Chương trình KC-01:
Nghiên cứu khoa học phát triển công nghệ thông tin và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

Báo cáo kết quả nghiên cứu

PHẦN MỀM CÓ SỬ DỤNG CHỨNG CHỈ SỐ

Quyển 8A: "Dùng chứng chỉ số với các dịch vụ Web và Mail"

HÀ NỘI-2004

Báo cáo kết quả nghiên cứu

PHẦN MỀM CÓ SỬ DỤNG CHỨNG CHỈ SỐ

Quyển 7A: "Dùng chứng chỉ số với các dịch vụ Web và Mail"

Chủ trì nhóm thực hiện:
PGS. TS. Lê Mỹ Tú

Mục lục

Chương I
Giao thức Secure Socket Layer

1. Giới thiệu

Secure Sockets Layer (SSL) là một giao thức có thể được đặt ở giữa giao thức tầng mạng kết nối định hướng tin tưởng (TCP/IP) và tầng giao thức ứng dụng (FTP, HTTP, telnet ...). SSL cung cấp dịch vụ truyền thông có bảo mật giữa client và server bằng việc cho phép client và server xác thực lẫn nhau sử dụng chữ ký số và bảo mật thông tin trao đổi qua lại bằng cách mã hóa các thông tin đó.

Giao thức này được thiết kế để có thể trợ giúp một loạt các thuật toán sử dụng cho việc mã hóa, hàm băm và chữ ký số. Giao thức SSL có ba phiên bản:

-SSLv2: đây là phiên bản đầu tiên của giao thức SSL do Netscape Corporation thiết kế, chưa có trợ giúp chain certificate.

-SSLv3: đây là phiên bản SSL version 3.0 do Netscape Corporation thiết kế, đã có trợ giúp chain certificate và được suport cho tất cả các trình duyệt phổ thông.

-TLSv1: đây là giao thức Transport Layer Security version 1.0, dựa trên cơ sở của SSLv3, được thiết kế bởi IETF (Internet Engineering Task Force) nhưng hiện nó chưa được support cho tất cả các trình duyệt thông dụng (chẳng hạn Netscape là không có giao thức này).

Chú ý:

-Một đặc điểm quan trọng của SSLv3 và TLSv1 là có trợ giúp việc nạp chuỗi các certificate (certificate chain). Với đặc điểm được bổ sung này sẽ cho phép server và client có thể thực hiện việc xác thực lẫn nhau mà có thể đối tượng thực hiện xác thực không cần phải cài các intermediate issuers.

-TLSv1 dựa trên nền tảng là SSLv3 trong đó có bổ sung phần block padding cho các thuật toán mã khối, chuẩn hoá thứ tự các message và bổ sung thêm các thông báo trong phiên liên lạc.

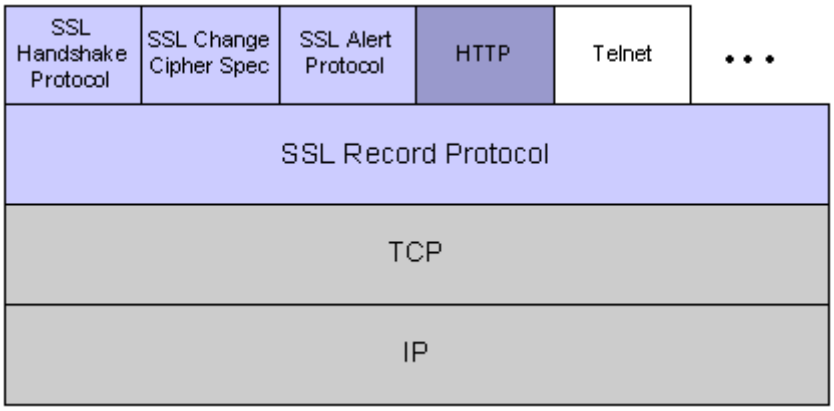
-Các phiên bản trên cũng như các thuật toán mã hoá, thuật toán trao đổi khoá, hàm băm hoàn toàn có thể được chỉ ra cụ thể khi thiết lập cấu hình sử dụng SSL cho Web server (Apache server), và một số trình duyệt (trong các trình duyệt phổ thông IE không có thuộc tính này).

Với nhu cầu thực tế hiện nay SSLv2 ít được sử dụng. Bên cạnh đó do có sự tương ứng giữa SSLv3 và TLSv1, hơn nữa hiện tại trong thực tế TLSv1 chưa được tích hợp cho một số trình duyệt phổ thông (Netscape chẳng hạn) nên trong phần này chúng tôi chỉ trình chi tiết về giao thức SSLv3 (đối với TLSv1 hoàn toàn tương tự).

2-Giao thức SSLv3

Giao thức SSLv3 gồm hai thành phần Handshake protocol và Record protocol. SSLv3 Record protocol cung cấp cơ chế bảo mật với các thuật toán mã hoá như DES, RC4,... và là giao thức kết nối tin tưởng với việc sử dụng hàm kiểm tra MAC trong quá trình trao đổi dữ liệu. Còn SSLv3 Handshake protocol thực hiện việc xác thực đối tác, trao đổi các giá trị secure sử dụng cho SSLv3 Record

protocol. Toàn bộ giao thức SSLv3 và mối liên hệ của nó với tầng ứng dụng và tầng TCP có thể mô tả như sơ đồ dưới đây:

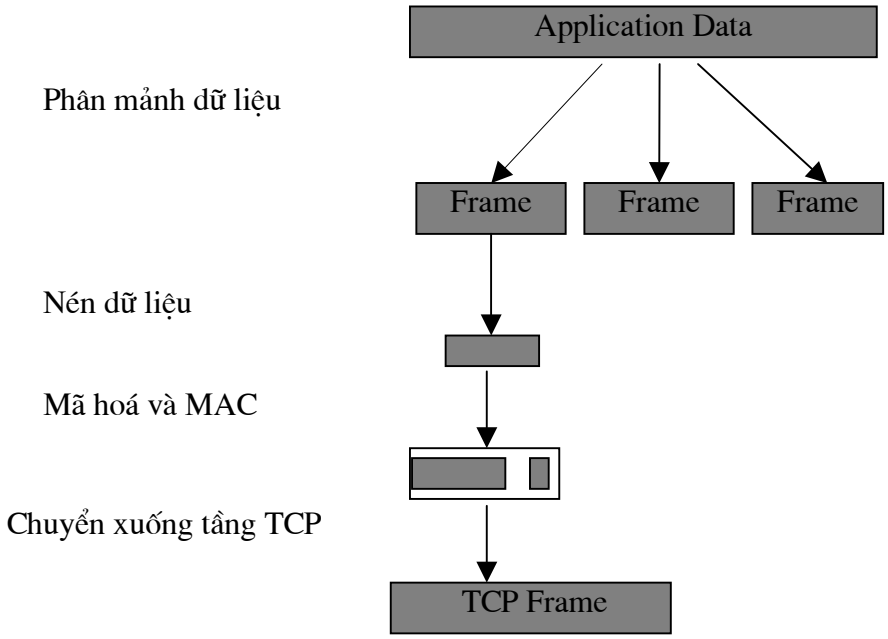


Hình 1.

Dưới đây chúng tôi sẽ trình bày tuần tự chi tiết các tiến trình được thực hiện khi sử dụng giao thức SSLv3.

2.1-Tầng giao thức SSLv3 Record

Giao thức SSLV3 Record là một tầng giao thức. Đối với mỗi tầng giao thức nói chung, một gói dữ liệu sẽ bao gồm các trường độ dài, mô tả và nội dung dữ liệu. SSLv3 Record nhận dữ liệu cần gửi từ tầng trên phân nhỏ thành từng block, nén dữ liệu, bổ sung dữ liệu kiểm tra, mã hoá và gửi. Khi nhận dữ liệu về tiến trình được thực hiện ngược lại: giải mã, kiểm tra, gỡ nén và sắp xếp lại rồi gửi lên tầng trên. Cụ thể có thể diễn giải các giai đoạn trong giao thức này như sau:



Hình 2.

Trong đó Application data có thể là dữ liệu của SSL handshake protocol, SSL change Cipher Spec, SSL Alert protocol hoặc dữ liệu của các ứng dụng khác như HTTP, Telnet, ... Để phân biệt được từng loại dữ liệu đó trong mỗi frame dữ liệu

của SSL record đều có phần header để phân biệt. Cụ thể mỗi frame dữ liêu có cấu truc như sau:

```
enum {
        change_cipher_spec(20), alert(21),
         handshake(22), application_data(23), (255)
}ContentType;
struct{
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[SSLPlaintext.length];
} SSLPlaintext;
```

Trong đó:

type chính là phần header chỉ ra loại dữ liệu gì.

version phiên bản SSL.

length độ dài dữ liệu thật theo byte (lớn nhất là 2^{14}-1).

fragment dữ liệu.

- ***Nén và gỡ nén dữ liệu:***

Sau khi nhận được dữ liệu từ tầng trên, giao thức SSL record sẽ thiết lập nên các frame dữ liệu có cấu trúc là các SSLPlaintext. Các frame này sẽ được thực hiện nén bằng thuật toán nén được thiết lập bởi handshake protocol tạo thành các frame dữ liệu tương ứng được goi là SSLCompressed có cấu trúc như sau:

```
struct{
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[SSLCmpressed.length];
} SSLCompressed;
```

Trong đó

type và version giữ nguyên từ SSLPlaintext.

length độ dài SSLCompressed.fragment theo byte, không quá 2^{14}-1 +1024 (tức là thuật toán nén không được làm tăng thêm độ dài của dữ liệu thật quá 1024 byte).

fragment dữ liệu nén.

Tương ứng khi gỡ nén dữ liệu nếu độ dài dữ liệu nhận được lơn hơn 2^{14}-1 bytes thì sẽ xuất hiện thông báo lỗi.

- ***Thực hiện mã hoá và MAC***

Để bảo vệ dữ liệu trên đường truyền giao thức SSL sử dụng thuật toán mã hoá và MAC được định nghiã trong CipherSpec hiện tại. Đối với mỗi phiên liên lạc sau khi thực hiện xong giai đoạn handshake hai bên sẽ thiết lập được thuật toán mã hoá chung, và tính được các thuộc tính cho hàm MAC. Thuật toán mã hoá và hàm MAC sẽ thực hiện biến đổi cấu trúc SSLCompressed thành SSLCiphertext. Khi nhận được SSLCiphertext quá trình giải mã sẽ thực hiện thao tác ngược lại. SSLCiphertext có cấu trúc như sau:

```
struct{
        ContentType type;
```

```
        ProtocolVersion version;
        uint16 length;
        select (CipherSpec.cipher_type){
            case stream: GenericStreamCipher;
            case block: GenericBlockCipher;
        }fragment;
    }SSLCiphertext;
```

Trong đó:

type, version giữ nguyên từ SSLCompressed.

length độ dài theo byte của SSLCiphertext.fragment (không quá 2^{14}+2048).

fragment dữ liệu sau khi mã hoá SSLCompressed.fragment, bao gồm cả MAC.

-Trường hợp không mã hoá hoặc dùng thuật toán mã dòng:

```
    stream-cipher struct{
        opaque content[SSLCompressed.length];
        opaque MAC[CipherSpec.hash_size];
    }GenericStreamCipher;
```

stream_cipher: tên thuật toán mã hoá.

Với MAC được sinh như sau:

```
    hash(MAC_write_secret+pad_2+
        hash(MAC_write_secret+pad_1+seq_num+
        SSLCompressed.type+SSLCompressed.length+
        SSLCompressed.fragment))
```

pad_1 giá trị 0x36 được lặp lại 48 lần với MD5, 40 lần với SHA.

pad_2 giá trị 0x5c được lặp lại 48 lần với MD5, 40 lần với SHA.

seq_num số thứ tự của frame đang xử lý.

hast tên thủ tục thực hiện hàm hash được định nghĩa trong CihperSpec.

Chú ý:

Hàm hash được thực hiện trước khi mã hoá tức là khi thủ tục mã hoá thực hiện dữ liệu đầu vào có cả kết quả của hàm hash.

-Trường hợp mã khối:

Với trường hợp mã khối cấu trúc của GenericBlockCipher như sau:

```
    block_cipher struct{
        opaque content[SSLCompressed.length];
        opaque MAC[CipherSpec.hash_size];
        uint8 padding[GenericBlockCipher.padding_length];
        uint8 padding_length;
    } GenericBlockCipher;
```

padding bổ sung để độ dài của plaintext chia hết cho độ dài của block mã khối được dùng.

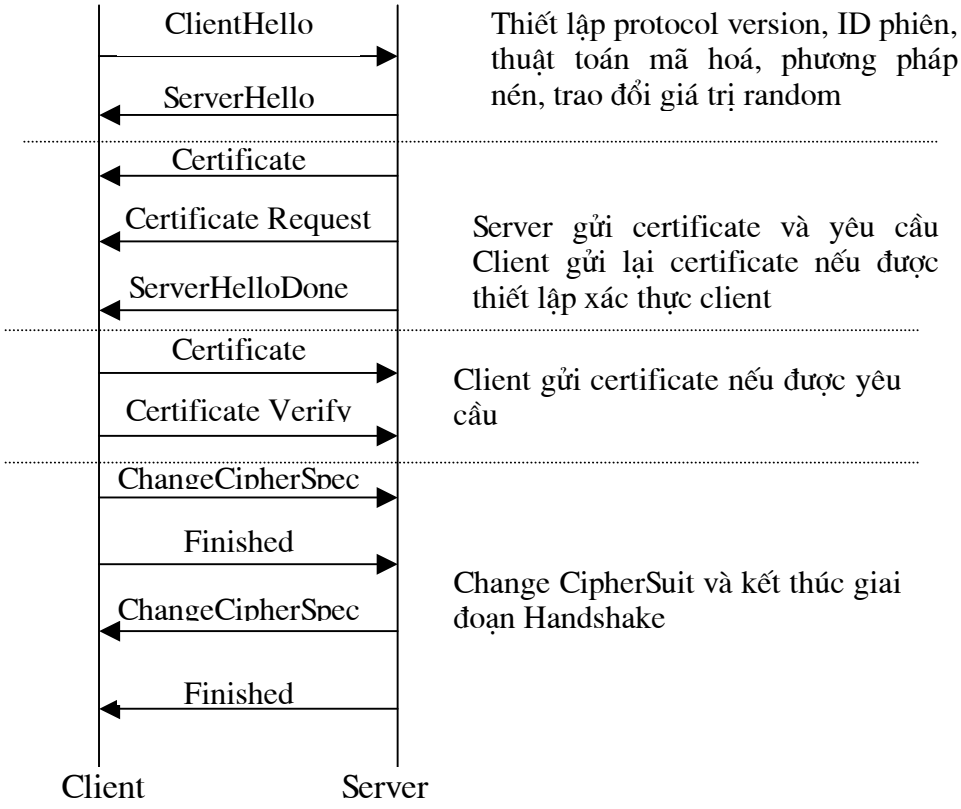
padding_length độ dài của padding.

Chú ý:
-Cũng như đối với trường hợp dùng mã dòng, đối với trường hợp dùng mã khối dữ liệu đầu vào khi mã một frame bao gồm cả kết quả hàm MAC.
-Giá trị khởi đầu của IV dùng cho frame dữ liệu đầu tiên được tạo trong quá trình thiết lập phiên liên lạc (Handshake), còn khi thực hiện mã hoá các frame tiếp theo, IV sẽ là block cuối cùng của ciphertext của frame trước nó.

Đối với các loại dữ liệu của các ứng dụng sử dụng SSL như HTTP, Telnet, ... chung ta không quan tâm. Dưới đây chúng tôi sẽ trình bày chi tiết định dạng của các loại dữ liệu trong giai đoạn thiết lập phiên liên lạc (Handshake, change Cipher Spec, Alert protocols).

2.2-SSLv3 Handshake protocol

Các tham số mật mã liên quan đến một phiên liên lạc được thực hiện thông qua SSLv3 Handshake Protocol, nó nằm ngay bên trên SSL Record Layer. Khi SSL client và SSL server bắt đầu một phiên liên lạc chúng cần thống nhất về phiên bản của giao thức sẽ được dùng, lựa chọn thuật toán mã hoá cho phiên liên lạc, có thể có hoặc không việc xác thực lẫn nhau, và sử dụng thuật toán mã hoá khoá công khai để sinh khoá chung cho phiên liên lạc đó.
Có thể mô phỏng giai đoạn thực hiện thiết lập phiên liên lạc bởi sơ đồ dưới đây:



Tất cả các messages trao đổi qua lại giữa server và client phải được biểu diễn theo một cấu trúc định trước. Định dạng của cấu trúc dữ liệu trong giai đoạn handshake như sau:

```
enum {
        hello_request(0),client_hello(1),server_hello(2),certificate(11),
        server_key_exchange(12),certificate_request(13),
        server_hello_done(14),certificate_verify(15),
        client_key_exchange(16),finished(20),(255)
}
struct {
        HandshakeType mstype;
        uint24 length;
        select (HandshakeType) {
                case hello_request: HelloRequest;
                case client_hello: ClientHello;
                case server_hello: ServerHello;
                case certificate: Certificate;
                case server_key_exchange: ServerKeyExchange;
                case certificate_request: CertificateRequest;
                case server_hello_done: ServerHelloDone;
                case certificate_verify: CertificateVerify;
                case client_key_exchange: ClientKeyExchange;
                case finished: Finished;
}
```

Cụ thể quá trình thực hiện SSLv3 Handshake qua các bước trao đổi dữ liệu giữa client/server như sau:

- ***Hello Messages.***

-Khi một client có nhu cầu kết nối tới server, server sẽ gửi một message gọi là Hellorequest tới client đó. Dưới đây là định dạng của gói Hellorequest:

```
struct { } Hello;
```

-Nhận được hellorequest, Client gửi clienthello gồm: protocol version, một giá trị random, Session ID, danh sách các thuật toán mã hoá, danh sách các mode nén dữ liệu. định dạng của gói clienthello:

```
struct {
        uint32 gmt_unix_time;
        opaque random_bytes[28];
}
opaque SesionID<1..32>;
uint8 CipherSuite[2];
enum {null(0),(255)} CompressionMethod;
 struct {
        ProtocolVersion client_version;
        Random random;
        SessionID session_id;
        CipherSuite cipher_suites<0..2^16-1>;
        CompressionMethod compression_methods<0..2^8-1>;
} ClientHello;
```

Trong đó:

random_bytes: 28 byte sinh từ bộ sinh random.

gmt_unix_time: thời gian hiện hành.
client_version: phiên bản SSL clietn dùng.
session_id: số ID của phiên liên lạc
cipher_suites: danh sách các thuật toán mã hoá client có.
compression_methods: danh sách các thuật toán nén client có.

-Sau khi nhận được clienthello, server sẽ gửi trả lời bằng gói dữ liệu gọi là serverhello gồm protocol version, giá trị sinh ngẫu nhiên, Session ID, danh sách các thuật toán mã hoá trong danh sách clienthello mà nó có và các chế độ nén.

Định dạng của gói ServerHello:

```
struct {
        ProtocolVersion server_version;
        Random random;
        SesionID sesion_id;
        CipherSuite cipher_suite;
        CompressionMethod compression_method;
}ServerHello
```

Trong đó một số tham số cần chú ý sau:

cipher_suites: một thuật toán mã hoá được chọn từ danh sách các thuật toán của client gửi sang.

compression_methods: một thuật toán nén được chọn trong các thuật toán client gửi sang.

- ***Server Certificate***

Server gửi tiếp server certificate (có cả danh sách các chain certificate, cũng có thể là null trong trường hợp server không có certificate). Định dạng của gói Certificate:

```
opaque ASN.1Cert<1..2^24-1>
struct {
        ASN1.Cert certificate_list<1..2^24-1>;
}Certificate;
```

- ***Server Key Exchange message***

Trong trường hợp không có certificate, hoặc có certificate nhưng chỉ sử dụng để ký (DSS certificate, signing-only RSA certificate) hoặc FORTEZZA KEA key exchange được dùng, server sẽ gửi ServerKeyExchange:

```
struct{
        select(KeyExchangeAlgorithm){
                case diffie_hellman:
                        ServerDHParams params;
                        Signature signature_params;
                case rsa:
                        ServerRSAParams params;
                        Signature signature_params;
                case fortezza_kea:
                        ServerFortezzaParams params;
        };
```

}ServerKeyExchange;

Chú ý:
-Biến params là một kiểu cấu trúc trong đó lưu các tham số thực hiện thực hiện một thuật toán trao đổi khoá công khai nào đó. Chẳng hạn với Diffie-Hellman params gồm p, g, và tham số khoá công khai của server g^x mod p.
-Signature ở đây chỉ là một cấu trúc chỉ ra thuật toán băm và kết quả sử dụng hàm băm các giá trị random của client, server và params của server. Chẳng hạn với dsa:

Signature là SHA(ClientHello.random + ServerHello.random + ServerParams).

- ***Certificate Request***

Sau đó Server sẽ gửi tiếp CertificateRequest với định dạng như sau:

```
struct {
        ClientCertificateType certificate_types <1..2^8-1>;
        DistinguishedName certificate_authorities <3..2^16-1>;
} CertificateRequest;
```

Trong đó ClientCertificateType liệt kê các kiểu certificate (rsa_sign, dss_sign, ...) mà server có thể chấp nhận. DistinguishedName là danh sách các DN của các certificate được server chấp nhận là certificate authority (CA certificate).

- ***Server hello done***

Đây là message thông báo hết giai đoạn gửi serverhello, định dạng của goi này như sau:

```
struct{ } ServerHelloDone;
```

- ***Client certificate***

Client chỉ được phép gửi gói dữ liệu này sau khi đã nhận được gói ServerHelloDone và nó cũng chỉ được gửi nếu server có yêu cầu certificate. Nếu không có một certificate nào tương ứng với một trong các kiểu certificate, và do một CA nào đó trong CertificateRequest server gửi cho thì client sẽ gửi thông báo no_certificate, khi đó tuỳ thuộc vào việc có yêu cầu hay không yêu cầu xác thực client mà server đưa ra khuyến cáo hoặc báo lỗi trong quá trình handshake. Trường hợp nếu client có certificate nó sẽ được gửi trong gói có định dạng certificate chúng tôi đã nêu ở trên.

- ***Client key exchange message***

Gói này được gửi từ client với định dạng như sau:

```
struct {
        select (KeyExchangeAlgorithm) {
                case rsa: EncryptedPreMasterSecret;
                case difie_hellman: ClientDiffieHellmanPublic;
                case fortezza_kea: FortezzaKeys;
        }
}ClientKeyExchange;
```

Trong đó:

- Nếu rsa được chọn EncryptedPreMasterSecret sẽ là kết quả mã hoá khoá công khai RSA (với các tham số khoá của server) của đầu vào là 48 byte (46 byte random sinh bởi client, 2 byte chỉ version cao nhất của giao thức mà client được support)
- Nếu DH được chọn, ClientDiffieHellmanPublic là thành phần công khai của client (g^y mod p).
- Nếu fortezza_kea được chọn, FortezzaKey là một cấu trúc chỉ ra đầy đủ các thành cần cho cho việc thiết lập khoá riêng của phiên liên lạc đó.

- ***Certificate verify***

Đây là gói dữ liệu thông báo quá trình xác thực lẫn nhau giữa client và server.

- ***Finished***

Kết thúc giai đoạn Handshake (gói dữ liệu này chỉ được trao đổi ngay sau khi quá trình thống nhất thuật toán mã hoá kết thúc).

2.3-Change cipher spec và Alert protocol

- ***Change cipher spec protocol***

Giao thức này chỉ bao gồm một message trong đó thực hiện chức năng thông báo việc thiết lập các thuộc tính mật mã cho phiên liên lạc đã hoàn thành. Message này chỉ có một byte duy nhất với giá trị là 1.

```
struct{
        enum{change_cipher_Spec(1),(255)} type;
}ChangeCipherSpec;
```

Change cipher spec message được gửi từ cả máy client lẫn máy server để thông báo cho bên nhận biết bắt đầu từ các frame dữ liệu tiếp theo sẽ được mã hoá bởi CipherSpec và khoá vừa thiết lập được. Client gửi message ngay sau khi gửi xong KeyExchange message, còn server gửi ngay sau khi nhận và xử lý xong KeyExchange từ client.

- ***Alert protocol***

Một trong những kiểu dữ liệu được suport bởi SSL record layer là kiểu alert (message thông báo). Các alert message truyền tải các thông báo lỗi trong quá trình thiết lập cũng như trao đổi dữ liệu của một phiên liên lạc. Cũng như các loại message khác, alert message cũng được mã hoá và nén.
Cấu trúc của alert message như sau:

```
enum {warning(1),fatal(2),(255)} AlertLevel;
enum {
        close_notify(0),
        unexpected_message(10),
        bad_record_mac(20),
        decompression_failure(30),
        handshake_failure(40),
        no_certificate (41),
        bad_certificate(42),
        unsuported_certificate(43),
        certificate_revoked(44),
```

```
        certificate_expired(45),
        certificate_unknow(46),
        illegal_parameter(47),
        (255)
} AlertDescription;
struct {
        AlertLevel level;
        AlertDescription description;
} Alert;
```

Quá trình tính khoá cho phiên liên lac:

Nếu mọi bước trong quá trình handshake đều trôi chảy, server/client sẽ thực hiện xong bước xác thực lẫn nhau và hai bên thiết lập được thuật toán mã hoá, thuật toán MAC và giá trị "mầm khoá" chung sử dụng cho phiên liên lạc đó.
Các thuộc tính mật mã sau khi thiết lập giữa hai máy sẽ được lưu vào CipherSpec dùng trong suốt phiên liên lạc đó. Cấu trúc của CipherSpec như sau:

```
enum {stream,block} CipherType;
enum {true, false} IsExportable;
enum {nul, rc4, rc2, des, 3des, des40, fortezza} BulkCipherAlgorithm;
enum {null, md5, sha1} MACAlgorithm;
struct {
        BulkCipherAlgorithm bulk_cipher_algorithm;
        MACAlgorithm mac_algorithm;
        CipherType cipher_type;
        IsExportable is_exportable;
        uint8 hash_size;
        uint8 key_masterial;
        uint8 IV_size;
} CipherSpec;
```

Giá trị mầm khoá được đặt vào biến pre_master_secret (trường hợp thuật toán trao đổi khoá RSA được dùng, giá trị mầm khoá được server giải mã từ biến EncryptPreMasterSecret, nếu thuật toán DH được chọn giá trị mầm khoá sẽ là g^x.y mod p, ...), từ giá trị khoá ban đầu này quá trình tính khoá cho phiên liên lạc được thực hiện trên mỗi máy như sau:

-Chuyển pre_master_key thành master_secret bằng cách:

```
master_secret=MD5(pre_master_secret + SHA('A' +pre_master_secret +
                ClientHello.random + ServerHello.random)) +
                MD5(pre_master_secret + SHA('BB' +pre_master_secret +
                ClientHello.random + ServerHello.random)) +
                MD5(pre_master_secret + SHA('CCC' +pre_master_secret +
                ClientHello.random + ServerHello.random)) ;
```

-Chuyển đổi từ master_secret thành keys và MAC secrets: từ master_secret sẽ được băm thành các giá trị khoá, IV và MAC secret sử dụng cho các thuộc tính được chỉ ra trong CipherSpec hiện tại.

```
key_block=
        MD5(master_secret + SHA('A' +pre_master_secret +
        ClientHello.random + ServerHello.random)) +
```

MD5(master_secret + SHA('BB' +pre_master_secret +
ClientHello.random + ServerHello.random)) +
MD5(master_secret + SHA('CCC' +pre_master_secret +
ClientHello.random + ServerHello.random)) + ...;

cho đến khi key_block đủ cho những thuộc tính được chỉ trong CipherSpec, tức là đủ để lấp đầy các biến:

client_write_MAC_secret[CipherSpec.hash_size]
server_write_MAC_secret[CipherSpec.hash_size]
client_write_key[CipherSpec.key_masterial]
server_write_key[CipherSpec.key_masterial]
client_write_IV[CipherSpec.IV_size]
server_write_IV[CipherSpec.IV_size]

Nếu đối với các thuật toán mã hoá none-export, thì quá trình tính khoá, IV và MAC secret đến đây là kết thúc. Còn đối với các thuật toán mã hoá exportable thì quá trình tính khoá cần thêm một bước nữa:

filnal_client_write_key=MD5(client_write_key +
ClientHello.random +
ServerHello.random));
filnal_server_write_key=MD5(server_write_key +
ClientHello.random +
ServerHello.random));

và IV thì được tính như:

client_write_IV=MD5(ClientHello.random + ServerHello.random);
server_write_IV=MD5(ClientHello.random + ServerHello.random);

Chương II
Sử dụng chứng chỉ số với dịch vụ Web

Sau khi được cấp chứng chỉ người sử dụng có thể dùng nó cho nhiều mục đích khác nhau. Trong chương này chúng tôi trình bày việc sử dụng chứng chỉ được cấp trên ứng dụng Web. Cụ thể là việc sử dụng một chứng chỉ thiết lập một ứng dụng thương mại điện tử ở mức đề mô (E-shop) trên Apache server để mọi người sử dụng chỉ có thể truy cập đến nó qua https, và cài đặt một chứng chỉ vào trình duyệt Internet Explorer trên môi trường Windows, một chứng chỉ cho Netscape trên Linux, để người sử dụng có thể dùng một trong hai trình duyệt trên truy cập đến E-Shop bằng cách sử dụng https.

1-Cài đặt chứng chỉ được cấp cho trình duyệt

1.1-Cài đặt chứng chỉ cho trình duyệt Internet Explorer

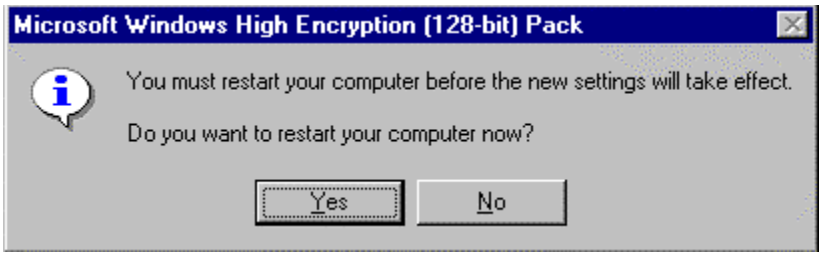
1.1.1-Bước 1: Cài đặt tiện ích trợ giúp

Các chứng chỉ hệ thống MyCA cấp cho người dùng hiện tại đều sử dụng thuật toán chữ ký số RSA với số modulo 1024 bít (đây cũng chính là độ dài khoá công khai của người dùng), tuy nhiên đối với ứng dụng Internet Explorer 5.0 (hoặc phiên bản thấp hơn) chỉ cho phép cài đặt các chứng chỉ với độ dài khoá công khai không quá 512 bít. Vì vậy để có thể cài đặt được chứng chỉ đã được cấp cho trình duyệt IE, người sử dụng cần cài đặt phần mềm hỗ trợ trước. Tiện ích thực hiện việc cài đặt phần mềm hỗ trợ là tệp ie5dom.exe được cung cấp cùng chương trình sinh khoá (trong đĩa mềm thứ nhất).

Để cài đặt người sử dụng chỉ cần nhấp đúp chuột vào tện tệp ie5dom.exe, khi đó trên màn hình xuất hiện hộp hội thoại như hình 1.

Hình 1

Người sử dụng chọn nút lệnh "Yes", quá trình nâng cấp được tiến hành và kết thúc khi trên màn hình xuất hiện thông báo như hình 2.



Hình 2

Người sử dụng chọn nút lệnh "Yes" để kết thúc quá trình cài đặt, máy tính sẽ được khởi động lại để tiện ích đã cài có hiệu lực.

1.1.2-Bước2: Cài đặt chứng chỉ cho Internet Explorer

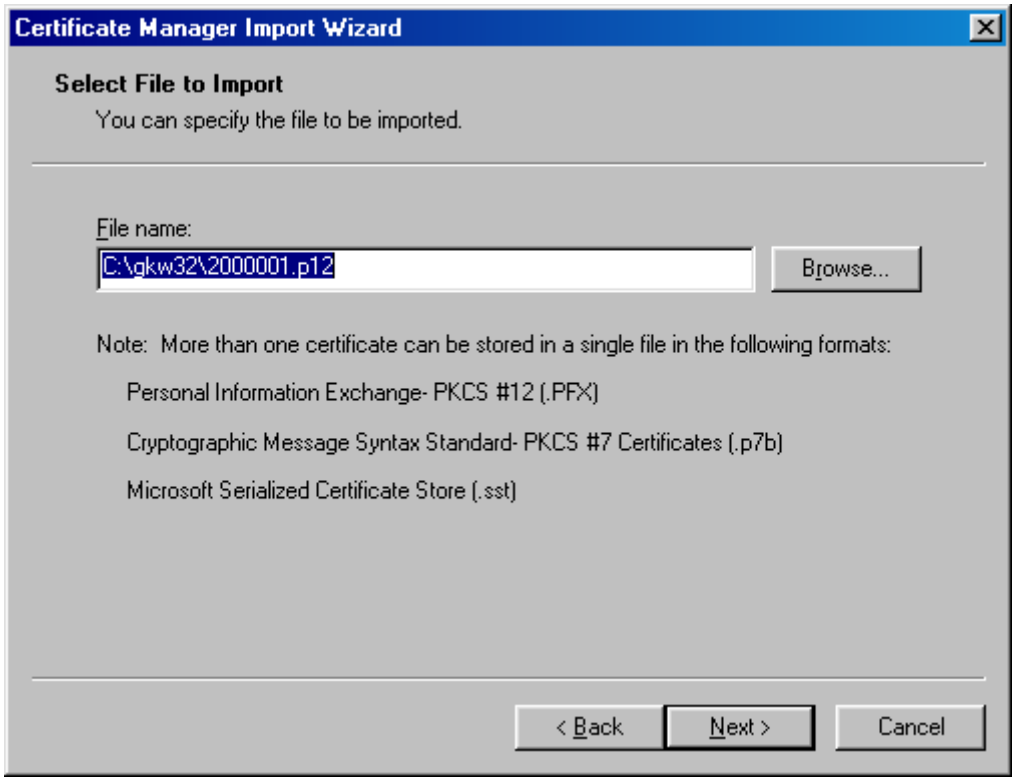
Giả sử người sử dụng được cấp chứng chỉ đưới định dạng PKCS12 là tệp 2000001.p12. Để cài đặt chứng chỉ cho Internet Explorer, người sử dụng chọn vào tệp này, rồi nhấn chuột phải.

Hình 3

Chọn chức năng "Install PFX", trên màn hình xuất hiện hộp hội thoại như hình 4.

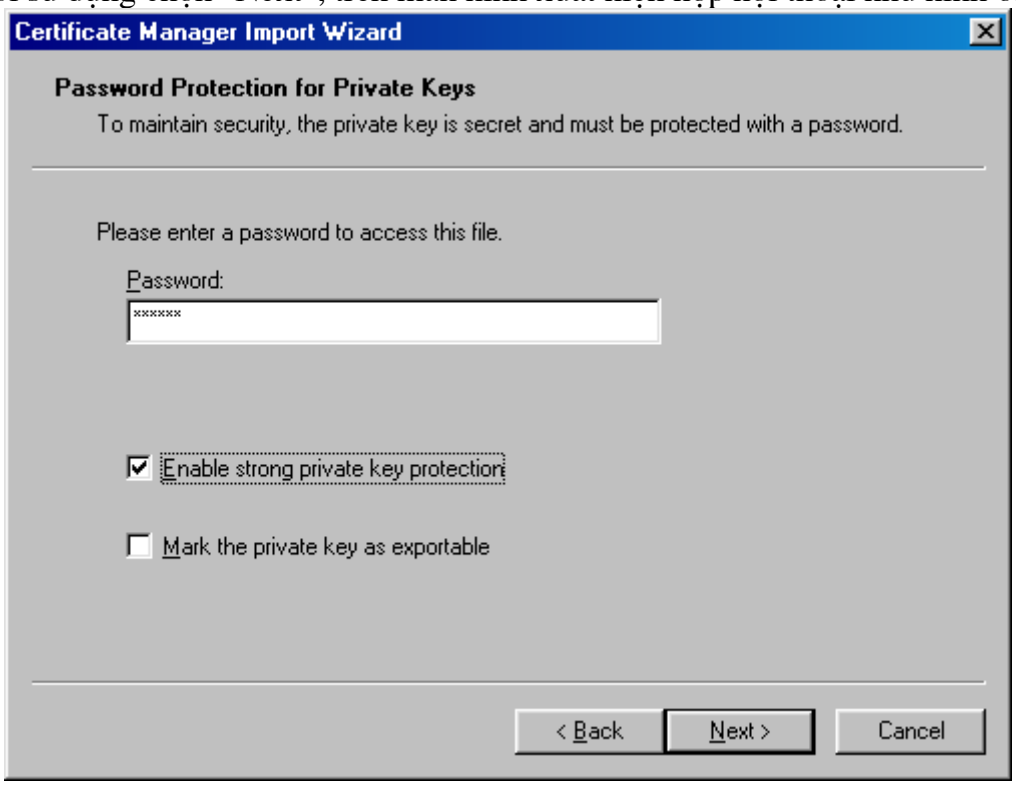
Hình 4

Người sử dụng chọn "Next", trên màn hình xuất hiện hộp hội thoại như hình 5.



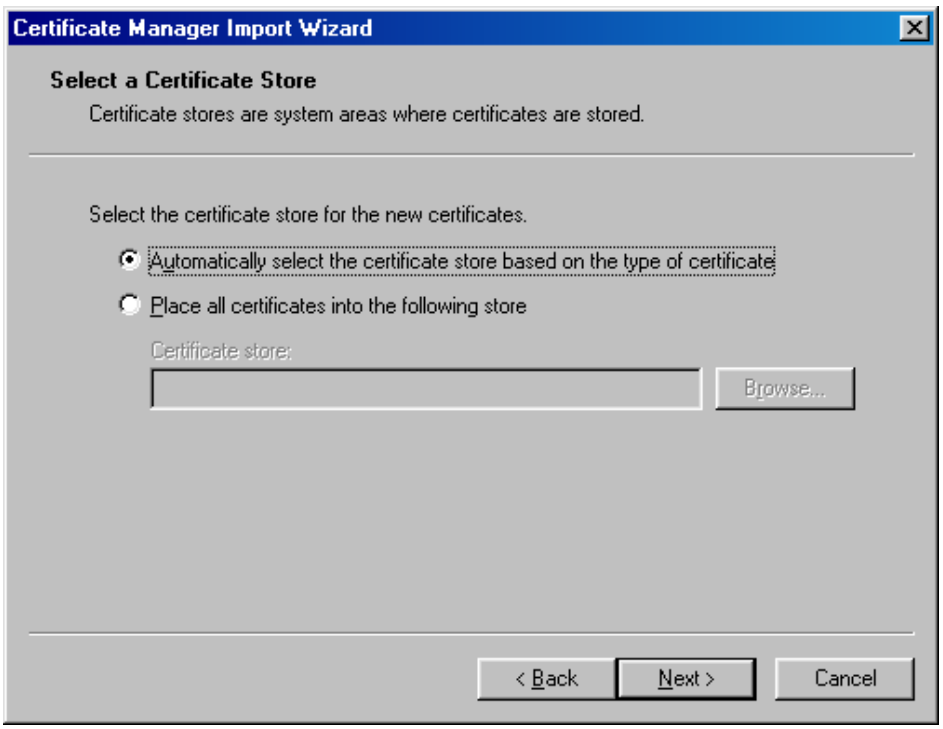
Hình 5

Người sử dụng chọn "Next", trên màn hình xuất hiện hộp hội thoại như hình 6.



Hình 6

Người sử dụng nhập mật khẩu (là mật khẩu do trung tâp cung cấp khi trung tâm CA thực hiện việc chuyển đổi định dạng của chứng chỉ trước khi cấp cho người sử dụng). Chọn "Next", trên màn hình xuất hiện hộp hội thoại như hình 7.

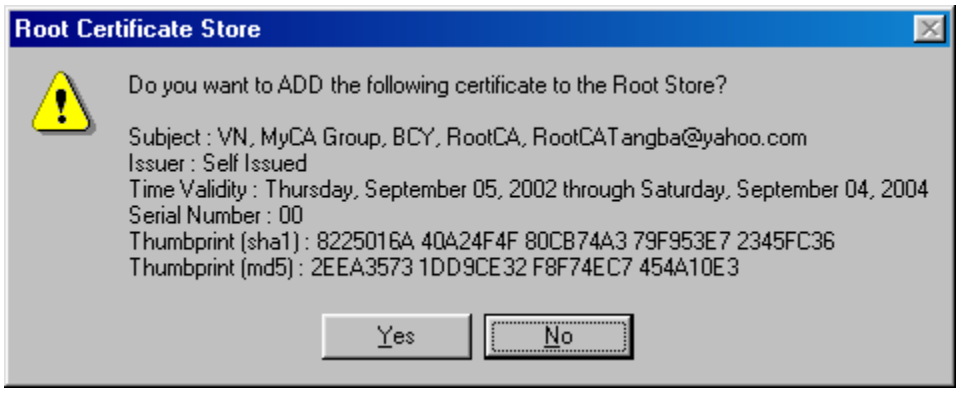


Hình 7

Người sử dụng chọn "Next", trên màn hình xuất hiện hộp hội thoại như hình 8.

Hình 8

Người sử dụng chọn "Finish", trên màn hình xuất hiện hộp hội thoại như hình 9 (có thể phải đợi trong ít giây, tùy thuộc vào tốc độ máy tính).

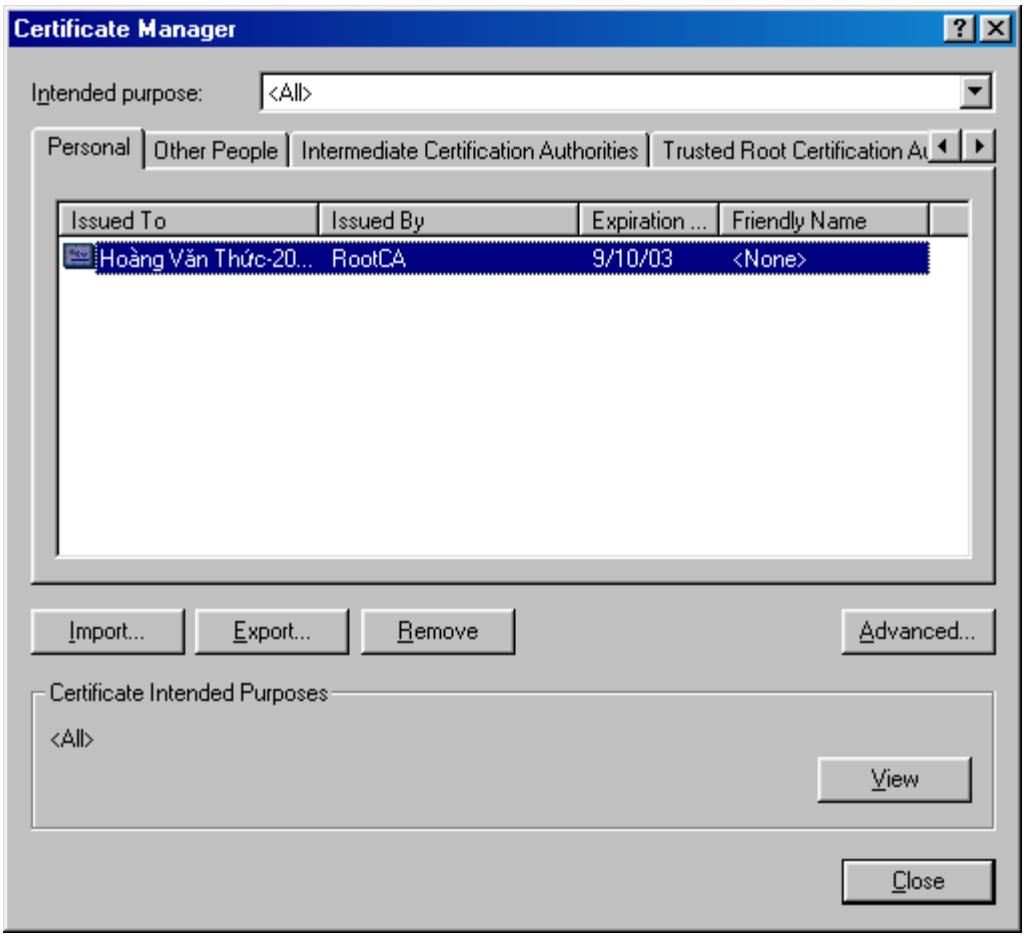


Hình 9

Người sử dụng chọn "Yes", trên màn hình xuất hiện hộp hội thoại thông báo quá trình cài đặt chứng chỉ đã kết thúc.

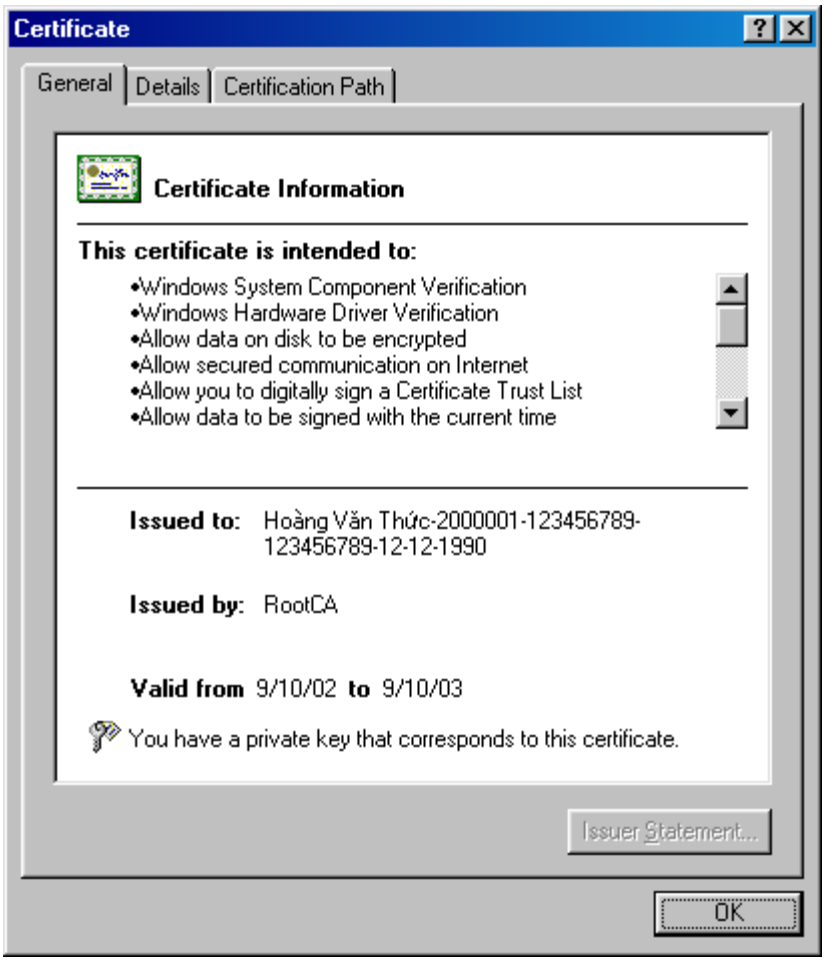
Hình 10

Sau khi cài đặt nếu người sử dụng mở trình duyệt IE, chọn menu /Tools/Internet Options, chọn tab "contents" rồi chọn nút lệnh "Certificate" sẽ thấy xuất hiện chứng chỉ vừa được cài đặt trong mục



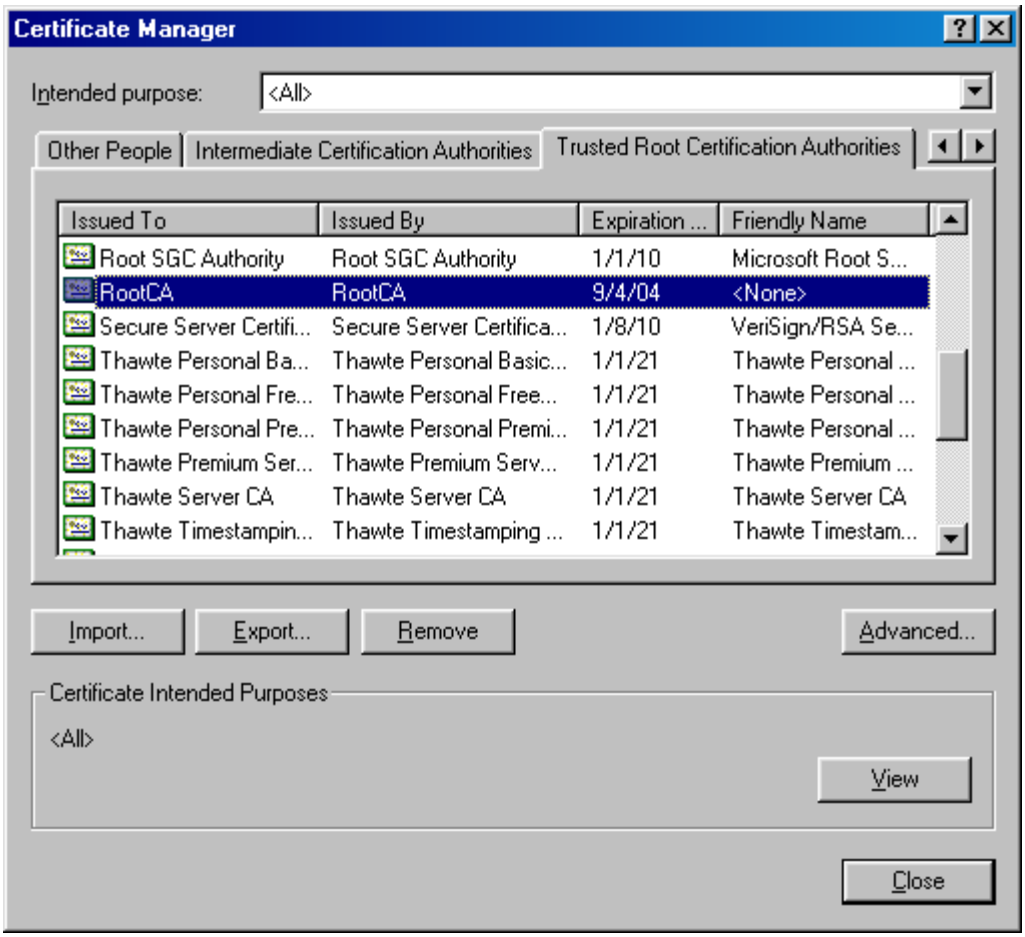
Hình 11

Muốn xem lại thông tin về chứng chỉ của mình người sử dụng chọn nút lệnh "View", trên màn hình xuất hiện hộp hội thoại như hình 12.



Hình 12

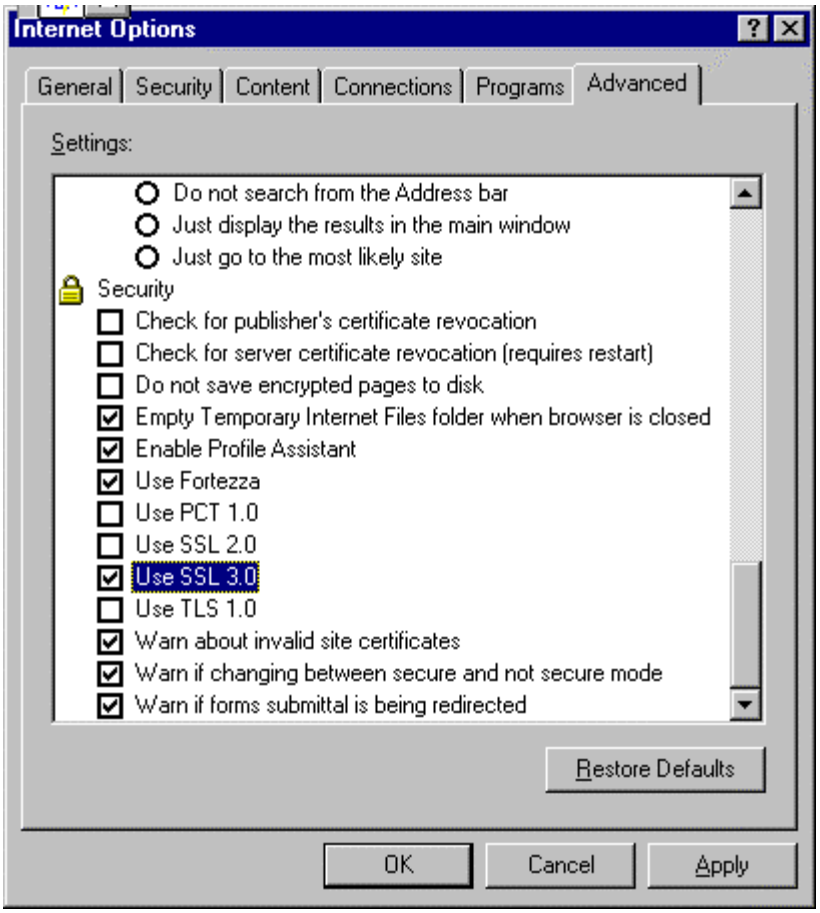
Cùng với việc cài đặt chứng chỉ của người sử dụng, các chứng chỉ của CA (khi sử dụng mô hình chain CA) cũng đồng thời được cài đặt, nếu chọn tab "Trust Root Certificate Authorities" sẽ thấy xuất hiện RootCA phát hành ra chứng chỉ của người sử dụng như hình 13 (ở đây chứng chỉ của người sử dụng được RootCA ký nên trong danh sách chỉ có một CA được cài đặt đó là Root CA).



Hình 13

Nội dung chứng chỉ của Root CA sẽ được lưu vào Registry (trong mục /system/certificates/Root/Certificate).

Sau khi cài đặt xong các chứng chỉ, người sử dụng cần thiết lập cấu hình cho IE: Chọn menu Tools/Internet Options/Advanced, hộp hội thoại xuất hiện như hình 14.



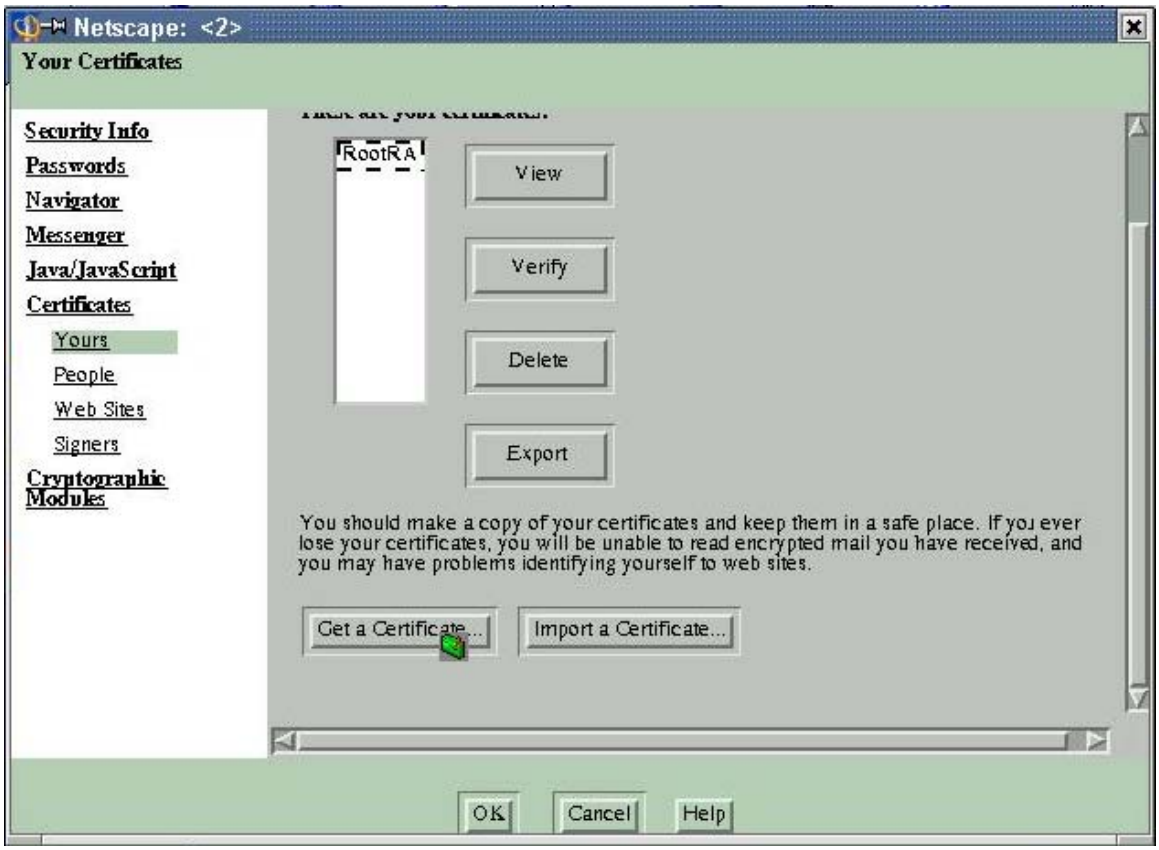
Hình 14

Trong mục "Settings" chọn "Use SSL 3.0" (hoặc "Use TLS 1.0"), rồi nhấn "OK".

1.2-Cài đặt chứng chỉ cho trình duyệt Netscape

Quá trình cài đặt chứng chỉ cho Netscape theo các bước sau:

-Trên menu của trình duyệt Netscape chọn chức năng "Security", hộp hội thoại xuất hiện như hình 15

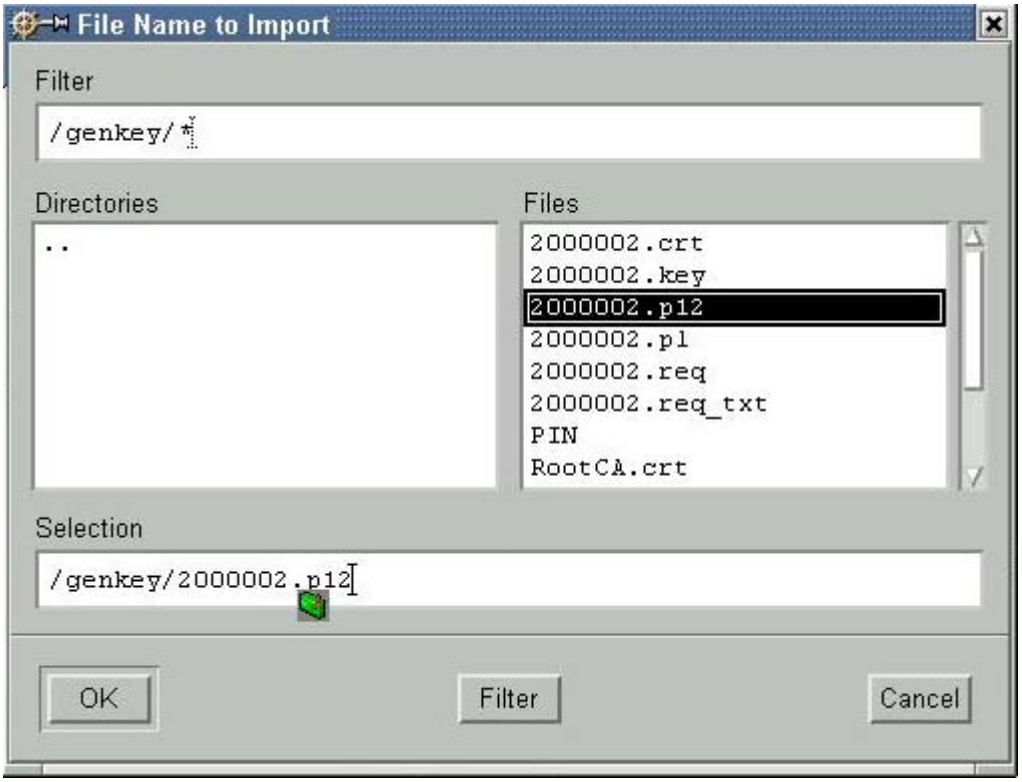


Hình 15

-Chọn "Yours" trong mục "Yours Certificates", rồi chọn "Import a Certificate", trên màn hình xuất hiện hộp hội thoại như hình 26

Hình 26

-Người sử dụng nhập mật khẩu để truy nhập tới cơ sở dữ liệu lưu chứng chỉ của Netscape (mật khẩu này người sử dụng thiết lập bằng cách sử dụng mục Passwords trong hộp hội thoại 15), chọn "OK", trên màn hình xuất hiện hộp hội thoại như hình 17.

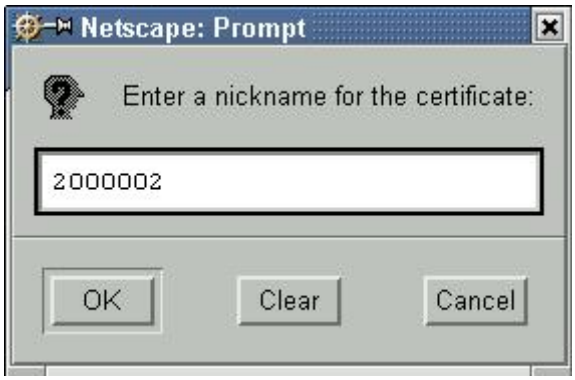


Hình 21

-Người sử dụng chọn tệp chứng chỉ cần cài đặt (ở đây là tệp 2000002.p12), rồi nhấn "OK", trên màn hình xuất hiện hộp hội thoại như hình 18.

Hình 18

-Người sử dụng gõ mật khẩu (là mật khẩu do trung tâm cung cấp khi thực hiện cấp chứng chỉ cho người sử dụng), nhấn "OK" trên màn hình xuất hiện hộp hội thoại như hình 19

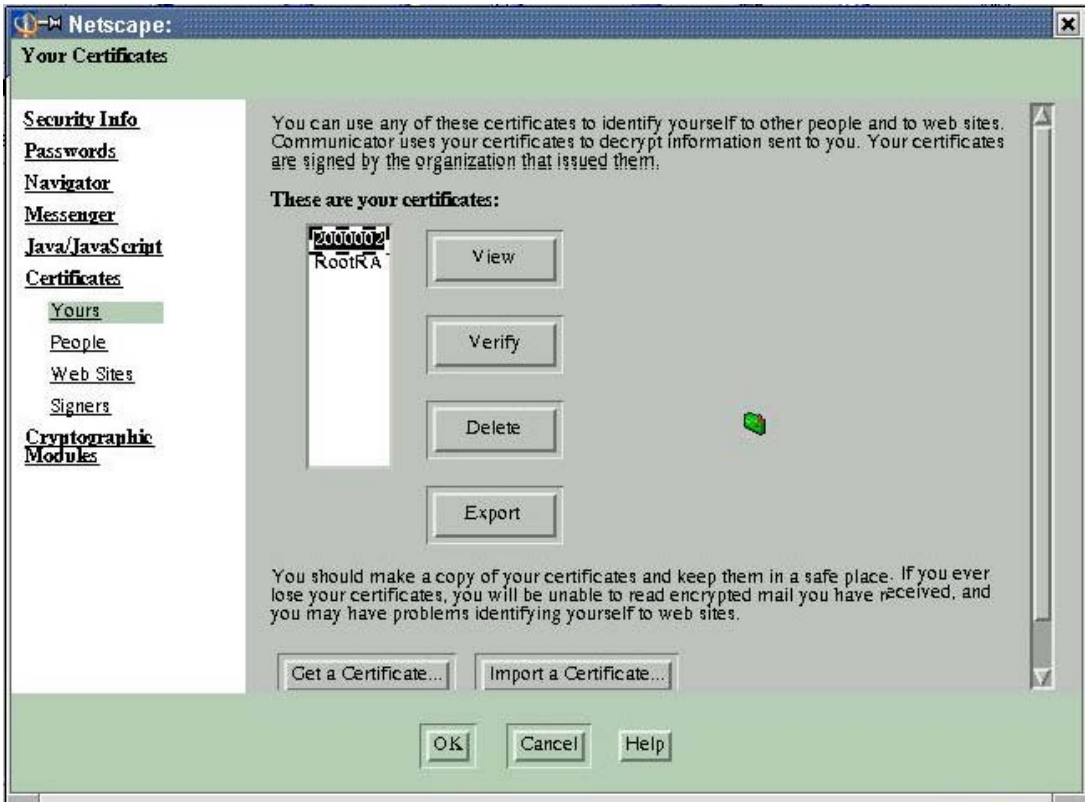


Hình 19

-Người sử dụng nhập tên chứng chỉ (tên này chỉ có ý nghĩa khi hiển thị trong danh mục các chứng chỉ được cài đặt), nhấn "OK" hộp hội thoại xuất hiện như hình 20 thông báo việc cài đặt chứng chỉ đã thành công.

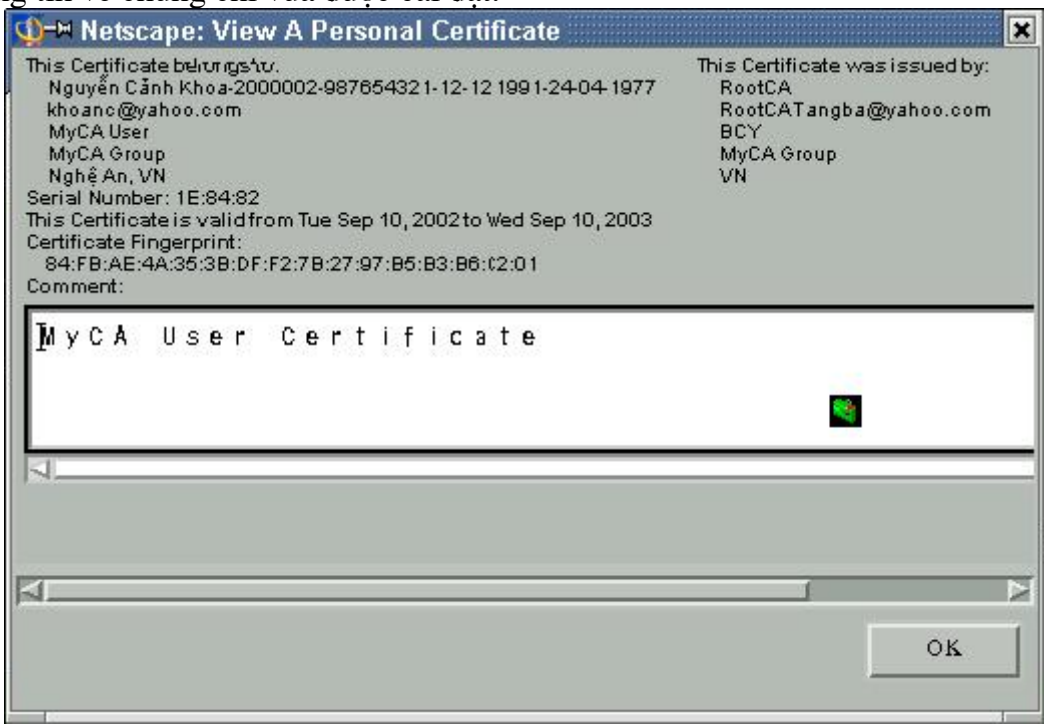
Hình 20

-Sau khi cài đặt xong chứng chỉ, ta thấy chứng chỉ vừa được cài đặt xuất hiện trong danh sách các chứng chỉ đã được cài đặt như hình 21.



Hình 21

-Người sử dụng nút lệnh "view" trên màn hình sẽ xuất hiện hộp hội thoại hiển thị thông tin về chứng chỉ vừa được cài đặt.

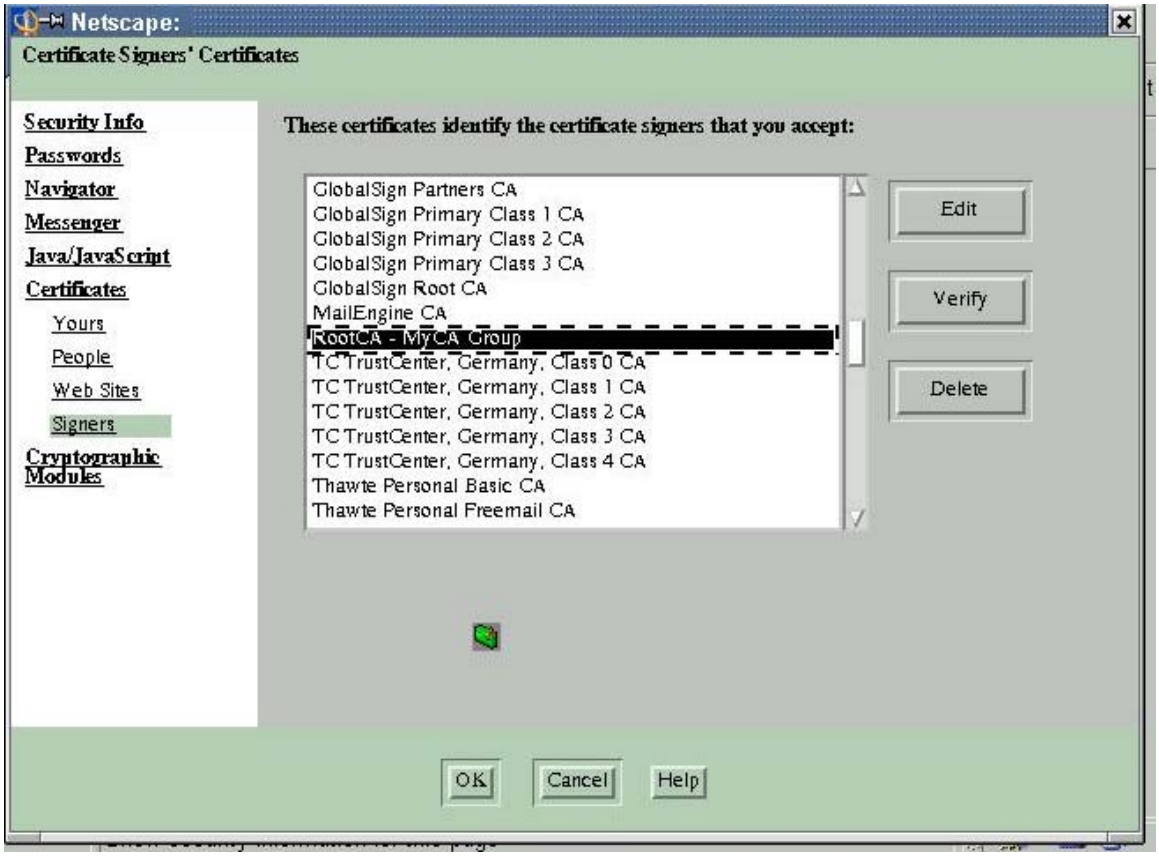


Hình 22

-Nếu dừng ở đây khi chọn nút lệnh "Verify" trong hình 21 sẽ xuất hiện thông báo:

Hình 23

-Để chứng chỉ có hiệu lực, người sử dụng cần thiết lập thuộc tính cho Netscape chấp nhận Root CA vừa cài là Certificate Authority, bằng cách: trong hình 21 chọn mục "Signers", khi đó hộp hội thoại hình 21 có dạng như hình 24.



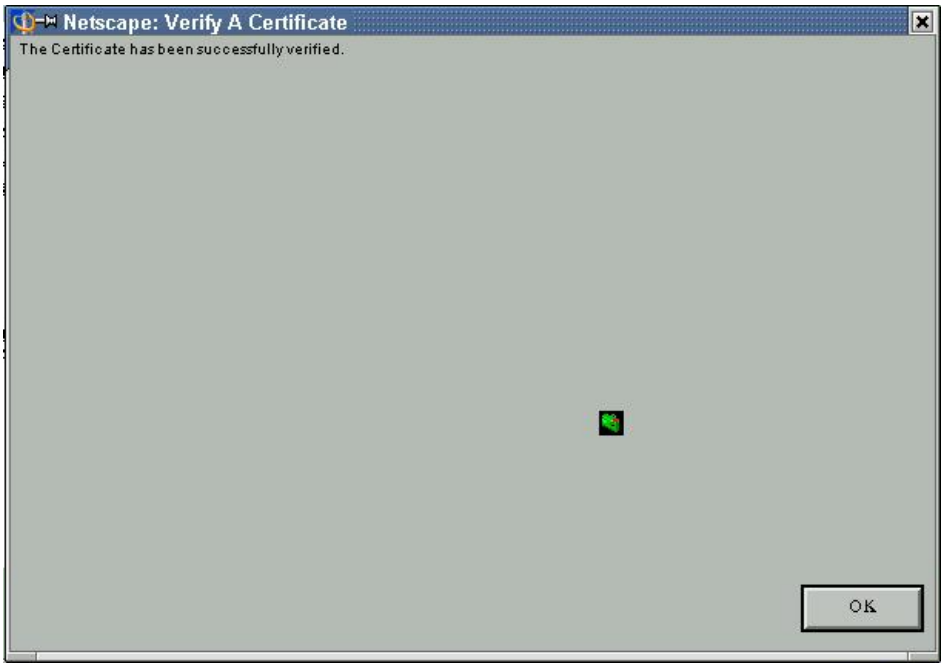
Hình 24

-Trong danh sách các chứng chỉ của CA chọn Root CA-MyCA Group, rồi chọn nút lệnh "Edit" trên màn hình xuất hiện hộp hội thoại như hình 25

Hình 25

-Người sử dụng check vào các lựa chọn trong mục "This Certificate belong to a Certifying Authority" rồi nhấn "OK".

-Đến đây nếu chọn nút lệnh "Verify" trên hình 23 sẽ có thông báo như hình 26.



Hình 26

2-Cập nhật CTL và CRL từ Public Database Server

Để cập nhật CRL, tải các CTL của người khác (sử dụng cho mục đích bảo mật Mail chẳng hạn), hoặc tải chuỗi các chứng chỉ của các CA tầng trên, người sử dụng có thể dùng trang publicdatabase.

Để truy cập được tới publicdatabase người sử dụng cần thiết lập cấu hình trên máy của mình như sau:

-Đối với trường hợp người sử dụng dùng môi trường Linux cần bổ sung thêm dòng:

200.1.1.1 publicdatabase

vào tệp /etc/hosts

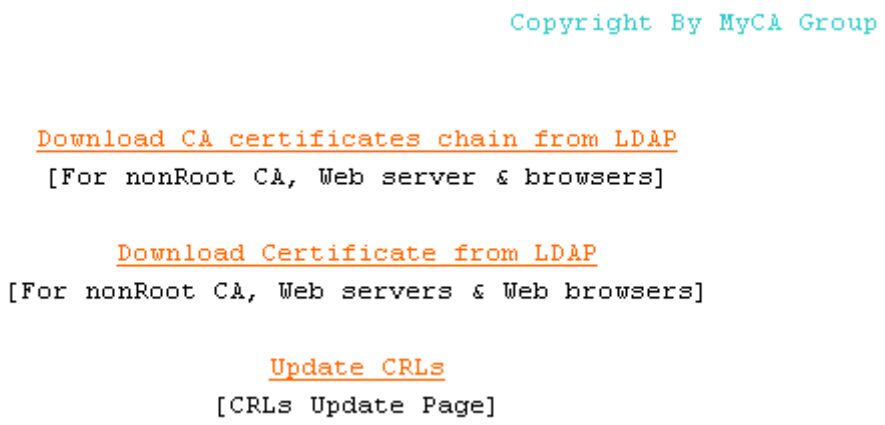
-Đối với người sử dụng dùng môi trường Windows cần bổ sung dòng :

200.1.1.1 publicdatabase

vào tệp c:\Windows\hosts

Trong đó 200.1.1.1 là địa chỉ IP của máy Public Database Server.

Khi truy cập tới trang http://publicdatabase, giao diện chính của trang publicdatabase xuất hiện như hình 27



Hình 27

Qui trình sử dụng các chức năng trên trang Publicdatabase chúng tôi đã trình bày kỹ trong chương 4 tài liệu về hệ thống phát hành và quản lý chứng chỉ số theo mô hình tập trung.

3-Cài đặt thiết lập cấu hình phần mềm E-shop có sử dụng chứng chỉ được cấp trên Apache server.

E-shop là một bộ chương trình nguồn mang tính mô phỏng việc thực hiện mua bán hàng hóa thông qua một trang Web (thương mại điện tử) trên môi trường Linux, toàn bộ phần mềm lưu trên một tệp eshopcom.zip. Sau khi gỡ nén tệp eshopcom.zip ta được thư mục eshop.com trong đó có toàn bộ các tệp chương trình mô phỏng quá trình mua bán trên web, và giao diện để thực hiện quá trình đó.
Quá trình cài đặt và thiết lập E-shop được tiến hành thông qua các bước sau:

3.1-Cài đặt phần mềm E-shop

Một số yêu cầu về phần mềm khi thiết lập E-shop:
-Việc trao đổi và lưu trữ dữ liệu khi chương trình thực hiện được tác giả của phần mềm sử dụng hệ quản trị cơ sở dữ liệu MySQL, do đó để chương trình có thể chạy được cần cài đặt MySQL.
-Bản chất của bộ chương trình là tạo nên một trang Web trên Web server, do đó cần cài đặt Apache server.

Để khởi tạo E-shop cần thực hiện:
-Gỡ nén tệp eshopcom.zip được thư mục eshop.com
-Tạo cơ sở dữ liệu trên MySQL bằng cách chuyển thư mục hiện hành thành ..../eshop.com/db và chạy lệnh:

mysqladmin create eshop

-Khởi tạo cơ sở dữ liệu eshop bằng cách thực hiện lệnh:

mysql eshop < eshop.sql

Quá trình cài đặt chương trình E-shop kết thúc, để sử dụng chương trình cần thiết lập cấu hình trên Apache server.

3.2- Thiết lập cấu hình E-shop có sử dụng chứng chỉ trên Apache server

Sau khi cài đặt phần mềm E-shop, để thiết lập cơ chế chỉ cho phép các browser truy nhập đến thông qua kênh bảo mật https, giả sử người muốn thiết lập eshop đã đăng ký và được cấp chứng chỉ dùng cho Web server với ID là 2000003. Khi đó cần thực hiện các bước thiết lập như sau:

-Thực hiện việc kết nối tới Public Database Server của hệ thống để dowload các CRL do các CA của hệ thống phát hành (nếu muốn kiểm soát được tính hợp lệ của tất cả các chứng chỉ của người dùng trong toàn bộ hệ thống thì cần tải tất cả các CRL của các CA về). Việc tải CRL thông qua Public Database Server chúng tôi đã trình bày trong mục 7.2. Sau khi có tệp CRL, chuyển các tệp này vào thư mục:

```
/etc/httpd/conf/ssl.crl/RootCA.crl
```

(ở đây chúng tôi chỉ trình bày việc sử dụng một tệp CRL do Root CA phát hành)

-Chuyển tệp lưu khoá riêng (được sinh bởi phần mềm sinh khoá và sinh tệp yêu cầu) vào thư mục:

```
/etc/httpd/conf/ssl.key/2000003.key
```

-Chuyển chứng chỉ được hệ thống MyCA cấp, RootCA.crt, chain.crt (nếu là nonroot CA cấp chứng chỉ) vào thư mục:

```
/etc/httpd/conf/ssl.crt/2000003.crt
/etc/httpd/conf/ssl.crt/RootCA.crt
/etc/httpd/conf/ssl.crt/chain.crt
```

-Thực hiện việc thiết lập E-shop trên Apache như sau:

```
<VirtualHost 200.1.1.1:443>
DocumentRoot /homme/Thuchv/eshop.com/
      ServerName eshop.com
ErrorLog logs/eshop /error.log
      CustomLog logs/eshop/access.log common
SSLEngine on
SSLProtocol +SSLv3 +TLSv1
SSLCertificateFile /etc/httpd/conf/ssl.crt/2000003.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/2000003.key
SSLCertificateChainFile /etc/httpd/conf/ssl.crt/chain.crt
SSLCACertificateFile /etc/httpd/conf/ssl.crt/RootCA.crt
SSLCACertificatePath /etc/httpd/conf/ssl.crt/
SSLCARevocationFile /etc/httpd/conf/ssl.crl/RootCA.crl
SSLVerifyClient require
SSLVerifyDepth  10
</VirtualHost>
```

-Khởi động lại Apache bằng cách thực hiện lệnh:

```
/etc/rc.d/init.d/httpd restart
```

4-Sử dụng https truy nhập tới E-shop

4.1- Sử dụng trình duyệt Internet Explorer truy nhập tới E-Shop

4.1.1-Trường hợp chứng chỉ của client và server chưa bị huỷ bỏ

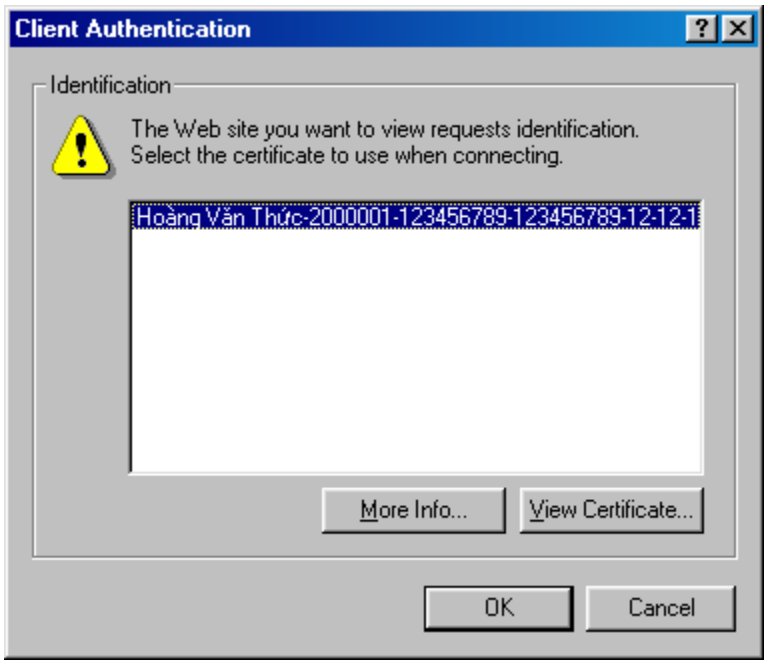
Để kết nối được E-shop, trong tệp c:\windows\hosts cần được bổ sung dòng sau:

200.1.1.1 eshop.com

Sau khi thực hiện việc cài đặt các chứng chỉ và thiết lập cấu hình cho IE, để kết nối tới E-Shop trên Apache Server sử dụng https, trên màn hình Internet Explorer xuất hiện khuyến cáo như hộp hội thoại 28.

Hình 28

Chọn "Ok", trên màn hình xuất hiện hộp hội thoại như hình 29



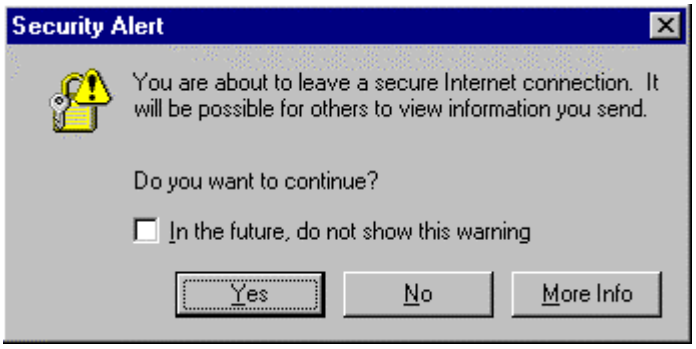
Hình 29

Người sử dụng chọn chứng chỉ của mình để Server xác thực (nếu không muốn xác thực client thì cần bỏ mục SSLVerifyClient trong tệp cấu hình của Apache Server, khi đó hộp hội thoại trên sẽ không xuất hiện), rồi nhấn "OK", trên màn hình xuất

hiện hộp hội thoại như hình 30. Thông báo về quá trình xác thực chứng chỉ giữa Server và browser.

Hình 30

Chọn "Yes", trong quá trình làm việc, mỗi khi có sự thay đổi trang truy nhập trên Server sẽ có hộp hội thoại khuyến cáo xuất hiện như hình 31, muốn cho hộp hội thoại này không xuất hiện check vào mục: " In the future, do not show this warning".



Hình 31

Cuối cùng trang https://eshop.com sẽ hiển thị trên màn hình IE như hình 32.

Hình 32

4.1.2-Trường hợp một trong hai chứng chỉ bị huỷ bỏ

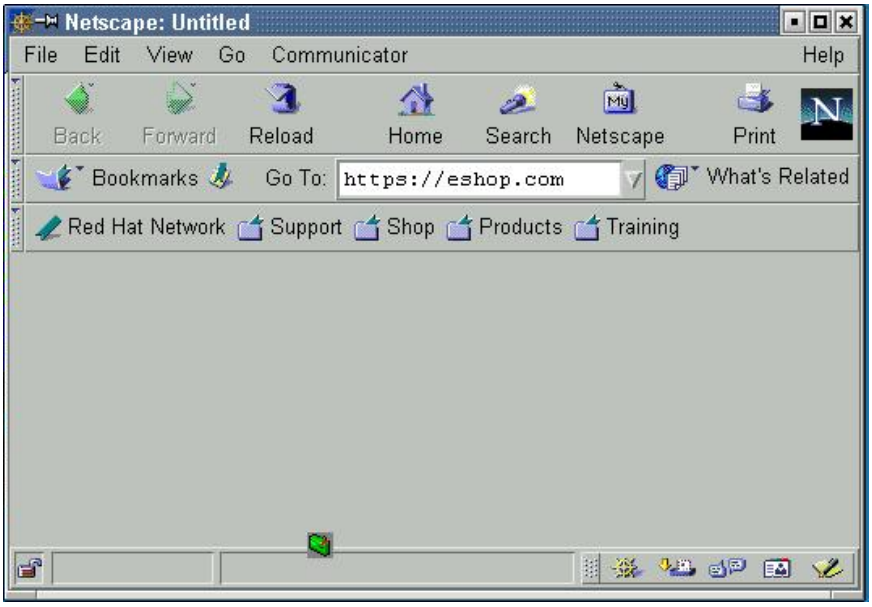
-Trường hợp chứng chỉ của client đã bị huỷ bỏ, quá trình kết nối tới server sẽ không thực hiện được, trên màn hình IE xuất hiện thông báo "**The page cannot be displayed**" (chú ý rằng thông báo này chỉ xuất hiện khi thuộc tính SSLVerifyClient được thiết lập là require).

-Trường hợp ngược lại, nếu chứng chỉ của server đã bị huỷ bỏ và tệp CRL có thông tin về việc chứng chỉ của server đã được huỷ bỏ cài đặt cho client thì quá trình làm việc giữa client và Server vẫn thực hiện được tuy nhiên trong quá trình kết nối sẽ không thực hiện xác thực server.

4.2- Sử dụng trình duyệt Netscape truy nhập tới E-Shop

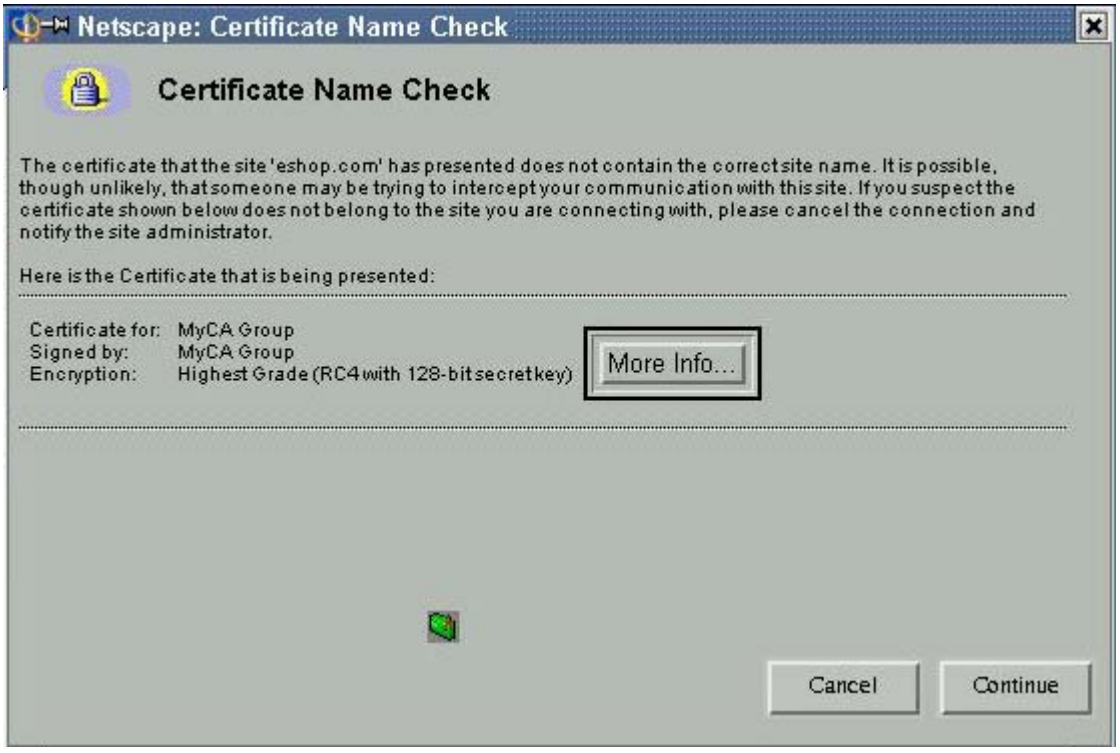
4.2.1-Trường hợp chứng chỉ của client và server chưa bị huỷ bỏ

Sau khi thực hiện việc cài đặt chứng chỉ, để kết nối tới E-Shop trên Apache Server sử dụng https, người sử dụng chạy trình duyệt Netscape, mở trang https://eshop.com.



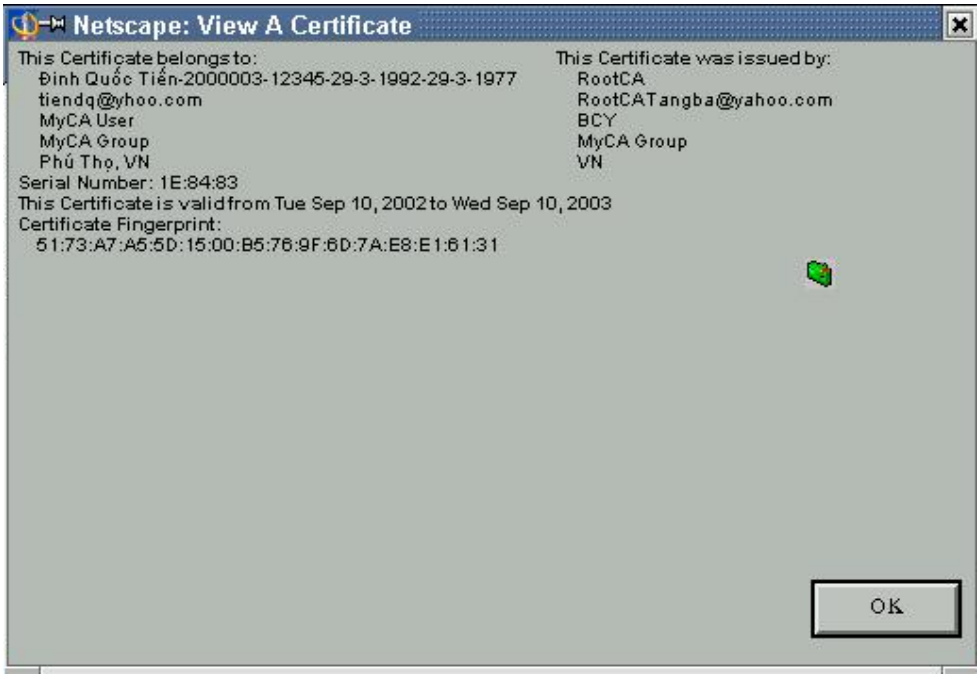
Hình 33

Trên màn hình Netscape xuất hiện hộp hội thoại như hình 34, thông báo về chứng chỉ trên server.



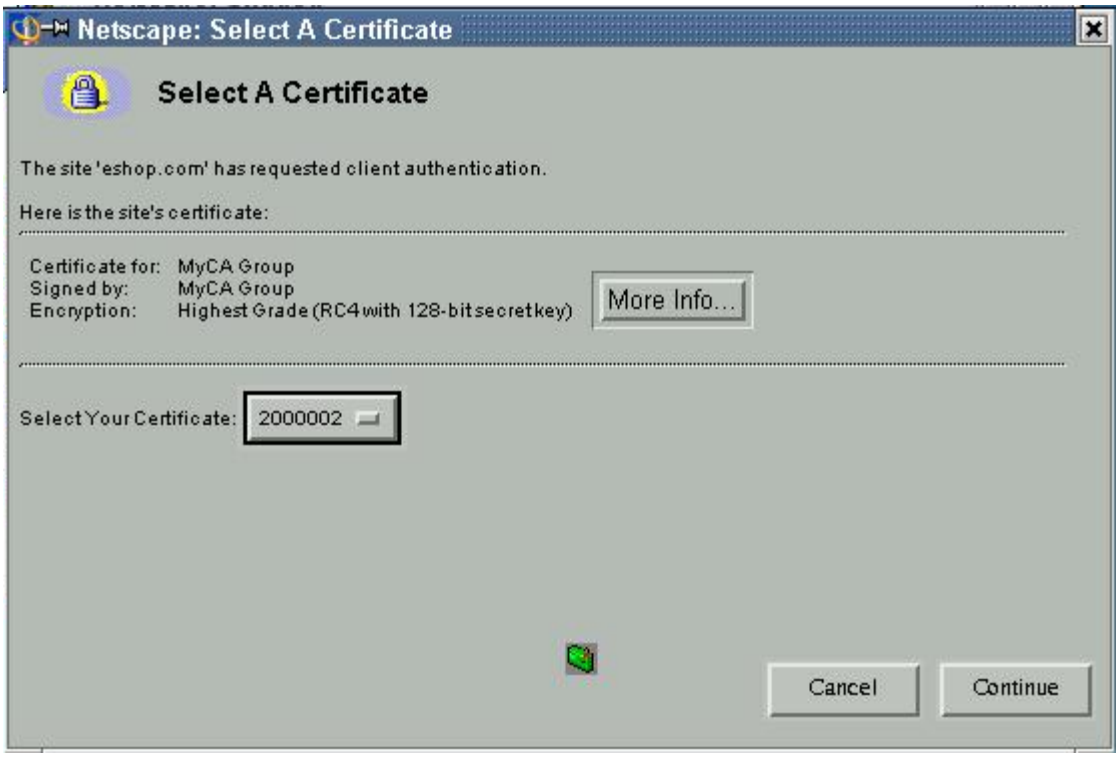
Hình 34

Nếu người sử dụng muốn xem thông tin về chứng chỉ của server thì chọn vào mục "More Info", hộp hội thoại như hình 35 xuất hiện



Hình 35

Từ hộp hội thoại 34 chọn "Continue", trên màn hình xuất hiện hộp hội thoại như hình 36, để người sử dụng chọn chứng chỉ làm việc với server.



Hình 36

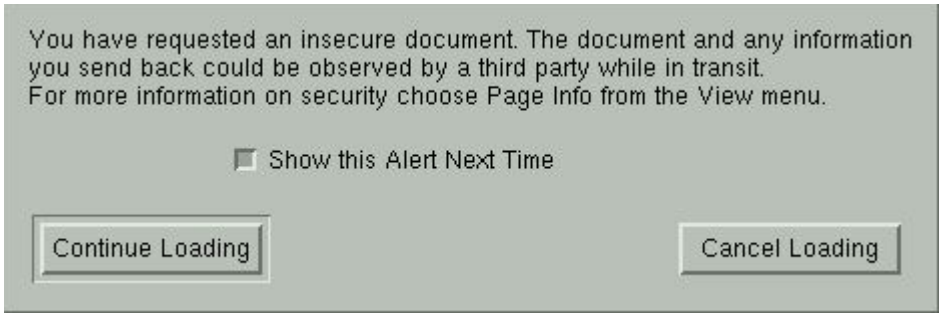
Chọn "Continue", trên màn hình xuất hiện hộp hội để người sử dụng nhập mật khẩu để truy nhập cơ sở dữ liệu của Netscape

Hình 37

Người sử dụng nhập mật khẩu rồi chọn "OK", trên màn hình xuất hiện hộp hội thoại thông báo quá trình xác thực đã kết thúc:

Hình 38

Khi chuyển từ trang web sử dụng chế độ bảo mật sang trang web không sử dụng chế độ bảo mật trên màn hình xuất hiện hộp hội thoại như hình 39



Hình 39

4.2.2-Trường hợp một trong hai chứng chỉ bị huỷ bỏ

Nếu chứng chỉ của client bị huỷ bỏ, khi thực hiện kết nối tới server quá trình được thực hiện từ hộp hội thoại 34 đến hộp hội thoại 37. Tuy nhiên quá trình kết nối đến server có sử dụng chế độ bảo mật sẽ không thành công với hộp hội thoại thông báo:

Hình 40

Nếu chứng chỉ của server bị huỷ bỏ, khi một client (đã cập nhật CRL trong đó chứng chỉ của server đã bị huỷ bỏ) kết nối đến server có sử dụng chế độ bảo mật sẽ không thành công, trên màn hình xuất hiện thông báo:

Hình 41

Chương III
Sử dụng chứng chỉ số với dịch vụ Mail

1. Giới thiệu

Một trong những ứng dụng quan trọng của chứng chỉ số đó là mã hoá và xác thực thư điện tử. Người sử dụng có thể cài chứng chỉ số (của mình) đã được phát hành từ một Trung tâm cấp chứng chỉ số vào một hệ mail nào đó để thực hiện trao đổi thư điện tử có bảo mật với một người dùng khác mà cùng chung hệ thống CA. Trong chương này chúng tôi nhằm giới thiệu cách sử dụng chứng chỉ số được cấp bởi hệ thống MyCA trên Outlook Express.

Giả thiết rằng bạn đã có một tài khoản người dùng được cấp trên máy Server, và bạn đã được cấp một chứng chỉ số tương ứng với tài khoản đó (nhất thiết địa chỉ email đăng ký trong chứng chỉ số phải trùng với địa chỉ email của tài khoản trên máy Server).

Thiết lập hệ thống của chúng tôi như sau:

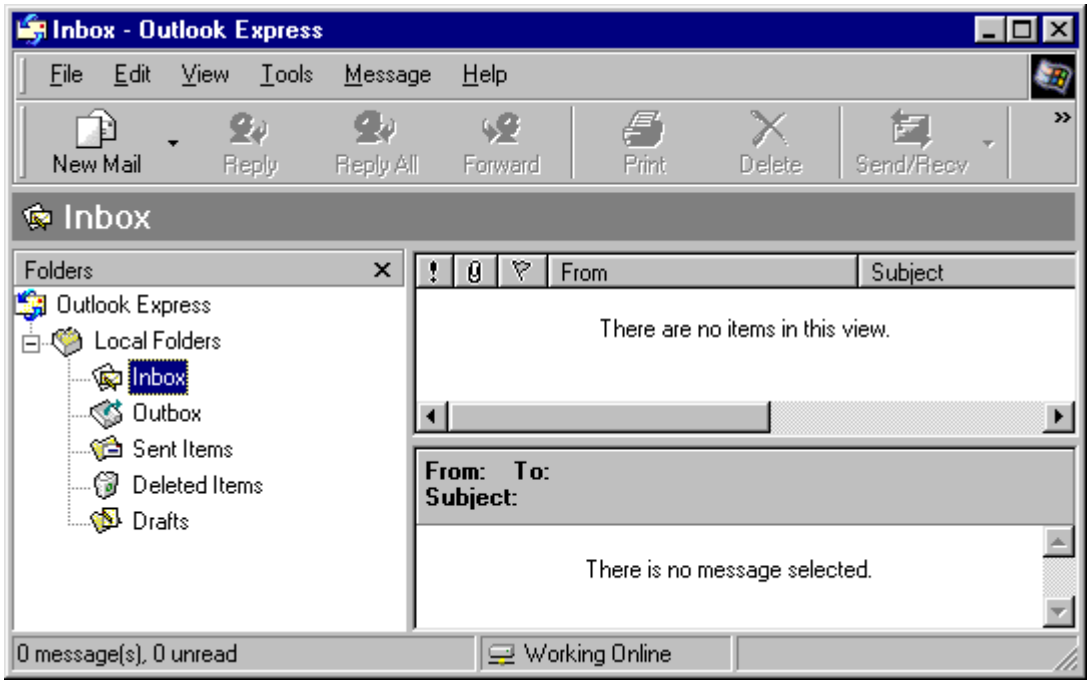
- Một máy cài Windows 2000 Advanced Server và Exchange Server 2000.
- 2 máy Windows 98.

Trên máy Server tạo 2 tài khoản người dùng là user1 và user2 có địa chỉ email là user1@myca.org và user2@myca.org. Trên 2 máy windows 98 lần lượt từng máy tạo địa chỉ email tương ứng với địa chỉ email được cấp trên Server. Như vậy, 2 người dùng trên 2 máy có thể gửi thư cho nhau bình thường (không xác thực và mã hoá). Để có thể gửi thư bảo mật giữa 2 người sử dụng này thì họ cần phải có chứng chỉ số mà được cấp cùng một hệ thống CA nào đó.

Sử dụng hệ thống cấp chứng chỉ MyCA để phát hành 2 chứng chỉ số cho 2 người sử dụng có địa chỉ email lần lượt là user1@myca.org và user2@myca.org. 2 người dùng này đem chứng chỉ của mình cài vào Outlook. Đến đây họ có thể gửi thư có mã hoá và xác thực cho nhau. Dưới đây chúng tôi sẽ trình bày chi tiết những việc làm này.

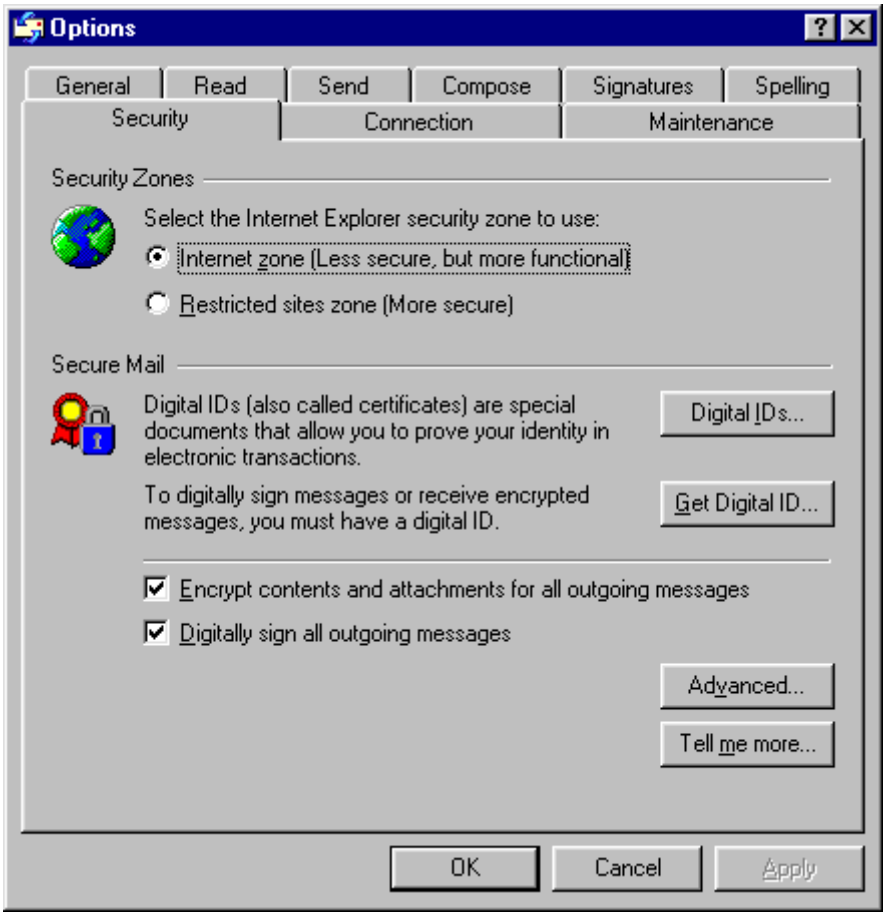
2. Đưa chứng chỉ số vào Outlook

Chạy Outlook, giao diện chương trình như Hình 1.



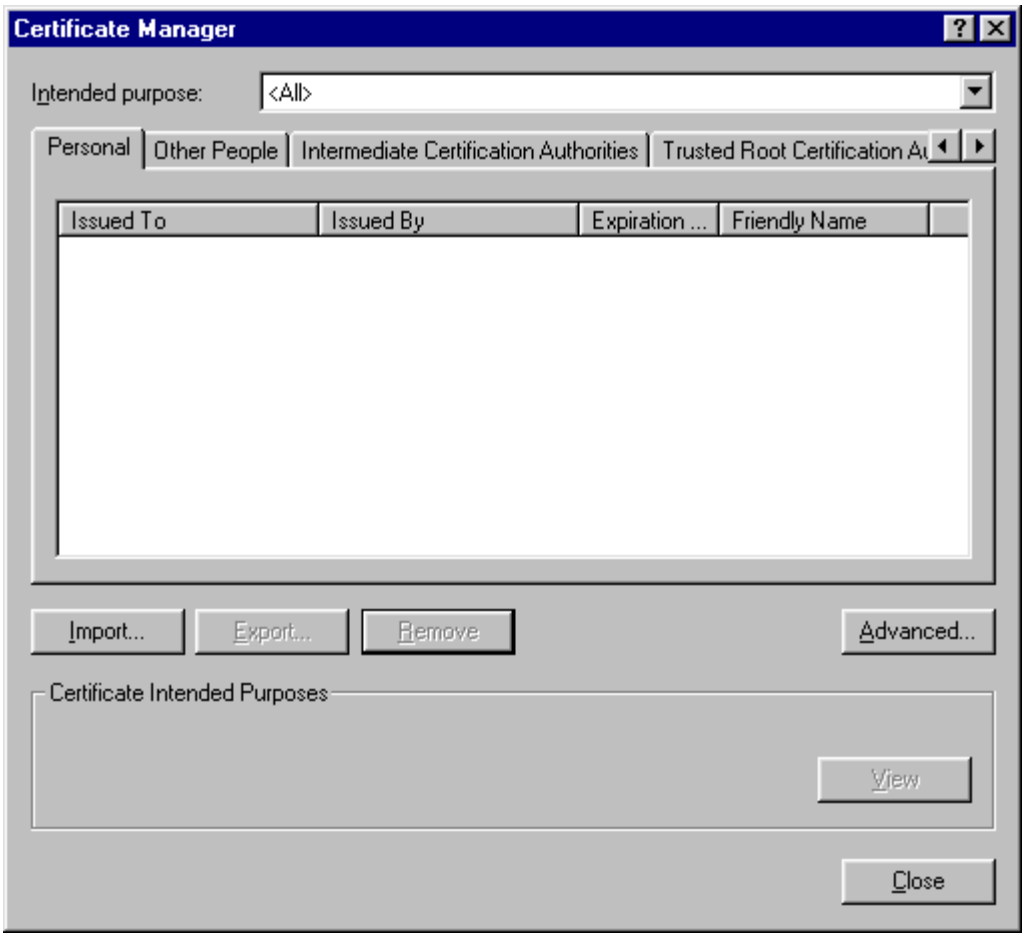
Hình 1

Để cài chứng chỉ số của bạn vào Outlook (chú ý là bạn phải chuyển đổi chứng chỉ theo định dạng PKCS#12 nếu chứng chỉ số được cấp từ hệ thống CA do người dùng sinh khoá), ở giao diện chính của Outlook chọn **Tools => Options => Security**, khi đó xuất hiện hộp thoại như Hình 2.



Hình 2

Với hộp thoại này bạn chọn **Digital IDs...**, xuất hiện hộp thoại như Hình 3.

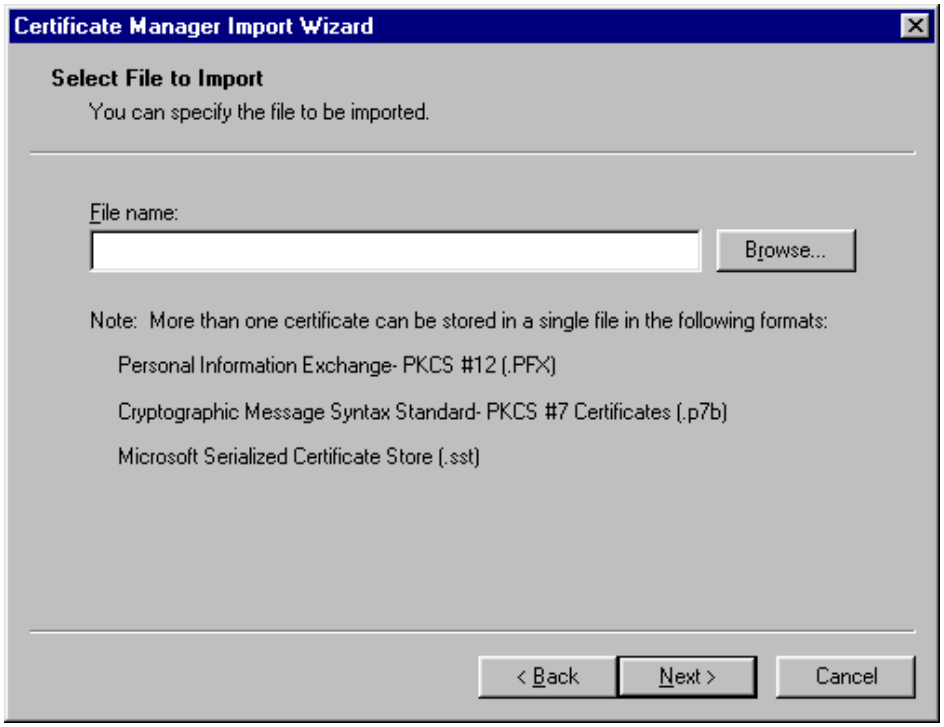


Hình 3

Trong mục **Personal** của hộp thoại **Certificate Manager** ở trên chưa có chứng chỉ nào được cài vào Outlook. Chọn **Import...** xuất hiện hộp thoại như Hình 4.

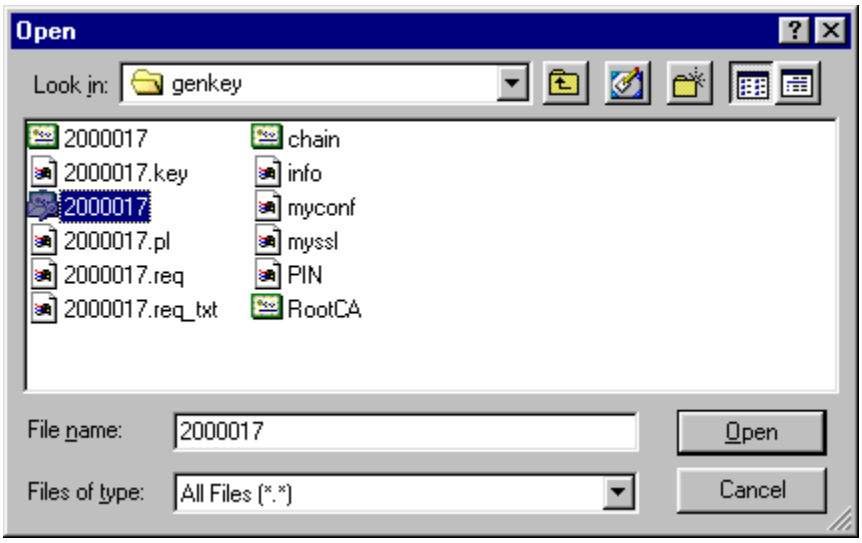
Hình 4

Chọn **Next>** để tiếp tục, xuất hiện hộp thoại như Hình 5.

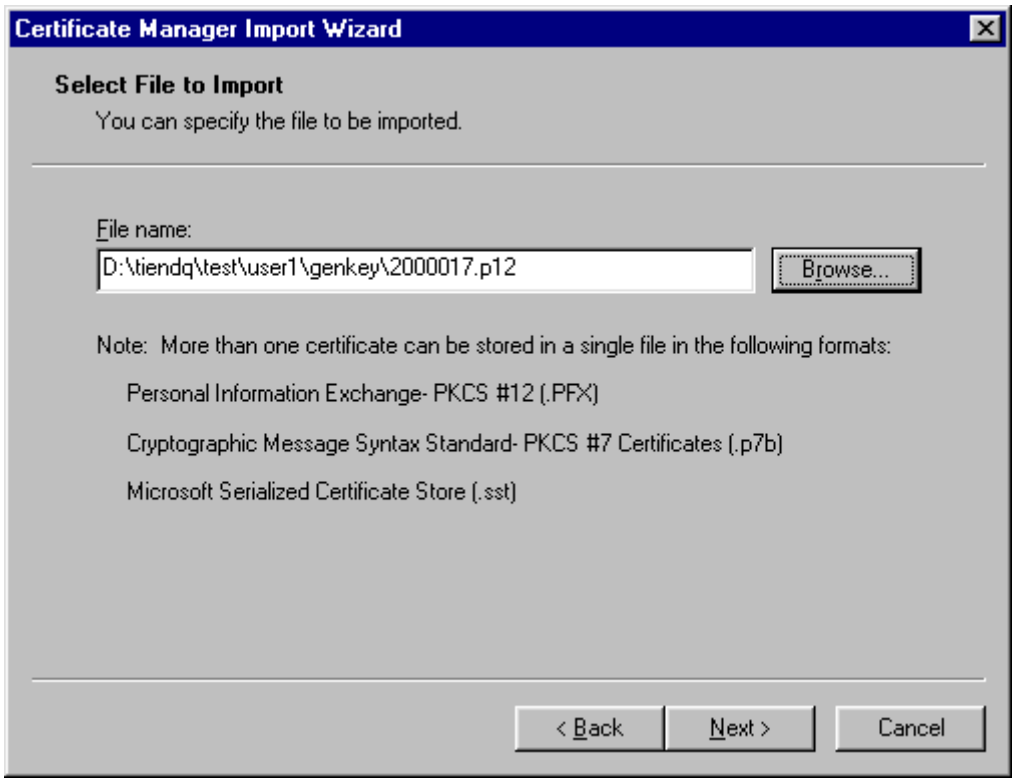


Hình 5

Hộp thoại yêu cầu chọn tệp chứng chỉ của bạn. Chọn **Browse...** để chỉ ra tệp chứng chỉ, nhưng chú ý rằng tệp chứng chỉ này với định dạng PKCS#12. Trong ví dụ của chúng tôi thì người dùng có tệp chứng chỉ là 2000017.p12 đặt trong thư mục d:\tiendq\test\user1\genkey, được thể hiện trong Hình 6 và Hình 7.

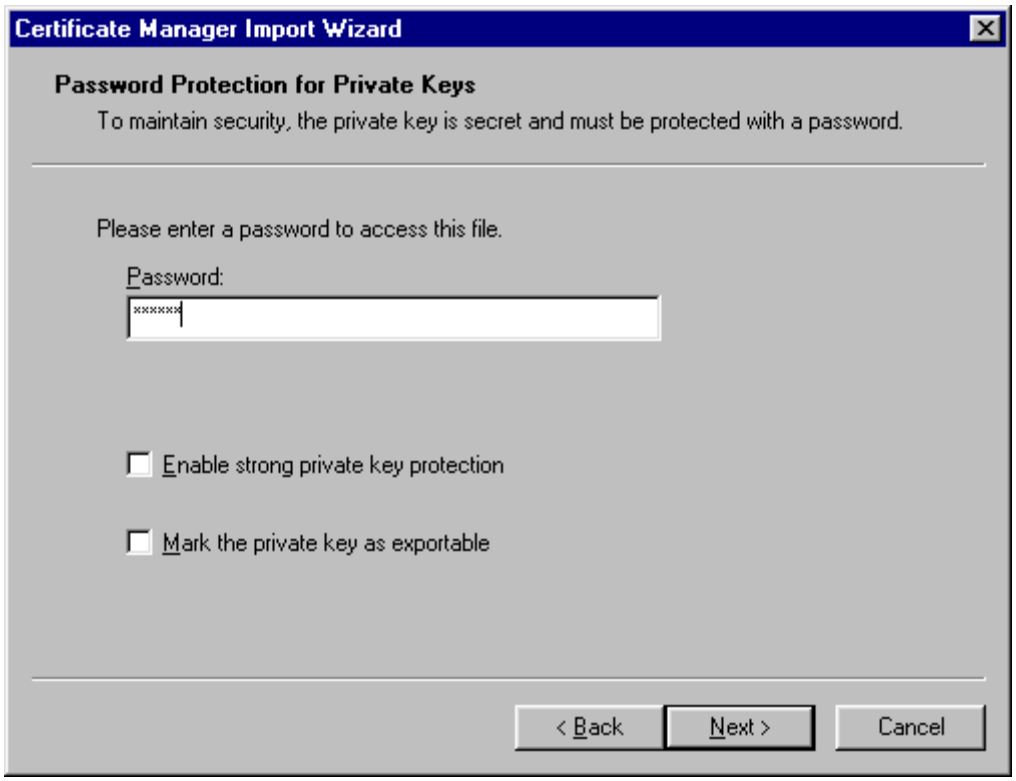


Hình 6



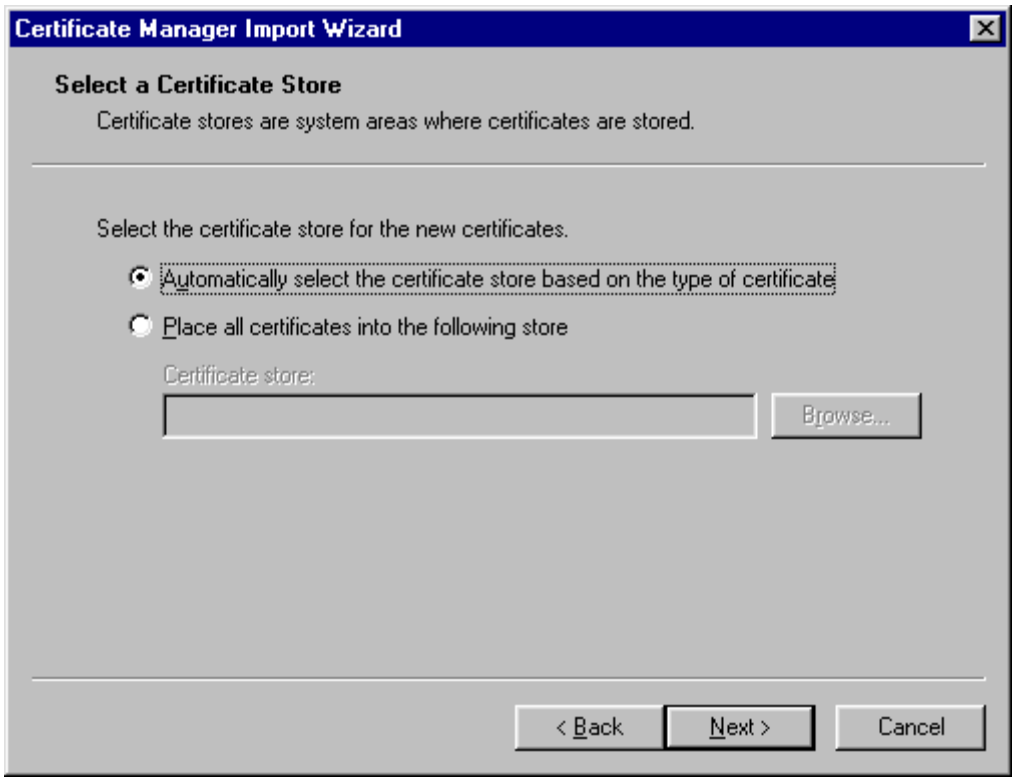
Hình 7

Chọn **Next>** để tiếp tục, xuất hiện hộp thoại như Hình 8.



Hình 8

Hộp thoại yêu cầu bạn vào mật khẩu giải mã tệp chứng chỉ (vào đúng mật khẩu khi bạn chuyển đổi định dạng PKCS#12). Chọn **Next>** để tiếp tục, xuất hiện hộp thoại như Hình 9.



Hình 9

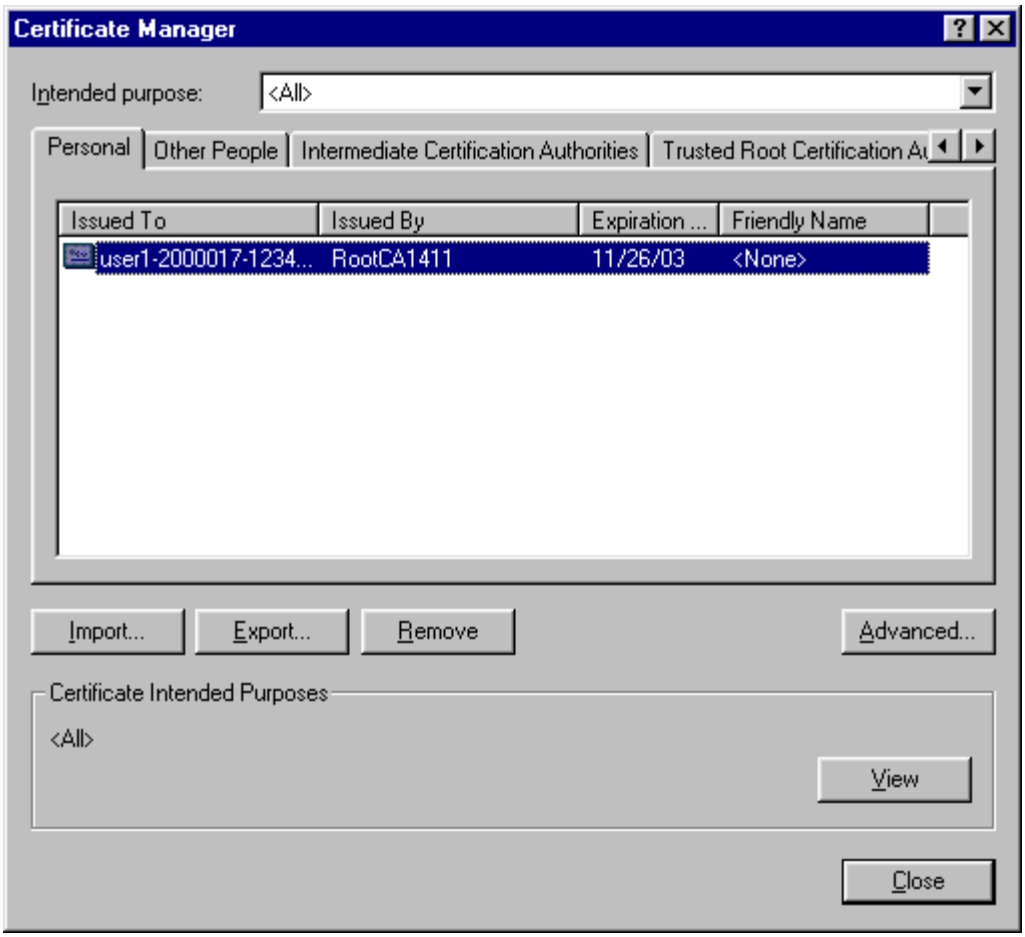
Hộp thoại yêu cầu lựa chọn nơi lưu trữ chứng chỉ của bạn, ở đây chọn mục **Automatically select the certificate store based on the type of certificate**. Chọn **Next>** để tiếp tục, xuất hiện hộp thoại như Hình 10.

Hình 10

Chọn **Finish** để kết thúc, nếu thành công sẽ xuất hiện hộp thoại như Hình 11.

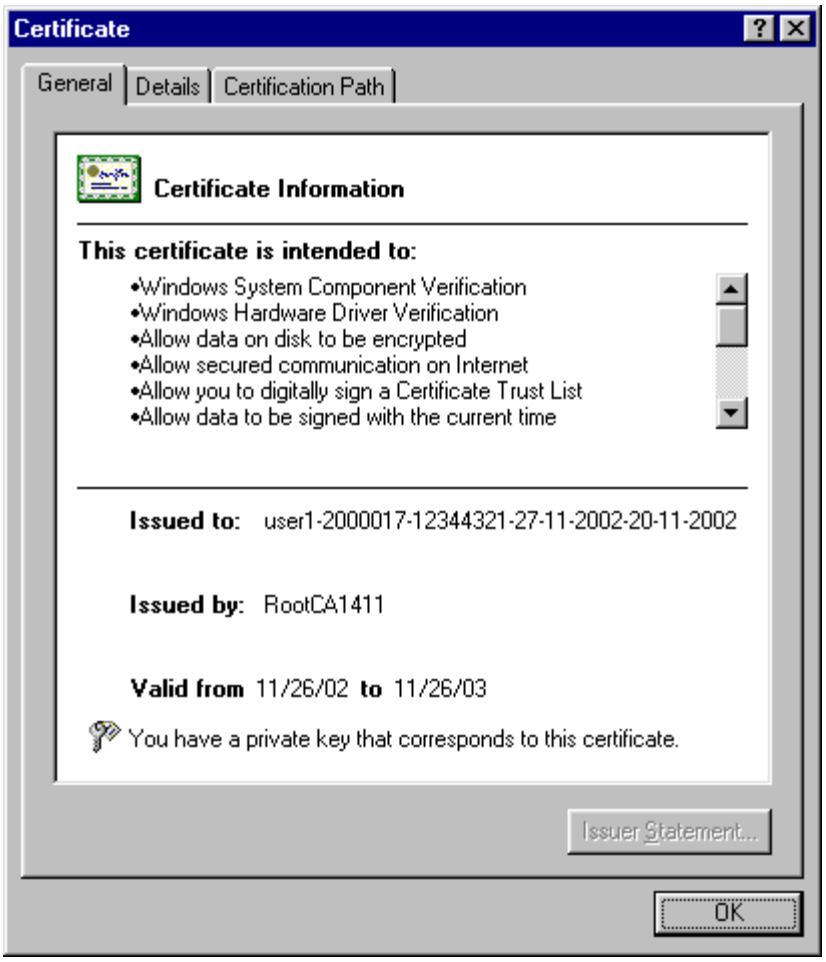
Hình 11

Chọn OK, như vậy bạn đã hoàn thành công việc cài chứng chỉ số vào Outlook. Để kiểm tra lại điều này, bạn vào **Tools => Options =>Security => Digital IDs...** xuất hiện hộp thoại **Certificate Manager** trong mục **Personal** xuất hiện dòng mô tả chứng chỉ của bạn. Ví dụ, chúng tôi cài chứng chỉ của user1 như Hình 12.



Hình 12

Để hiển thị chi tiết nội dung chứng chỉ bạn chọn **View**, khi đó xuất hiện hộp thoại có dạng như Hình 13.

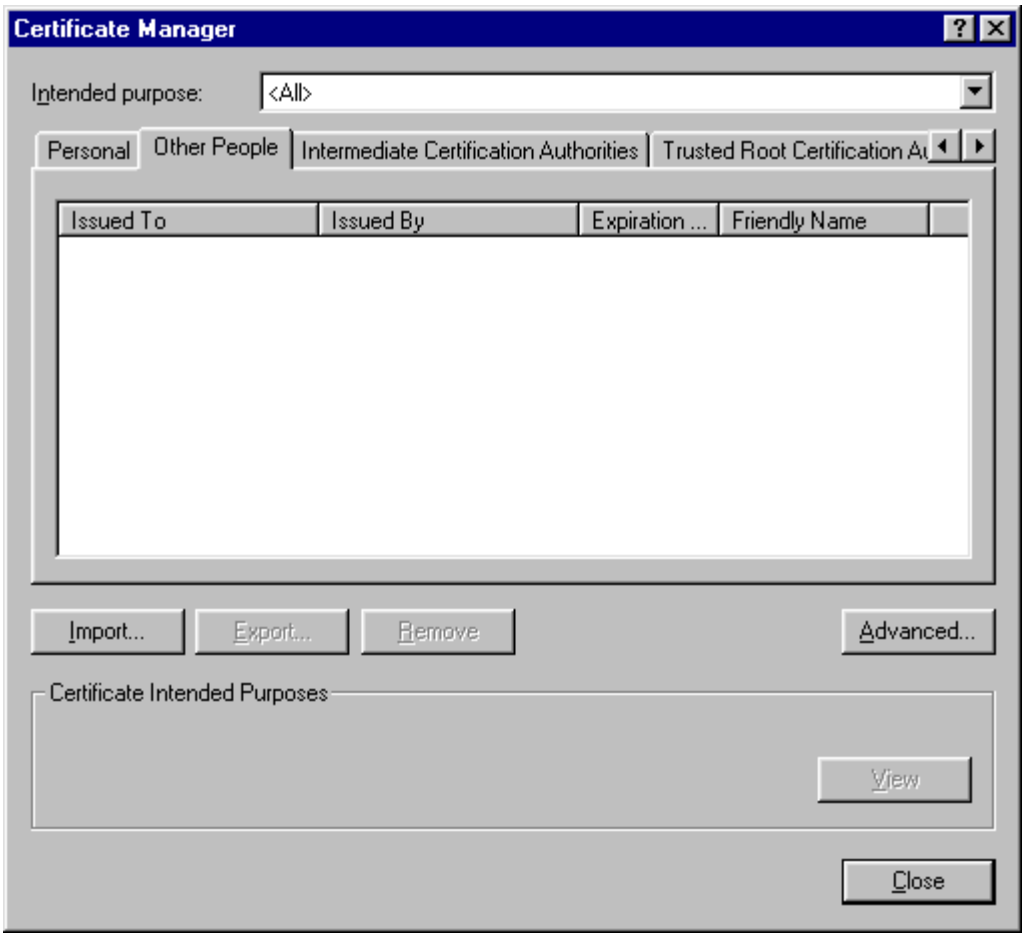


Hình 13

Đến đây bạn có thể thực hiện gửi thư điện tử có xác thực và mã hoá với một người sử dụng khác cũng đã cài chứng chỉ số của họ trên Outlook. Chú ý rằng chứng chỉ số của bạn và người sử dụng kia phải được cấp cùng một hệ thống CA, ở đây chúng tôi đang sử dụng hệ thống MyCA để cấp chứng chỉ số, CA phát hành chứng chỉ có tên là RootCA1411.

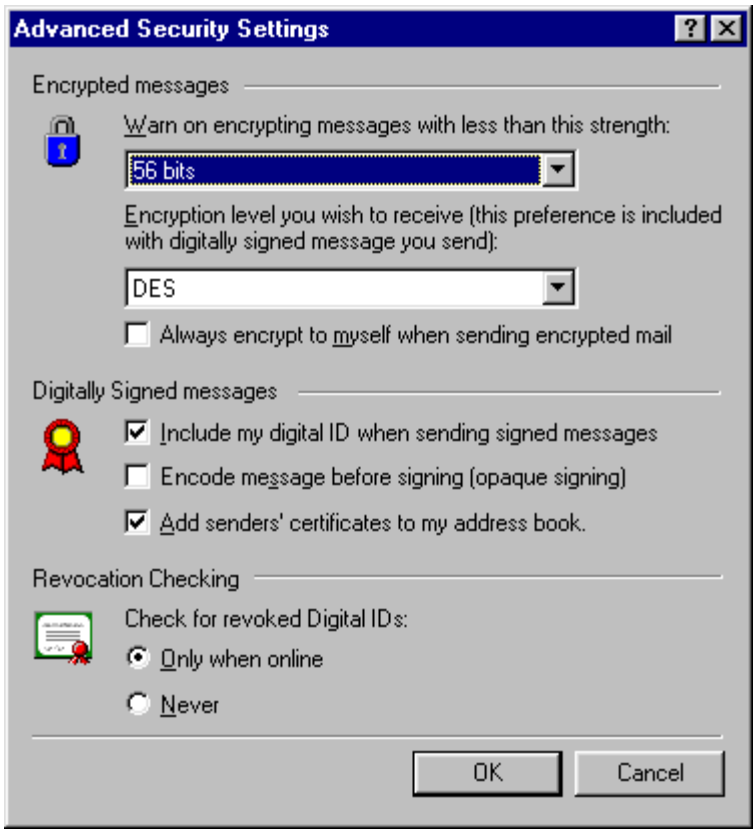
3. Sử dụng chứng chỉ số để xác thực và mã hoá thư điện tử trên Outlook

Trong mục này chúng tôi sẽ trình bày ứng dụng chứng chỉ số để xác thực và mã hoá thư điện tử trên Outlook, với điều kiện là bạn đã thực hiện cài đặt thành công chứng chỉ số của mình vào Outlook (đã trình bày ở trên). Vấn đề đặt ra là bạn chưa có khoá công khai của đối tác để mã hoá, ngược lại đối tác của bạn cũng chưa có khoá công khai của bạn để thực hiện việc kiểm tra chữ ký. Để giải quyết vấn đề này, lần đầu tiên bạn gửi thư điện tử cho đối tác chỉ thực hiện ký lên nội dung đó và chú ý rằng nội dung thư không yêu cầu bảo mật, đồng thời trong Outlook phải đặt tuỳ chọn gửi chứng chỉ của bạn kèm theo. Khi nhận được thư của bạn, đối tác sẽ có chứng chỉ số của bạn, và sau đó đối tác gửi chứng chỉ số của họ bằng cách tương tự như bạn đã làm. Đến đây bạn và đối tác có thể thực hiện gửi thư có xác thực và mã hoá cho nhau. Công việc này sẽ được trình bày chi tiết dưới đây.



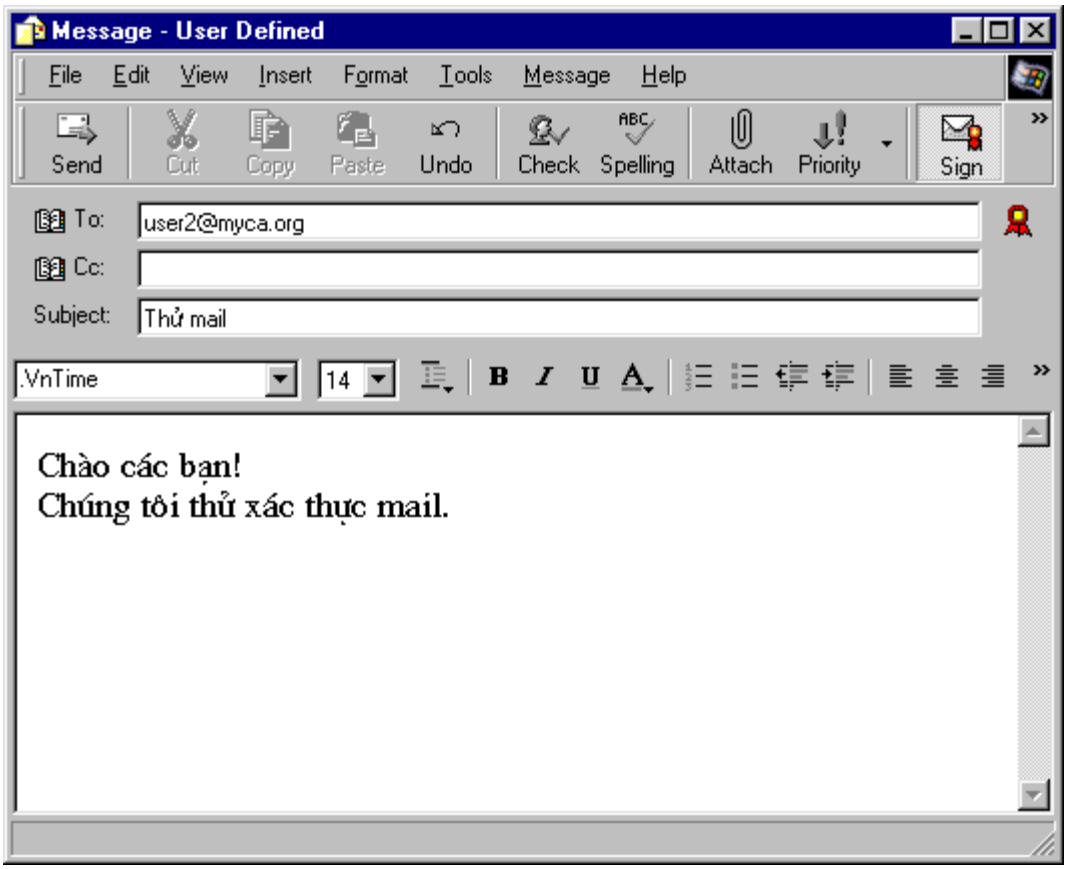
Hình 14

Trong Hình 14, mục **Other People** trong hộp thoại **Certificate Manager** không có chứng chỉ của đối tác mà bạn cần trao đổi thư điện tử mật. Lần đầu tiên bạn gửi chứng chỉ của mình cho đối tác bằng cách ký lên bức thư (chọn **Sign**), và sau đó chọn **Send** để gửi. Chú ý trên Outlook cả bên gửi và bên nhận phải được đặt 2 tuỳ chọn: **Include my digital ID when sending signed messages** và **Add the senders' certificates to my address book** như Hình 15. Hộp thoại còn cho phép thiết lập thuật toán mã hoá (ở đây chúng tôi đặt là DES).



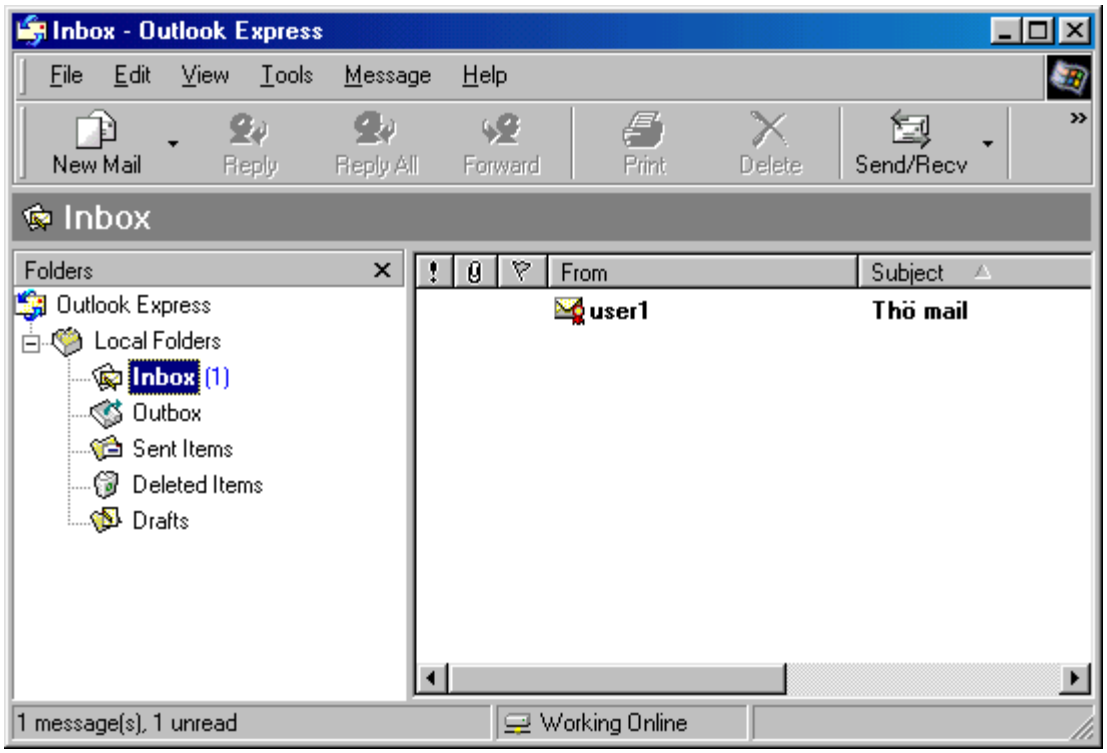
Hình 15

Trong ví dụ của chúng tôi, đối tác cần gửi có địa chỉ email là user2@myca.org như Hình 16.



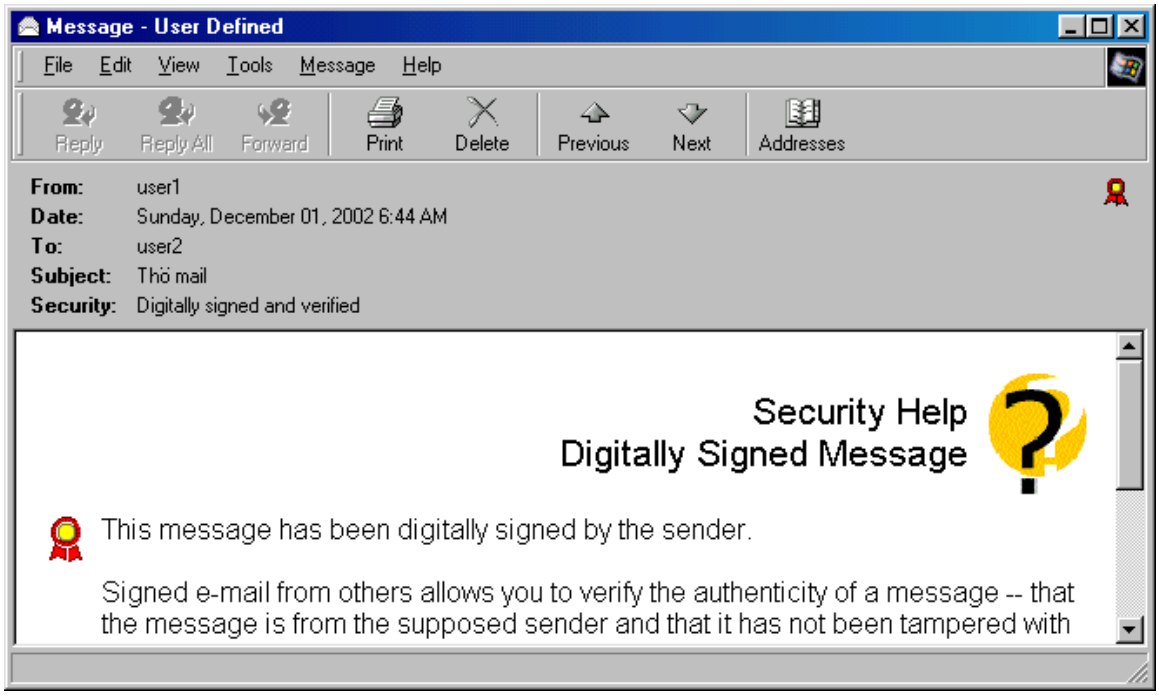
Hình 16

Chọn **Send** và sau đó chọn **Send/Recv**. Khi đó đối tác mở Outlook, nhận được một thư trong **Inbox** như Hình 17.



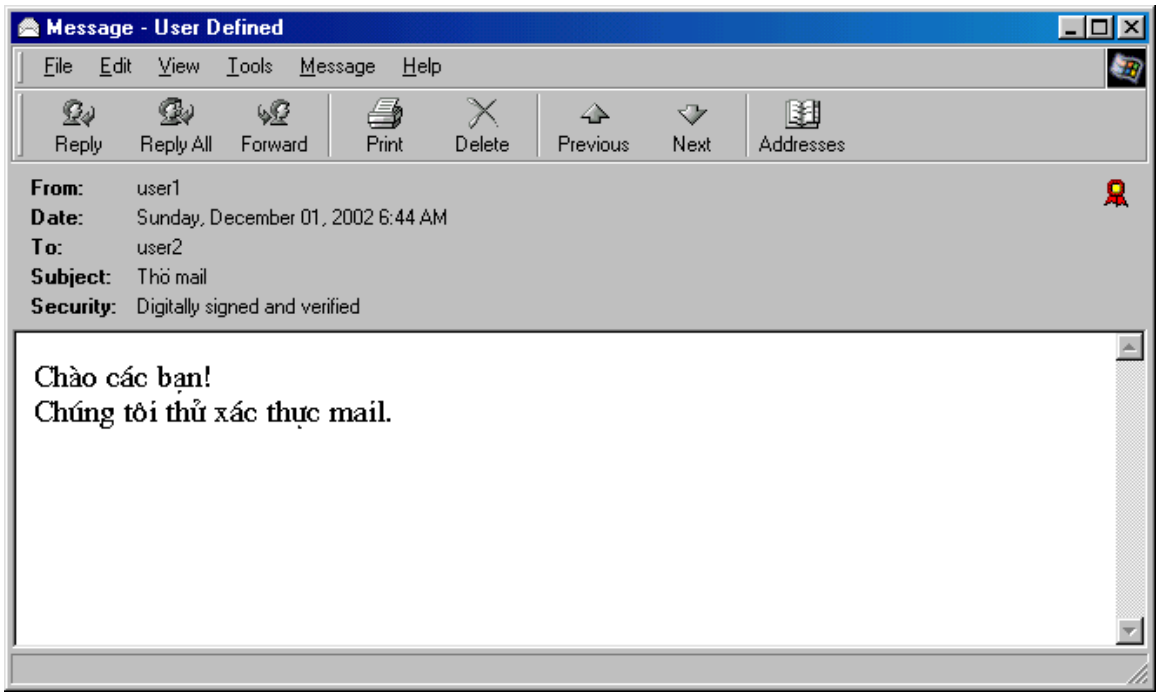
Hình 17

Mở thư nhận được từ user1, xuất hiện hộp thoại như Hình 18.



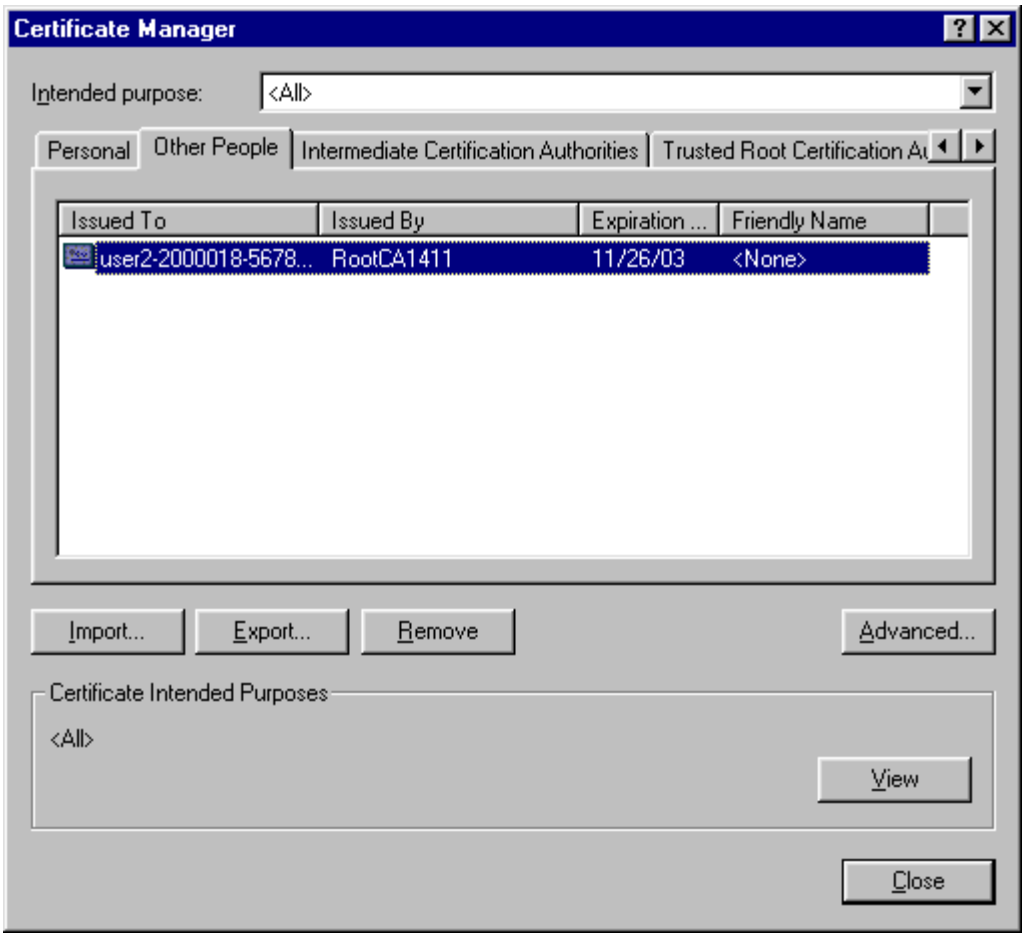
Hình 18

Ngoài những thông tin về người gửi và nhận thì có thêm mục **Security: Digitally signed and verified**, tức là bức thư này đã được ký từ người gửi và được kiểm tra là đúng. Muốn xem nội dung thư, chọn **Continue**, sẽ hiển thị nội dung thư như Hình 19.



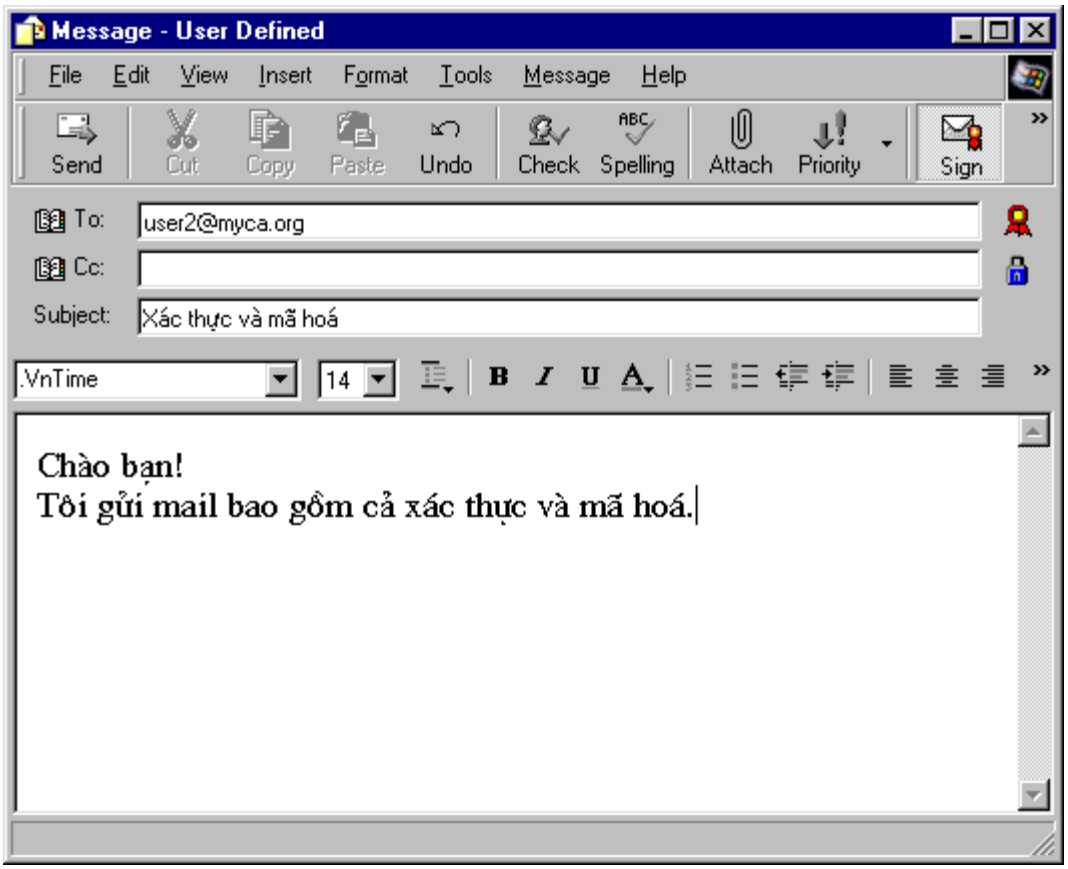
Hình 19

Sau đó đối tác sẽ gửi chứng chỉ của họ cho bạn như bạn đã từng làm ở trên. Để kiểm tra rằng chứng chỉ của đối tác đã có trong Outlook hay chưa bạn chọn mục **Other People** của hộp thoại **Certificate Manager**, mục này chứa chứng chỉ của các đối tác của bạn. Ví dụ như Hình 20.



Hình 20

Đến đây cả 2 bên đều có chứng chỉ của nhau, và có thể thực hiện gửi thư có xác thực và bảo mật cho nhau. Ví dụ bạn có thể soạn một thư điện tử như Hình 21.

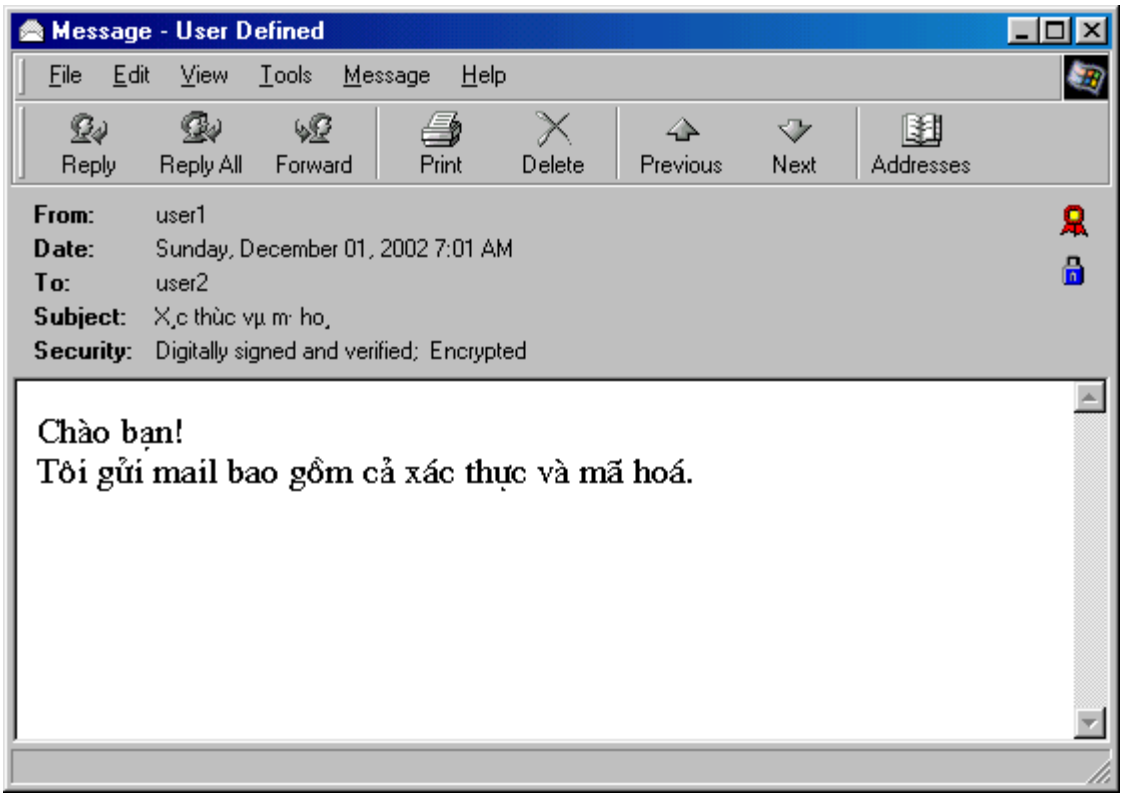


Hình 21

Chú ý, để gửi thư có xác thực và mã hoá bạn chọn cả 2 tuỳ chọn **Sign** và **Encrypt**. Chọn **Send** và **Send/Recv**, để gửi tới đối tác của bạn. Bên nhận nhận được thư, hiển thị nội dung có dạng như Hình 22.

Hình 22

Để xem nội dung thư, chọn **Continue** (giải mã), khi đó xuất hiện nội dung thư có dạng như Hình 23.



Hình 23

Đây là bức thư gửi có xác thực và mã hoá (trong mục **Security: Digital signed and verified; Encrypted**).

4. Cập nhật danh sách các chứng chỉ đã huỷ bỏ

Một người dùng có thể huỷ bỏ chứng chỉ của mình (với rất nhiều lý do), để huỷ bỏ một chứng chỉ bạn có thể tham khảo chương 6. Do vậy, thông thường mỗi phiên làm việc bạn nên cập nhật lại danh sách các chứng chỉ đã bị huỷ bỏ do CA của bạn phát hành, điều này giúp bạn có thể biết được trạng thái chứng chỉ của đối tác (chứng chỉ đã bị huỷ bỏ hay chưa?), bạn tham khảo chương 7. Khi đã lấy được danh sách các chứng chỉ huỷ bỏ (giả sử tệp chain.crl) thì các bước thực hiện cập nhật vào Outlook tương tự như đối với việc cài một chứng chỉ vào Outlook, chúng tôi sẽ không trình bày lại ở đây.

Trong mục này chúng tôi giả sử rằng người sử dụng có tên user1 với địa chỉ email là user1@myca.org đã huỷ bỏ chứng chỉ. Trước khi làm việc, người sử dụng user2 tiến hành cập nhật danh sách các chứng chỉ bị huỷ bỏ do hệ thống CA của mình phát hành. Người sử dụng user2 sử dụng Outlook soạn thư và gửi thư có xác thực và mã hoá cho user1, sau chọn **Send** để gửi cho user1 thì sẽ xuất hiện thông báo như Hình 24.

Hình 24

Outlook thông báo là chứng chỉ của user1 đã bị huỷ bỏ, user2 sẽ không thể thực hiện gửi thư có xác thực và bảo mật cho user1 được nữa. Chú ý rằng, user2 vẫn có thể gửi thư kèm theo chữ ký cho user1 bởi chứng chỉ của user2 chưa bị huỷ bỏ nên user1 vẫn có thể kiểm tra chữ ký của user2.

Ngược lại, khi user1 gửi thư có xác thực và mã hoá cho user2 thì điều gì sẽ xảy ra khi mà chứng chỉ của user1 đã bị huỷ bỏ?. Khi nhận được thư của user1, user2 hiển thị nội dung sẽ có dạng như Hình 25.

Hình 25

Trong mục **Security:** có thông báo **Digitally signed - signing digital ID is revoked**, tức là bức thư này đã được ký bởi một chứng chỉ đã huỷ bỏ. Chú ý rằng, nếu user1 gửi thư chỉ chọn Encrypt (mà không ký lên thư đó) thì user2 vẫn đọc được bình thường (tức là giải mã được), bởi chứng chỉ của user2 chưa bị huỷ bỏ.