada interval tersebut. Sedangkan nilai y dihasilkan dengan menghitung g^x mod p. Untuk memenuhi ketentuan-ketentuan di atas maka panjang m harus >= 160 bit (konsekuensinya maka nilai q paling sedikit harus kurang dari 160 bit).

Jika cipher simetri yang lebih kuat daripada DES digunakan, maka harus digunakan nilai m yang lebih panjang. Dan nilai p minimum adalah 512 bit.

2.7.1 Pembangkitan Parameter

Agen seharusnya menghasilkan parameter domain (g, p, q) menggunakan algoritma yang diturunkan dari [FIPS-186] dan [X942]. Jika algoritma ini digunakan maka kebenaran dari hasil pembangkitan dapat diverifikasi oleh pihak ketiga pada algoritma yang terdapat pada bab 2.5.

2.7.1.1 Pembangkitan nilai p, q

Algoritma di bawah ini digunakan untuk membangkitkan pasangan nilai p, q dimana q adalah panjang m dan p adalah panjang L.

1. *Set* m' = m/160 dimana / merepresentasikan pembagian bilangan integer dengan pembulatan ke atas.
   contoh. 200/160 = 2.

2. *Set* L' = L/160

3. *Set* N' = L/1024

4. pilih sebuah bit string *SEED* dengan panjang >= m

5. *Set* U = 0

6. For i = 0 to m' – 1 {
   U = U + (SHA1[SEED + i] XOR SHA1[(SEED + m' + i)) * 2^(160 * i)
   }

   Untuk =160, hal ini akan mengurangi nilai, U = SHA1[SEED] XOR SHA1[(SEED+1) mod 2^160] pada algoritma [FIPS-186].

7. Nilai q didapatkan dari U dengan menghitung U mod (2^m) dan mengatur bit yang paling signifikan (yaitu 2^(m-1) bit) dan bit yang paling tidak signifikan menjadi 1.

   Pada operasi boolean, q = U OR 2^(m-1) OR 1. Dimana, 2^(m-1) < q < 2^m.

8. Gunakan algoritma yang robust untuk mengetes apakah q adalah bilangan prima.

9. Jika q bukan bilangan prima maka menuju langkah 4.
10. Isikan counter = 0

11. *Set* R = *seed* + 2*m' + (L' * *counter*)

12. *Set* V = 0

13. For i = 0 to L'-1 do
    V = V + SHA1(R + i) * 2^(160 * i)

14. *Set* W = V mod 2^L

15. *Set* X = W OR 2^(L-1)
    Dimana, 0 <= W < 2^(L-1) dan X >= 2^(L-1)

16. *Set* p = X - (X mod (2*q)) + 1

17. Jika p > 2^(L-1) gunakan test yang untuk mengetest apakah p adalah bilangan prima. Jika tidak maka lakukan 19.

18. Jika p adalah bilangan prima maka output p, q, *seed*, *counter* dan *stop*.

19. *Set counter* = *counter* + 1

20. Jika *counter* < (4096 * N) maka lakukan 8.

21. Output "*failure*"

Sebuah test dikatakan robust jika besar probabilitas dari bilangan non-prima yang ada paling besar 2^-80. Algoritma ini terdapat pada [FIPS-186] dan [X942].

2.7.1.2 Pembangkitan g

Berikut adalah algoritma yang diturunkan dari [FIPS-186] yang digunakan untuk membangkitkan g.

1. *Assign* nilai j = (p - 1)/q.

2. Set h = integer berapapun, dimana h berada dalam interval 1 < h < p - 1 dan h selalu berbeda dengan nilai-nilai yang diberikan sebelumnya.

3. *Set* g = h^j mod p

4. Jika g = 1 lakukan langkah 2

2.7.2 Validasi Parameter

ASN.1 untuk kunci DH pada [PKIX] meliputi elemen j dan *validation-Parms* yang mungkin dapat digunakan oleh pihak yang menerima (recipients) kunci untuk memverifikasi kevalidan parameter yang dibangkitkan.

Dua pengecekan yang mungkin dilakukan:
1. Memverifikasi p = qj + 1. Hal ini dilakukan untuk memenuhi kriteria parameter X9.42.

2. Melakukan verifikasi saat pembangkitan p,q oleh prosedur dalam [FIPS-186].

Hal ini menunjukkan bahwa parameter yang digunakan dipilih secara random dan tidak mempunyai bentuk tertentu.

Informasi validasi terhadap sertifiaksi pada agen merupakan kejadian lokal yang terjadi antara agen dan CA.

2.8 Ephemeral-Static Mode

Pada mode Ephemeral-Static, penerima (*recipient*) mempunyai sepasang kunci statis (dan *certified*), tetapi pengirim (*sender*) membangkitkan pasangan kunci baru untuk setiap pesan dan mengirimkannya menggunakan *originatorKey production*.

Jika kunci untuk pengirim adalah hasil pembangkitan baru untuk setiap pesan, maka ZZ rahasia yang di-*share* juga akan berbeda untuk setiap pesan dan partyAInfo mungkin dapat dihilangkan karena partyAInfo hanya akan digunakan untuk memisahkan pasangan multiple KEK yang dibangkitkan dengan kunci sekumpulan pasangan yang sama.

Jika kunci *ephemeral* pengirim digunakan untuk multiple pesan (contoh, pesan di-*cache* sebagai optimasi performansi) kemudian partyAInfo yang terpisah harus digunakan untuk setiap pesan. Semua implementasi yang sesuai dengan standart ini harus diimplementasikan menggunakan mode Ephemeral-Static.

Untuk melawan small subgroup attacks, penerima harus melakukan pengecekan seperti yang telah dijelaskan pada bab 2.5. Jika tidak dapat menentukan apakah dia berhasil atau gagal dalam melakukan operasi deskripsi oleh penerima maka penerima seharusnya memilih untuk mengabaikan pengecekan tersebut.

2.9 Static-Static Mode

Pada mode Static-Static, baik pengirim maupun penerima mempunyai sepasang kunci statis (dan *certified*). Dikarenakan pengirim dan penerima mempunyai kunci yang sama untuk setiap pesan, maka ZZ juga akan sama untuk setiap pesan. Sehingga partyAInfo harus digunakan (berbeda untuk setiap pesan) untuk memastikan bahwa pesan yang berbeda menggunakan KEK yang berbeda pula. Semua ketentuan ini dapat diimplementasikan menggunaan mode Static-Static.

Untuk mencegah small subgroup attacks baik originator dan recipient, keduanya harus melakuan validasi yang dijelaskan pada bab 2.5 atau melakukan verifikasi yaitu bahwa CA telah benar-benar melakukan verifikasi terhadap kevalidan kunci.

3. Diffie-Hellman Key Agreement

Protokol key agreement Diffie-Hellman (exponensial key agreeement) dikembangkan oleh Diffie and Hellman pada tahun 1976 dan dipublikasikan pada paper yang dengan judul "*New Directions in Cryptography*". Protokol ini memungkinkan dua user untuk saling bertukar kunci rahasia melalui media yang 'kurang aman' tanpa memperhatikan prioritas kerahasiaan.

Seperti yang telah dijelaskan pada bab-bab sebelumnya, protokol ini juga mempunyai dua parameter utama yaitu p dan g. Keduanya bersifat publik dan dapat digunakan oleh semua user dalam satu sistem. Parameter p merupakan bilangan prima dan parameter g (generator) adalah sebuah nilai integer yang kurang dari p ($g < p$) dengan properti sebagai berikut:

for $1<n< p-1$ {
terdapat g^k yang memenuhi
$n = g^k \bmod p$.
}

Sebagai contoh, misalkan Alice dan Bob telah menyetujui untuk men-*share* satu kunci rahasia menggunakan protokol *key agreement* Diffie-Hellman.

Beberapa proses yang berlangsung adalah:

1. Alice membangkitkan nilai privat A secara random dan Bob membangkitkan nilai privat B secara random. A dan B adalah sebuah integer.

2. Membangkitkan kunci publik masing-masing dari kunci privat yang telah dibangkitkan sebelumnya.

   Kunci publik milik Alice adalah $g^a \bmod p$, dan kunci publik Bob adalah $g^b \bmod p$.
   Kemudian keduanya saling menukar kunci publik masing-masing.

3. Alice menghitung $g^{ab} = (g^b)^a \bmod p$ dan Bob menghitung $g^{ba} = (g^a)^b \bmod p$

Karena nilai $g^{ab} = g^{ba} = k$, maka dapat dikatakan bahwa Alice dan Bob telah men-share kunci rahasia k.

Diffie-Hellman membuat algoritma pertukaran kunci yang keamanannya didasarkan pada fakta bahwa menghitung logaritma diskrit sangat sulit. Mula-mula Alice dan Bob menyepakati bilangan prima yang besar, n dan g, sedemikian sehingga $g < n$. Bilangan n dan g tidak perlu rahasia. Bahkan Alice dan Bob dapat membicarakannya melalui saluran yang tidak aman sekalipun. Protokol pertukaran kunci Diffie-Hellman dinyatakan dalam protokol 13 berikut:

Protokol 13:

1. Alice memilih bilangan bulat acak yang besar x dan mengirim hasil perhitungan berikut kepada Bob:
$$X = g^x \bmod n$$

2. Bob memilih bilangan bulat acak yang besar y dan mengirim hasil perhitungan berikut kepada Alice:
$$Y = g^y \bmod n$$

3. Alice menghitung
$$K = Y^x \bmod n$$

4. Bob menghitung
$$K' = X^y \bmod n$$

Jika perhitungan dilakukan dengan benar, maka K=K'. Baik K dan K' sama dengan g^{xy} mod n. Eve yang mendengarkan semua hal selama protokol berlangsung tidak dapat menghitung kunci K. Ia hanya memiliki informasi n,g,X dan Y, tetapi ia tidak mempunyai informasi nilai x dan y. Untuk mengetahui x atau y, ia perlu melakukan perhitungan logaritma diskrit, yang mana sangat sulit dikerjaan.

Varian dari algoritma Diffie-hellman dikemukakan oleh Hughes sebagai berikut:

Protokol 14:
1. Alice memilih bilangan bulat acak yang besar x dan menghitung:
$$K = g^x \bmod n$$

2. Bob memilih bilangan bulat acak yang besar y dan mengirim hasil perhitungan berikut kepada Alice:
$$Y = g^y \bmod n$$

3. Alice mengirim hasil perhitungan berikut kepada Bob
$$X = Y^x \bmod n$$

4. Bob menghitung $Z = y^{-1}$ (balikan y dalam modulo n) $K' = X^z \bmod n$

Jika perhitungan dilakukan dengan benar, maka K = K'. Keuntungan dari protocol ini, Alice dapat langsung mendapatkan kunci rahasia K sebelum interaksi dengan Bob. Alice dapat mengenkripsi pesannya kepada Bob sebelum protokol pertukaran kunci selesai.

Diffie and Hellman menggambarkan rata-rata dari dua pihak yang saling menyetujui untuk *shared* rahasia secara bersama-sama sehingga rahasia tersebut tidak dapat disadap oleh pihak lain. Rahasia tersebut kemudian dikonversi dalam bentuk kunci kriptografi pada algoritma simetris lainnya.

Key agreement Diffie-Hellman membutuhkan kedua pihak baik pengirim maupun penerima message mempunyai kunci yang berpasangan. Angka rahasia yang dapat digunakan bersama oleh pihak satu dengan pihak kedua dapat dibangkitkan dengan mengkombinasikan private key pihak satu dengan public key pihak yang lain. Angka rahasia ini kemudian dapat dikonversi menjadi material kunci kriptografi. Material kunci ini akan digunakan sebagai kunci enkripsi, *key-encryption key* (KEK), untuk mengenkripsi kunci enkripsi untuk isi, *content-encryption key* (CEK), yang akan digunakan untuk mengenkripsi pesan.

Pertukaran kunci pada Diffie-Hellman dapat diserang dengan man-in-middle attack. Pada serangan ini, Carol sebagai lawan mengintersepsi nilai publik milik Alice dan mengirimkan nilai publik dirinya pada Bob. Pada saat Bob mengtransmisikan nilai publiknya, Carol mengganti nilai tersebut dengan miliknya dan mengirimkannya kepada Alice. Oleh karena itu, maka Carol dan Alice mempunyai kunci yang di-share bersama dan begitu juga dengan Carol dan Bob. Setelah melakukan pertukaran kunci, maka Carol dapat dengan mudah mendeskripsi pesan-pesan yang dikirimkan oleh Alice maupun Bob sehingga Carol dapat membaca, memodifikasi isi dari pesan tersebut dan dapat mengirimkannya ke pihak yang lain sebelum dienkripsi kembali oleh kunci yang berkesesuaian. Hal ini dapat terjadi karena pada Diffie-Hellman tidak ada proses otentikasi pada *participant*. Solusi yang mungkin untuk mengatasi hal ini adalah dengan menggunakan *digital signature* dan protokol dengan variant yang lain.

Pada tahun 1992 dikembangkan protokol Diffie-Hellman yang sudah memiliki otentikasi untuk setiap participantnya sehingga permasalahan *man-in-the-middle attack* dapat teratasi. Kekebalan protokol ini adalah dengan memberikan kesempatan bagi dua pihak untuk mengotentikasi diri mereka sendiri pada pihak yang lain dengan menggunakan digital signature dan sertifikasi kunci publik.

4. *Key agreement* dengan *Elliptic Curves*

Beberapa peneliti melakukan pemerikasaan terhadap kripto sistem *elliptic curve* yang pertama kali dilakukan oleh Miller [MIL86] dan Koblitz [KOB87]. Kripto sistem elliptic curve yang berdasarkan pada masalah logaritma *elliptic curve* diskrit pada *finite field* memberikan beberapa keuntungan untuk sistem lain.

Panjang kunci dapat menjadi lebih pendek pada skema yang lain karena hanya eksponensial *attack* yang dapat mengetahui secara *curve* yang dipilih dan *elliptic curve* pada logaritma dikrit tetap *robust* meskipun pemfaktoran dan perkalian pada logaritma diskrit dapat dengan mudah dilakukan.

Definisi 1. Anggap E adalah sebuah *elliptic curve* yang didefenisikan pada *finite field* Fq dan anggap P Є E(Fq) merupakan sebuah titik dari order n. Dan Q Є E(Fq), maka masalah yang muncul pada logaritma diskrit *elliptic curve* adalah untuk menemukan nilai integer l, $0 \le l \le (n-1)$, dan $Q = l \cdot P$.

4.1 Pembangkitan Kunci

Pada makalah ini digunakan *elliptic curve* E yang didefinisikan pada *finite field* F_q dari p karakteristik.

Langkah pertama yang dilakukan adalah pemilihan terhadap parameter domain pada *elliptic curve*, yaitu:

1. q merupakan ukuran *field*, dimana q adalah hasil pangkat dari bilangan prima (pada prakteknya, q = p, merupakan bilangan prima ganjil atau $q = 2^m$)

2. Dua elemen field a, b Є F_q, dimana a dan b mendefinisikan persamaan dari *elliptic curve* E dari F_q
   Contoh:
   $y^2 = x^3 + ax + b$ dengan p > 3,
   dimana $4a^3 + 27b^2 \neq 0$

3. Dua elemen field x_p dan y_p pada F_q, dimana x_p dan y_p digunakan untuk mendefinisikan *finite point* $P = (x_p, y_p)$ dari order bilangan prima pada $E(F_q)$
   $P \neq O$, dimana O menunjukkan sebuah point yang *infinite*.

4. *order* n dari *point* P

Parameter domain pada elliptic curve dapat diverifikasi untuk memenuhi kebutuhan [LAW03].

Operasi pembangkitan kunci adalah sebagai berikut:

1. Memilih fungsi hash H,misalnya SHA-1.

2. Memilih nilai integer secara random s_A, s_B dengan batasan interval [1, n − 1].
   Nilai s_A adalah kunci rahasia untuk user A dan s_A adalah kunci rahasia untuk user B.

3. Menghitung point $y_A = -s_A \cdot P$ and $y_B = -s_B \cdot P$, dimana y_A adalah kunci publik untuk user A dan y_B adalah kunci publik untuk masing-masing user B.

4. Memilih ID_A dan ID_B, dimana ID_A adalah informasi identitas user A dan ID_B adalah informasi identitas user B.

4.2 Deskripsi Protokol

Protokol key agreement yang terotentikasi (authenticated key agreement protocol/AKAP) diantara Alice dan Bob adalah sebagai berikut:

1. Alice membangkitkan nilai integer secara random r_A, k_A (kunci *ephemeral*) yang berada pada interval [1, n-1] dan menghitung Q_A, V_A maka point pada E adalah:

   $$Q_A = r_A \cdot P, \quad V_A = -k_A \cdot P$$

   Alice mengirimkan V_A pada Bob.

2. Bob secara random memilih nilai integer r_B, k_{B1} , ..., k_{Bn} dari interval [1, n-1] dan menghitung Q_B, V_B, dimana i=1,..., n sehingga

   $$Q_B = r_B \cdot P, \qquad V_{Bi} = -k_{Bi} \cdot P$$

   Bob menghitung:

   $$e_B = H\left(x_{Q_B}, x_{V_{B_1}}, ..., x_{V_{B_n}}, x_{V_{A_1}}, ..., x_{V_{A_n}}, ID_B, ID_A\right)$$

   $$d_B = r_B + e_B \sum_{i=1}^{n} k_{B_i} + e_B s_B$$

   dimana x_{QB} merupakan koordinat dari Q_B, x_{VBi} adalah sumbu koordinat-x dari V_B dan x_{VAi} adalah sumbu koordinat-y dari V_A. dimana i = 1, …, n.

   Bob mengirim V_B, dimana i = 1, …, n, e_B, d_B kepada Alice.

3. Alice menghitung point U_B, dimana

   $$U_B = d_B \cdot P + e_B \sum_{i=1}^{n} \cdot V_{B_i} + e_B \cdot Y_B$$

   dan mengecek kevalidan e_B, dimana

   $$e_B = H\left(x_{U_B}, x_{V_{B_1}}, ..., x_{V_{B_n}}, x_{V_{A_1}}, ..., x_{V_{A_n}}, ID_B, ID_A\right)$$

   Jika ketentuan tersebut tidak dipenuhi maka Alice akan memberhentikan eksekusi. Jika tidak maka Alice akan menghitung:

   $$e_A = H\left(x_{Q_A}, x_{V_{A_1}}, ..., x_{V_{A_n}}, x_{V_{B_1}}, ..., x_{V_{B_n}}, ID_A, ID_B\right)$$

   $$d_A = r_A + e_A \sum_{i=1}^{n} k_{A_i} + e_A s_A$$

   dimana x_{QA} merupakan koordinat dari Q_A, x_{VAi} adalah sumbu koordinat-x dari V_A dan x_{VBi}

adalah sumbu koordinat-y dari V_B. dimana i = 1, …, n. Alice kemudian menghitung point pada K_A dengan persamaan:

$$K_A = -k_A \cdot V_B$$

dan kemudian mengirimkan e_A, d_A kepada Bob

4. Bob menghitung point U_A, dimana

$$U_A = d_A \cdot P + e_A \sum_{i=1}^{n} V_{A_i} + e_A \cdot Y_A$$

dan mengecek kevalidan e_A, dimana

$$e_A = H\left(x_{U_A}, x_{V_{A_1}}, ..., x_{V_{A_n}}, x_{V_{B_1}}, ..., x_{V_{B_n}}, ID_A, ID_B\right)$$

Jika ketentuan tersebut tidak dipenuhi maka Bob akan menghentikan eksekusi. Jika tidak maka Bob akan menghitung point pada K_{Bi} dengan persamaan:

$$K_B = -k_B \cdot V_A$$

Kunci rahasia yang di-share adalah point K = K_A = K_B.

5. Protokol Multiple *Key Agreement*

Pada protokol multiple key agreement setiap participant dapat men-*share* dua atau lebih kunci rahasia pada satu kali proses eksekusi protokol. Pembangkitan kunci sama dengan protokol key agreement lainnya.

Berikut adalah gambaran mengenai protokol multiple key agreement antara Alice dan Bob.

1. Alice membangkitkan nilai integer yang random yaitu r_A, k_{A1} , ..., k_{An} dengan interval [1, n − 1] dan menghitung point Q_A, V_{Ai} , dimana i = 1, ..., n, sehingga

$$Q_A = r_A \cdot P, \quad V_{Ai} = -k_{Ai} \cdot P$$

Alice mengirimkan point V_{Ai}, dimana i = 1, ..., n pada Bob.

2. Bob secara random memilih nilai integer r_B, k_{B1} , ..., k_{Bn} dari interval [1, n-1] dan menghitung Q_B, V_B, dimana i=1,..., n sehingga

$$Q_B = r_B \cdot P, \quad V_{Bi} = -k_{Bi} \cdot P$$

Bob menghitung:

$$e_B = H\left(x_{Q_B}, x_{V_{B_1}}, ..., x_{V_{B_n}}, x_{V_{A_1}}, ..., x_{V_{A_n}}, ID_B, ID_A\right)$$

$$d_B = r_B + e_B \sum_{i=1}^{n} k_{B_i} + e_B s_B$$

dimana x_{QB} merupakan koordinat dari Q_B, x_{VBi} adalah sumbu koordinat-x dari V_B dan x_{VAi} adalah sumbu koordinat-y dari V_A. dimana i = 1, …, n.

Bob mengirim V_B, dimana i = 1, …, n, e_B, d_B kepada Alice.

3. Alice menghitung point U_B, dimana

$$U_B = d_B \cdot P + e_B \sum_{i=1}^{n} \cdot V_{B_i} + e_B \cdot Y_B$$

dan mengecek kevalidan e_B, dimana

$$e_B = H\left(x_{U_B}, x_{V_{B_1}}, ..., x_{V_{B_n}}, x_{V_{A_1}}, ..., x_{V_{A_n}}, ID_B, ID_A\right)$$

Jika ketentuan tersebut tidak dipenuhi maka Alice akan memberhentikan eksekusi. Jika tidak maka Alice akan menghitung:

$$e_A = H\left(x_{Q_A}, x_{V_{A_1}}, ..., x_{V_{A_n}}, x_{V_{B_1}}, ..., x_{V_{B_n}}, ID_A, ID_B\right)$$

$$d_A = r_A + e_A \sum_{i=1}^{n} k_{A_i} + e_A s_A$$

dimana x_{QA} merupakan koordinat dari Q_A, x_{VAi} adalah sumbu koordinat-x dari V_A dan x_{VBi} adalah sumbu koordinat-y dari V_B. dimana i = 1, …, n. Alice kemudian menghitung point pada K_{Ai} dengan persamaan:

$$K_{A_i} = -k_{A_i} \cdot V_{B_i}, \quad i = 1, ..., n$$

dan kemudian mengirimkan e_A, d_A kepada B.

4. Bob menghitung point U_A, dimana

$$U_A = d_A \cdot P + e_A \sum_{i=1}^{n} V_{A_i} + e_A \cdot Y_A$$

dan mengecek kevalidan e_A, dimana

$$e_A = H\left(x_{U_A}, x_{V_{A_1}}, ..., x_{V_{A_n}}, x_{V_{B_1}}, ..., x_{V_{B_n}}, ID_A, ID_B\right)$$

dimana x_{UA} merupakan koordinat dari U_A, x_{VAi} adalah sumbu koordinat-x dari V_A dan x_{VBi} adalah sumbu koordinat-y dari V_B. dimana i = 1, …, n. Jika ketentuan tersebut tidak dipenuhi maka Bob akan menghentikan eksekusi. Jika tidak maka Bob akan menghitung point pada K_{Bi} dengan persamaan:

$$K_{B_i} = -k_{B_i} \cdot V_{A_i}, \quad i = 1, ..., n$$

Maka kunci rahasia yang di-share adalah point K_i = K_{Ai} = K_{Bi} dimana i =1, …, n.

6. Analisis

Pada makalah ini protokol yang digunakan menggunakan asumsi bahwa masalah logaritma diskrit *elliptic curve*.

Beberapa atribut yang digunakan pada protokol yang diajukan beserta penjelasan tingkat keamanan , yaitu:

1. Known-key Security
   Jika dua entitas A dan B mengeksekusi regular protocol yang sedang berjalan, maka sudah jelas bahwa keduanya pasti men-*share session key*.

2. (Perfect) forward secrecy
   Selama perhitungan session key K untuk setiap entity maka nilai integer random yang dihasilkan oleh masing-masing user, yaitu r_A, k_A, r_B, k_B akan tetap berlaku.

   Sebuah lawan yang bisa mendapatkan kunci privat user yaitu s_A or s_B harus melakukan ekstraksi nilai integer random (kunci ephemeral) r_A, k_A, r_B, k_B dari Q_A, V_A, Q_B, V_B untuk mengetahui session key sebelumnya muapun sesudahnya. Akan tetapi hal ini merupakan permasalahan logaritma diskrit elliptic curve.

3. Key-compromise impersonation
   Asumsikan bahwa kunci privat long-term S_A pada user A *disclose* (diperlihatkan). Maka sudah jelas bahwa lawan yang mengetahui nilai tersebut dapat berpura-pura menjadi A (*impersonate* A). Dan dapat berpura-pura menjadi B karena juga mengetahui kunci privat milik B.

   Untuk dapat sukses dalam melakukan misi tersebut maka pihak lawan harus mengetahui kunci *ephemeral* A, r_A dan k_A. Nilai r_A dan k_A dapat didapatkan dengan melakukan ekstraksi terhadap nilai *ephemeral* publik, Q_A dan V_A untuk dapat membangkitkan *session key* yang sama dengan A. Hal ini juga merupakan permasalahan logaritma diskrit *elliptic curve*.

4. Unknown key-share
   Anggap lawan C mencoba membuat A percaya bahwa session key telah di-share dengan B, akan tetapi B percaya bahwa session key di-share dengan C.

   Untuk membuat *unknown key-share attack*, C harus meng-*set* kunci publiknya menjadi tersertifikasi walaupun dia tidak mengetahui kunci privat yang benar. Pada hal ini, C menggunakan nilai point publik YA, YB dan P.
   Misalkan,

$$f_t(R_1, ..., R_l) = \sum_{i=1}^{l} t_i R_i$$

dimana R_i adalah point pada E dan t = (t1, ..., tl) adalah integer yang berada pada interval [1, n−1].

Kemudian C harus meng-set kunci publik miliknya YC menjadi,

$$Y_C = f_t(Y_A, Y_B, P)$$

Anggap C mendapatkan nilai YC yang tersertifikasi sebagai kunci publiknya dan anggap bahwa semuanya ini dapat mengeneralisasi model untuk *unknown key-share attack*.

Anggap bahwa VC adalah

$$V_C = f_p(Y_A, Y_B, P, V_B)$$

dan V'C adalah

$$V_C' = f_m(Y_A, Y_B, P, V_A)$$

dimana p = (p1, ..., pl) dan m = (m1, ...,ml) adalah nilai integer yang berada pada interval [1, n − 1].

Attack yang dilakukan C dapat berhasil dilakukan dengan memaksa A dan B untuk men-*share session key* $K = K_A = K_B$ selama protokol sedang berjalan. Pada prakteknya A dan B mendapatkan *session key* untuk mereka K_A dan K_B sesuai dengan persamaan berikut ini:

$$K_A = -k_A \cdot V_B, \quad K_B = -k_B \cdot V_C'$$

C tidak mengetahui sA, sB, kA, kB walaupun C dapat mengontrol nilai integer t_i, p_i, m_i. C dapat memaksakan sebuah persamaan $K_A = K_B$ untuk memegang nilai k_A dan k_B. Oleh karena itu persamaan di atas dapat diturunkan menjadi persamaan berikut:

$$k_A \cdot V_B = k_B \cdot V_C'$$

Persamaan di atas dapat diubah menjadi $a \cdot P = O$, dengan membuka nilai V_A, Y_C, V_C, V'_C yang tergantung terhadap P. dan persamaan ini tidak perlu diselesaikan untuk setiap ti, pi,mi, jika tidak ada informasi yang cukup mengenai s_A, s_B, k_A, k_B.

5. Key control
   Key-control adalah tidak mungkin untuk *third party*. Satu *key control attack* yang paling mungkin adalah oleh participant dari protokol B. party B harus membuat party A untuk membangkitkan *session key* KB yang telah dipilih sebelumnya oleh B, misal B harus menyelesaikan persamaan $K_B = -K_B . V_A$. Hal ini juga merupakan permasalahan logaritma diskrit *elliptic curve*.

Pada tabel 1 dituliskan perbandingan antara protokol yang dibahas dalam makalah dengan protokol lain. Kompleksitas diukur dengan menghitung jumlah komputasi yang dominan pada setiap party. Misalnya, eksponensial modular, perkalian *big integer* dengan *point* pada *elliptic curve*. Pada protokol yang dibahas dalam makalah ini setiap kunci ephemeral pada setiap *party* (*point* Q_A, V_A, Q_B, V_B) dibangkitkan pada fase prekomputasi.

Tabel 1 Perbandingan kompleksitas beberapa protokol

Protokol	Kompleksitas	Attack
AKAP	3	Tidak diketahui
SAKAP	1	Tidak diketahui
MQVKC	2.5	Tidak diketahui
MQV	2.5	UKA
ECDH	2	MMA
MTI/A0	3	UKA
MTI/C0	2	SSA
Joux	2	UKA
Smart	4	FSA

Berikut protokol yang akan dibandingkan pada tabel 1:

- MQV merupakan Protocol 1 of Law, Menezes, Qu, Solinas and Vanstone [LAW03].
- ECDH merupakan *elliptic curve* versi protokol Diffie-Hellman
- MQVKC merupakan protokol MQV dengan konfirmasi kunci.
- MTI/A0, MTI/C0 merupakan protokol yang diajukan oleh Matsumoto, Takashima dan Imai.
- Joux merupakan protokol dari Joux.
- Smart adalah protokol dari Smart.
- UKA-Unknown Key-share Attack
- MMA-Man in the Middle Attack
- SSA-Small Subgroup Attack.
- FSA-Forward Secrecy Attack

Pada ECDH yang merupakan *elliptic curve* versi protokol Diffie-Hellman membutuhkan 2 nilai integer multiplikasi pada setiap entitas dan hal ini tidak aman untuk *man-in-the-middle attack*. Protokol MTI/A0 dan MTI/C0 membutuhkan 3 dan 2 integer multiplikasi. Protokol MTI/A0 dapat diserang oleh *unknown key-share attack* dan MTI/C0 tidak aman dalam melawan *subgroup attack*. Begitu juga protokol Joux membutuhkan 1 integer multiplikasi dam 1 operasi pasangan Weil. Protokol Joux merupakan protokol yang tidak terotentikasi dan dapat terserang oleh *man-in-the-middle-attack*. Protokol Smart membutuhkan 2 nilai integer multiplikasi dan 2 pasang operasi Weil.

Protokol MVQ dengan konfirmasi kunci (MQVKC) resisten terhadap unknown key-share attacks. Protokol AKAP menyediakan semua atribut keamanan sebaik protokol MVQ dengan konfirmasi kunci.

Protokol MQV dengan konfirmasi kunci membutuhkan 2.5 modular yang eksponensial untuk setiap entitas sedangkan protokol AKAP membutuhkan 3 integer multiplikasi. Karena protokol MQV dengan konfirmasi kunci membutuhkan tambahan 1 message authentication code algorithm (MAC), dengan hal ini efisiensi protokol AKAP dapat bertindak sama dengan protokol MQVKC. Protokol AKAP dibentuk untuk memenuhi atribut keamanan yang tidak terdapat pada protokol MQV, ECDH, MTI/A0, MTI/C0, Joux dan Smart. Begitu protokol SAKAP membutuhkan hanya 1 multiplikasi integer untuk setiap entitas. Sehingga protokol SAKAP (dapat diganti dengan protokol AKAP) lebih efisien bila dibandingkan dengan protokol dengan konfirmasi kunci.

7. Kesimpulan dan Saran

Beberapa kesimpulan yang dapat ditarik dari pembahasan di atas, diantaranya:

1. Semua keamanan yang ada pada sistem yang dibahas dalam makalah ini ditentukan oleh kerahasiaan material kunci privat. Jika kunci privat pengirim atau penerima telah diketahui, maka semua pesan yang dikirim maupun yang diterima menggunakan kunci tersebut juga diketahui.
   Dan jika kunci privat penerima atau pengirim hilang (lupa) maka pengirim atau penerima tidak akan bisa membaca pesan yang telah dikirim menggunakan kunci tersebut.

2. Kunci statis Diffie-Hellman mudah dipecahkan oleh small subgroup attack [LAW98]. Pada prakteknya hal ini muncul pada kedua mode yang ditawarkan yaitu mode Static-Static dan untuk penerima selama menggunakan mode Ephemeral-Static

3. Untuk mengatasi kemungkinan adanya serangan, dimungkinkan dibangkitkan kunci yang resisten terhadap serangan [LIM97]

4. Untuk mengatasi *man-in-the-middle attack* pada Diffie-Hellman adalah dengan menggunakan *digital signature* dan protokol dengan *variant* yang lain.

5. Tingkat keamanan pada metode yang dibahas dalam makalah ini sangat tergantung oleh beberapa faktor.

   Faktor-faktor tersebut diantaranya panjang dari kunci simetri (misal, 2^l tingkat keamanan jika panjangnya l bits); ukuran/besar bilangan prima q (tingkatan keamanan 2^{m/2}); dan ukuran bilangan prima p (tingkatan keamanan meningkat sesuai dengan fungsi subexponential berdasarkan ukurannya dalam bit).

6. Jika kebanyakan kunci diturunkan dari pasangan bilangan prima p dan q, maka harus hati-hati untuk mendapatkan level yang lebih tinggi untuk bilangan prima tersebut. Pada beberapa kasus seluruh keamanan dibatasi oleh tiga level terendah.

7. Pada makalah ini diusulkan protokol keamanan untuk protokol key agreement yang terotentikasi (AKAP) yang berdasarkan Diffie-Hellman yang bekerja pada elliptic curve group.

   Kekurangan yang ada pada protokol ini adalah setiap partisipan untuk membangkitkan 2 angka random yang berada pada interval $[1, n - 1]$ pada satu kali eksekusi.

   Kekurangan lainnya adalah bahwa kebutuhan akan entitas lebih rendah dibandingkan modular eksponensial atau multiplikasi integer daripada protokol yang lainnya. Pada makalah ini protokol yang digunakan menggunakan asumsi bahwa masalah logaritma diskrit elliptic curve adalah aman.

Saran yang dapat diberikan sesuai dengan pembahasan di atas adalah:

- Untuk mengurangi biaya komputasi dari dua persamaan eksponensial pada protokol key agreement Diffie-Hellman adalah dengan menggunakan kunci yang lebih panjang daripada K (kunci yang dibutuhkan dan sesuai dengan entropi) dengan mengambil bit dari distribusi binomial. Hal ini dapat menaikkan jumlah kuadrat dan mengurangi jumlah perkalian karena perhitungan nilai kuadrat lebih murah daripada perkalian.

8. Referensi

[GUE06] Shay Gueron, Zuk (2006). Exponentiation Speed Up For Diffie-Hellman Key Agreement Protocol. Dept. Of Mathematics, University of Haifa. Israel.

[KOB87] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation, 48, 1987, 203-209.

[LAW98] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An efficient protocol for authenticated key agreement", Technical report CORR 98-05, University of Waterloo, 1998.

[LAW03] L. Law, A. Menezes, M. Qu, J. Solinas, & S. Vanstone, An efficient Protocol for Authenticated Key Agreement, Designs, Codes and Cryptography, 28 (2), 2003, 119-134.

[LIM97] C.H. Lim and P.J. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup", B.S. Kaliski, Jr., editor, Advances in Cryptology – Crypto '97, Lecture Notes in Computer Science, vol. 1295, 1997, Springer-Verlag, pp. 249-263.

[MIL86] V. Miller, Uses of elliptic curves in cryptography, Proceedings of Crypto'85, Santa Barbara, USA, 1986, 417-426.

[MUN06] Rinaldi Munir. (2006). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[POP05] C.Popescu (2005), A Secure Key Agreement Protocol Using Elliptic Curves, Department of Mathematics, University of Oradea. Romania