UNIT V

COMMUNICATION SECURITY

Communication Security: Authentications Protocols, E-mail Security, Web security, Social Issues, SDL based protocol verification and validation, Internet protocol, SDL based interoperability testing of CSMA/CD and CSMA/CA protocol using Bridge, Scalability testing-Applications.

5.1 AUTHENTICATION PROTOCOL

- An authentication protocol is a type of computer communications protocol or cryptographic protocol specifically designed for transfer of authentication data between two entities.
- It allows to authenticate the connecting entity (e.g. Client connecting to a Server) as well as authenticate itself to the connecting entity (Server to a client) by declaring the type of information needed for authentication as well as syntax.
- It is the most important layer of protection needed for secure communication within computer networks.

The general model that all authentication protocols use is this. Alice starts out by sending a message either to Bob or to a trusted KDC (Key Distribution Center), which is expected to be honest. Several other message exchanges follow in various directions. As these messages are being sent Trudy may intercept, modify, or replay them in order to trick Alice and Bob or just to gum up the works.

Nevertheless, when the protocol has been completed, Alice is sure she is talking to Bob and Bob is sure he is talking to Alice. Furthermore, in most of the protocols, the two of them will also have established a secret session key for use in the upcoming conversation. In practice, for performance reasons, all data traffic is encrypted using symmetric-key cryptography (typically AES or triple DES), although public-key cryptography is widely used for the authentication protocols themselves and for establishing the session key.

The point of using a new, randomly-chosen session key for each new connection is to minimize the amount of traffic that gets sent with the users' secret keys or public keys, to reduce the amount of cipher text an intruder can obtain, and to minimize the damage done if a process crashes and its core dump falls into the wrong hands. Hopefully, the only key present then will be the session key. All the permanent keys should have been carefully zeroed out after the session was established.

The authentication is based on the following using:

- Shared secret key
- Establishing a shared key: the Diffie-Hellman key exchange
- Key distribution center
- Kerberos
- Public-key cryptography

5.1.1 Authentication Based on a Shared Secret Key

For our first authentication protocol, we will assume that Alice and Bob already share a secret key, KAB . This shared key might have been agreed upon on the telephone or in person, but, in any event, not on the (insecure) network. This protocol is based on a principle found in many authentication protocols: one party sends a random number to the other, who then transforms it in a special way and then returns the result. Such protocols are called challenge-response protocols.

Notation for discussing protocols

- *A, B* are the identities of Alice and Bob.
- R_i*'*sare the challenges, where the subscript identifies the challenger.
- $K_i$are keys, where *i*indicates the owner.
- K_S is the session key.

The message sequence for our first shared-key authentication protocol is illustrated in Fig.. In message 1, Alice sends her identity, A, to Bob in a way that Bob understands. Bob, of course, has no way of knowing whether this message came from Alice or from Trudy, so he chooses a challenge, a large random number, RB , and sends it back to "Alice" as message 2, in plaintext. Random numbers used just once in challenge-response protocols like this one are called nonces. Alice then encrypts the message with the key she shares with Bob and sends the ciphertext, KAB (RB), back in message 3. When Bob sees this message, he immediately knows that it came from Alice because Trudy does not know KAB and thus could not have generated it. Furthermore, since RB was chosen randomly from a large space (say, 128-bit random numbers), it is very

unlikely that Trudy would have seen RB and its response from an earlier session. It is equally unlikely that she could guess the correct response to any challenge.

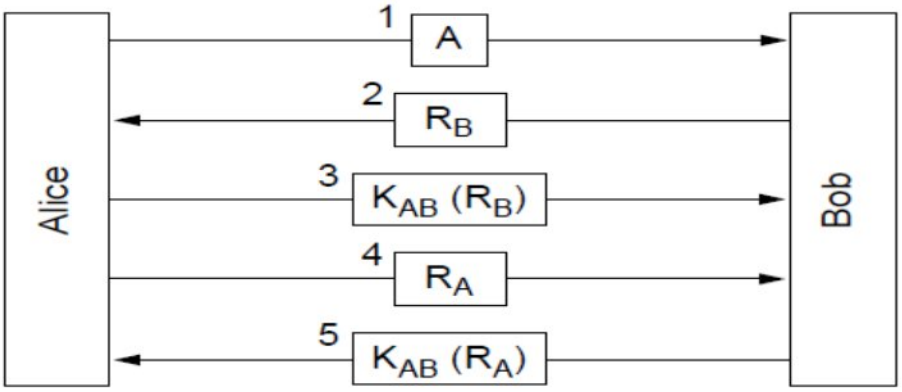


Fig 5.1 Two-way authentication using a challenge-response protocol

At this point, Bob is sure he is talking to Alice, but Alice is not sure of anything. For all Alice knows, Trudy might have intercepted message 1 and sent back RB in response. Maybe Bob died last night. To find out to whom she is talking, Alice picks a random number, RA and sends it to Bob as plaintext, in message 4. When Bob responds with KAB (RA), Alice knows she is talking to Bob. If they wish to establish a session key now, Alice can pick one, KS , and send it to Bob encrypted with KAB. The protocol of Fig. 5.1contains five messages. Let us see if we can be clever and eliminate some of them. One approach is illustrated in Fig. 5.2. Here Alice initiates the challenge response protocol instead of waiting for Bob to do it. Similarly, while he is responding to Alice's challenge, Bob sends his own. The entire protocol can be reduced to three messages instead of five.

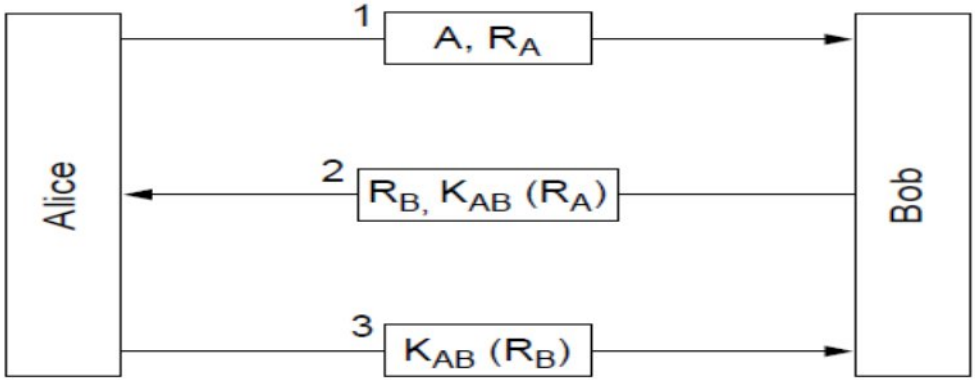


Fig 5.2 A shortened two-way authentication protocol

Is this new protocol an improvement over the original one? In one sense it is: it is shorter. Unfortunately, it is also wrong. Under certain circumstances, Trudy can defeat this protocol by using what is known as a reflection attack. In particular, Trudy can break it if it is possible to open multiple sessions with Bob at once. This situation would be true, for example, if Bob is a bank and is prepared to accept many simultaneous connections from teller machines at once. Trudy's reflection attack is shown in Fig. 5.3. It starts out with Trudy claiming she is Alice and sending RT . Bob responds, as usual, with his own challenge, RB . Now Trudy is stuck. What can she do? She does not know KAB (RB).

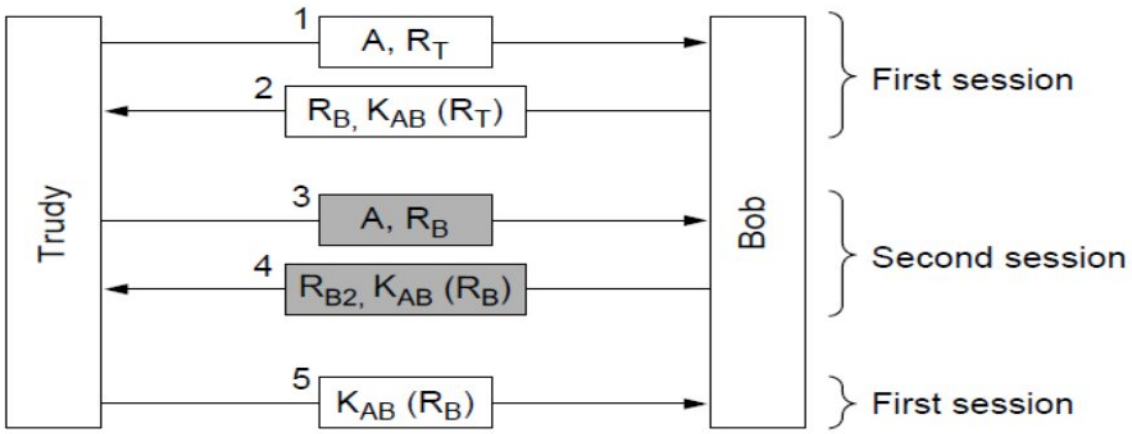


Fig 5.3 The reflection attack.

She can open a second session with message 3, supplying the RB taken from message 2 as her challenge. Bob calmly encrypts it and sends back KAB (RB)in message 4. We have shaded the messages on the second session to make them stand out. Now Trudy has the missing information, so she can complete the first session and abort the second one. Bob is now convinced that Trudy is Alice, so when she asks for her bank account balance, he gives it to her without question. Then when she asks him to transfer it all to a secret bank account in Switzerland, he does so without a moment's hesitation.

General design rules

1. Have initiator prove who she is before responder
2. Initiator, responder use different keys
3. Draw challenges from different sets
4. Make protocol resistant to attacks involving second parallel session

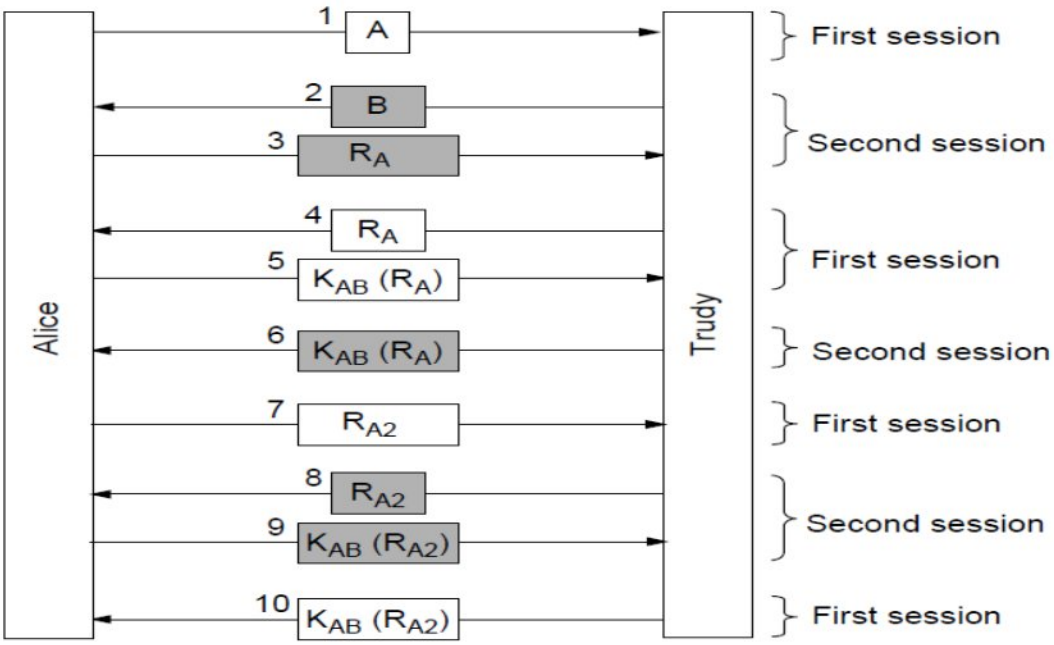


Fig 5.4 A reflection attack on the protocol

Trudy somehow subvert this protocol, because she cannot force either party to encrypt or hash a value of her choice, as happened. Both HMACs include values chosen by the sending party, something which Trudy cannot control. Using HMACs is not the only way to use this idea. An alternative scheme that is often used instead of computing the HMAC over a series of items is to encrypt the items sequentially using cipher block chaining.

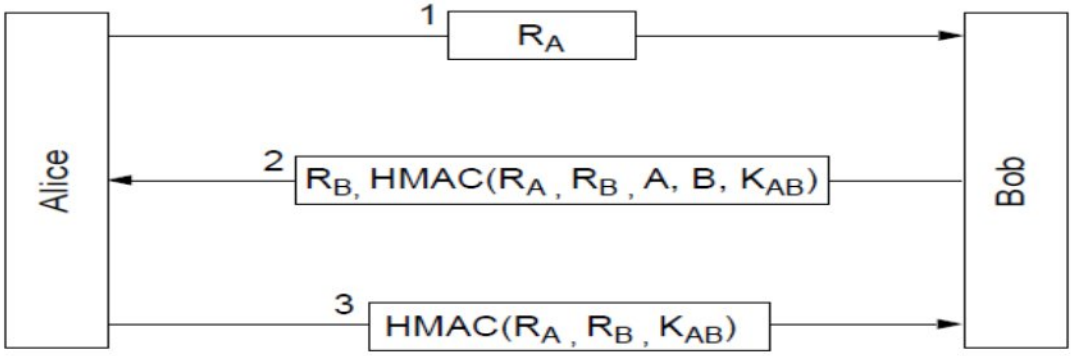


Fig 5.5 Authentication using HMACs

5.1.2 Establishing a Shared Key: The Diffie-Hellman Key Exchange

The protocol that allows strangers to establish a shared secret key is called the DiffieHellman key exchange (Diffie and Hellman, 1976) and works as follows. Alice and Bob have to agree on two large numbers, n and g, where n is a prime, (n - 1)/2 is also a prime and certain conditions apply to g. These numbers may be public, so either one of them can just pick n and g or tell the other openly. Now Alice picks a large (say, 512-bit) number, x, and keeps it secret. Similarly, Bob picks a large secret number, y.

Alice initiates the key exchange protocol by sending Bob a message containing (n, g, g x mod n), as shown in Fig. 5-1.4. Bob responds by sending Alice a message containing g y mod n. Now Alice raises the number Bob sent her to the xth power modulo n to get (g y mod n) x mod n. Bob performs a similar operation to get (g x mod n) y mod n. By the laws of modular arithmetic, both calculations yield g xy mod n. Lo and behold, Alice and Bob suddenly share a secret key, g xy mod n

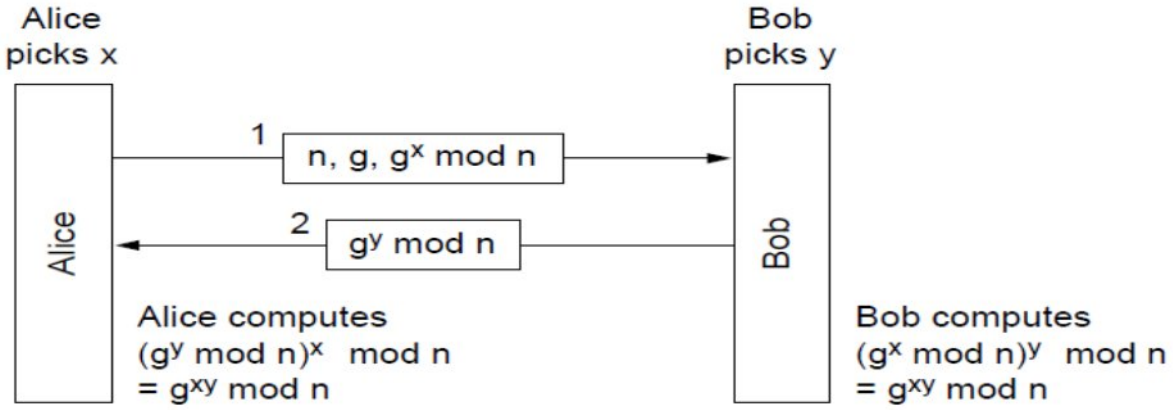


Fig 5.6 The Diffie-Hellman key exchange

Trudy, of course, has seen both messages. She knows g and n from message 1. If she could compute x and y, she could Fig. out the secret key. The trouble is, given only g x mod n, she cannot find x. No practical algorithm for computing discrete logarithms modulo a very large prime number is known.

To make the above example more concrete, we will use the (completely unrealistic) values of n = 47 and g = 3. Alice picks x = 8 and Bob picks y = 10. Both of these are kept secret. Alice's message to Bob is (47, 3, 28) because 3 8 mod 47 is 28. Bob's message to Alice is (17). Alice computes 17 8 mod 47, which is 4. Bob computes 28 10 mod 47, which is 4. Alice and Bob have independently determined that the secret key is now 4. Trudy has to solve the equation 3 x mod 47 = 28, which can be done by exhaustive search for small numbers like this, but not when all the numbers are hundreds of bits long. All currently-known algorithms simply take too long, even on massively parallel supercomputers.

Despite the elegance of the Diffie-Hellman algorithm, there is a problem: when Bob gets the triple (47, 3, 28), how does he know it is from Alice and not from Trudy? There is no way he can know. Unfortunately, Trudy can exploit this fact to deceive both Alice and Bob, as illustrated in Fig. 5.1.5. Here, while Alice and Bob are choosing x and y, respectively, Trudy picks her own random number, z. Alice sends message 1 intended for Bob. Trudy intercepts it and sends message 2 to Bob, using the correct g and n (which are public anyway) but with her own z instead of x. She also sends message 3 back to Alice. Later Bob sends message 4 to Alice, which Trudy again intercepts and keeps.

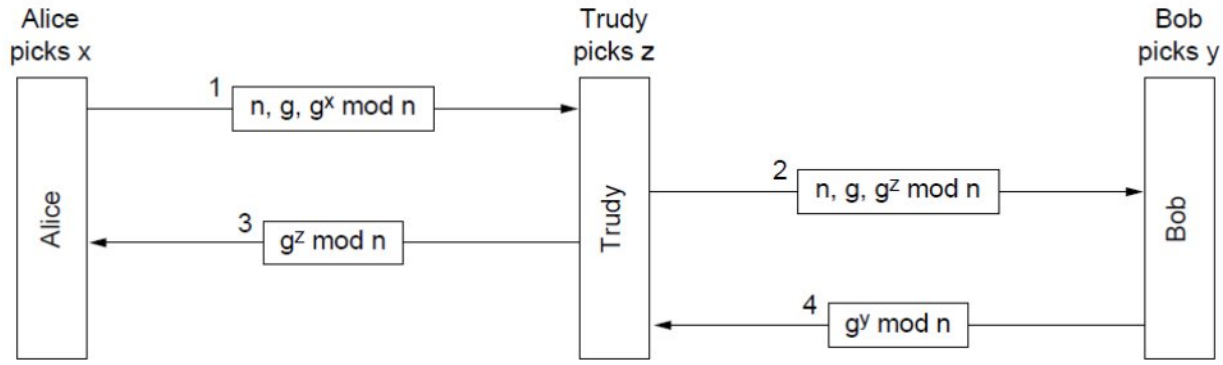


Fig 5.7 The man-in-the-middle attack

Now everybody does the modular arithmetic. Alice computes the secret key as g xz mod n, and so does Trudy (for messages to Alice). Bob computes g yz mod n and so does Trudy (for messages to Bob). Alice thinks she is talking to Bob so she establishes a session key (with Trudy). So does Bob. Every message that Alice sends on the encrypted session is captured by Trudy, stored, modified if desired, and then (optionally) passed on to Bob. Similarly, in the other direction. Trudy sees everything and can modify all messages at will, while both Alice and Bob are under the illusion that they have a secure channel to one another. This attack is known as the bucket brigade attack, because it vaguely resembles an old-time volunteer fire department

passing buckets along the line from the fire truck to the fire. It is also called the man-in-the middle attack.

5.1.3 Authentication Using a Key Distribution Center

Setting up a shared secret with a stranger almost worked, but not quite. On the other hand, it probably was not worth doing in the first place (sour grapes attack). To talk to n people this way, you would need n keys. For popular people, key management would become a real burden, especially if each key had to be stored on a separate plastic chip card.

A different approach is to introduce a trusted key distribution center (KDC). In this model, each user has a single key shared with the KDC. Authentication and session key management now goes through the KDC. The simplest known KDC authentication protocol involving two parties and a trusted KDC is depicted in Fig. 5.8.

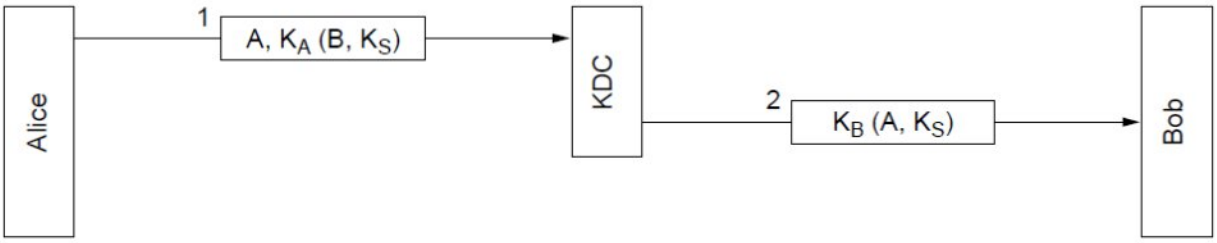


Fig 5.8 A first attempt at an authentication protocol using a KDC

The protocol begins with Alice telling the KDC that she wants to talk to Bob. This message contains a large random number, RA , as a nonce. The KDC sends back message 2 containing Alice's random number, a session key, and a ticket that she can send to Bob. The point of the random number, RA , is to assure Alice that message 2 is fresh, and not a replay. Bob's identity is also enclosed in case Trudy gets any funny ideas about replacing B in message 1 with her own identity so the KDC will encrypt the ticket at the end of message 2 with KT instead of KB . The ticket encrypted with KB is included inside the encrypted message to prevent Trudy from replacing it with something else on the way back to Alice.

Alice now sends the ticket to Bob, along with a new random number, RA2 , encrypted with the session key, KS . In message 4, Bob sends back KS (RA2 - 1) to prove to Alice that she is talking to the real Bob. Sending back KS (RA2) would not have worked, since Trudy could just have stolen it from message 3.

After receiving message 4, Alice is now convinced that she is talking to Bob and that no replays could have been used so far. After all, she just generated RA2 a few milliseconds ago.

The purpose of message 5 is to convince Bob that it is indeed Alice he is talking to, and no replays are being used here either. By having each party both generate a challenge and respond to one, the possibility of any kind of replay attack is eliminated.

Although this protocol seems pretty solid, it does have a slight weakness. If Trudy ever manages to obtain an old session key in plaintext, she can initiate a new session with Bob by replaying the message 3 corresponding to the compromised key and convince him that she is Alice (Denning and Sacco, 1981). This time she can plunder Alice's bank account without having to perform the legitimate service even once. Needham and Schroeder later published a protocol that corrects this problem (Needham and Schroeder, 1987). In the same issue of the same journal, Otway and Rees (1987) also published a protocol that solves the problem in a shorter way. Fig. 5.9.shows a slightly modified Otway-Rees protocol.

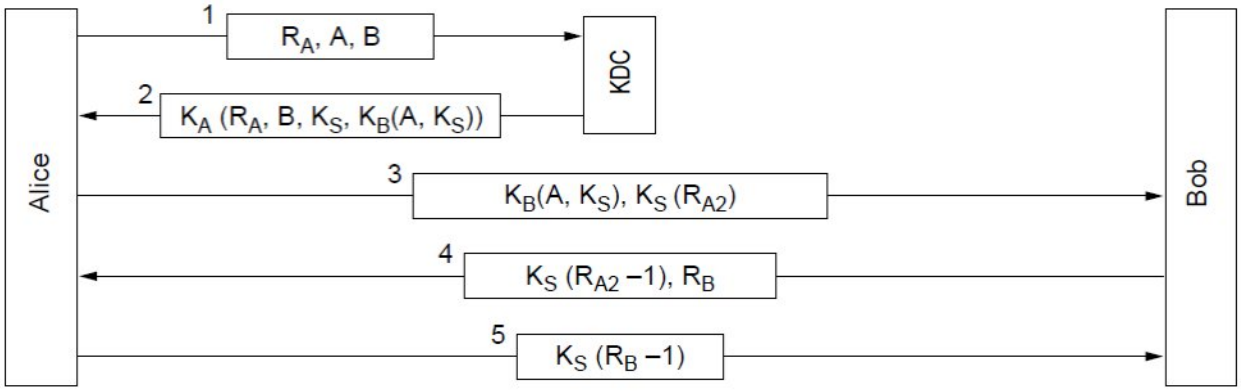


Fig 5.9 The Needham-Schroeder authentication protocol

In the Otway-Rees protocol, Alice starts out by generating a pair of random numbers, R, which will be used as a common identifier, and RA , which Alice will use to challenge Bob. When Bob gets this message, he constructs a new message from the encrypted part of Alice's message and an analogous one of his own. Both the parts encrypted with KA and KB identify Alice and Bob, contain the common identifier, and contain a challenge.

The KDC checks to see if the R in both parts is the same. It might not be because Trudy tampered with R in message 1 or replaced part of message 2. If the two Rs match, the KDC believes that the request message from Bob is valid. It then generates a session key and encrypts it twice, once for Alice and once for Bob. Each message contains the receiver's random number, as proof that the KDC, and not Trudy, generated the message. At this point both Alice and Bob are in possession of the same session key and can start communicating. The first time they exchange data messages, each one can see that the other one has an identical copy of KS , so the authentication is then complete.

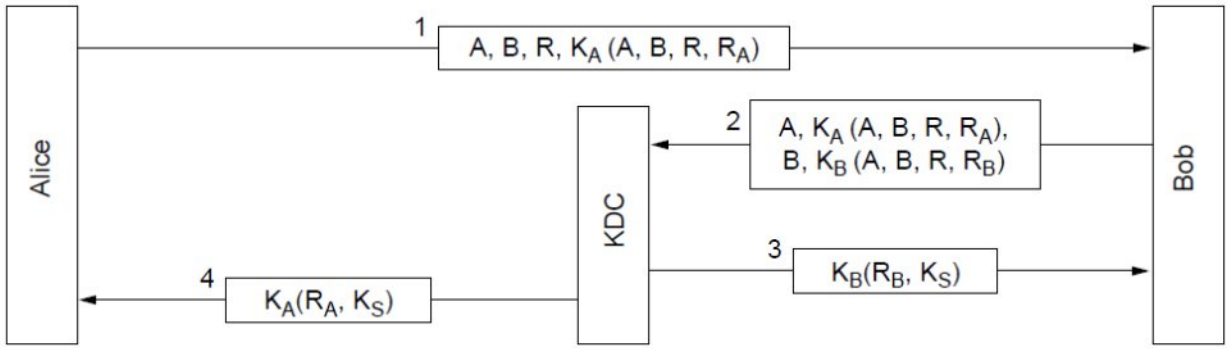


Fig 5.10 The Otway-Rees authentication protocol (slightly simplified).

5.1.4 Authentication Using Kerberos

An authentication protocol used in many real systems (including Windows 2000) is Kerberos, which is based on a variant of Needham-Schroeder. It is named for a multiheaded dog in Greek mythology that used to guard the entrance to Hades (presumably to keep undesirables out). Kerberos was designed at M.I.T. to allow workstation users to access network resources in a secure way. Its biggest difference from Needham-Schroeder is its assumption that all clocks are fairly well synchronized. The protocol has gone through several iterations. V4 is the version most widely used in industry, so we will describe it. Afterward, we will say a few words about its successor, V5. For more information, see (Steiner et al., 1988).

Kerberos involves three servers in addition to Alice (a client workstation):

- Authentication Server (AS): verifies users during login•
- Ticket-Granting Server (TGS): issues "proof of identity tickets"•
- Bob the server: actually does the work Alice wants performed•

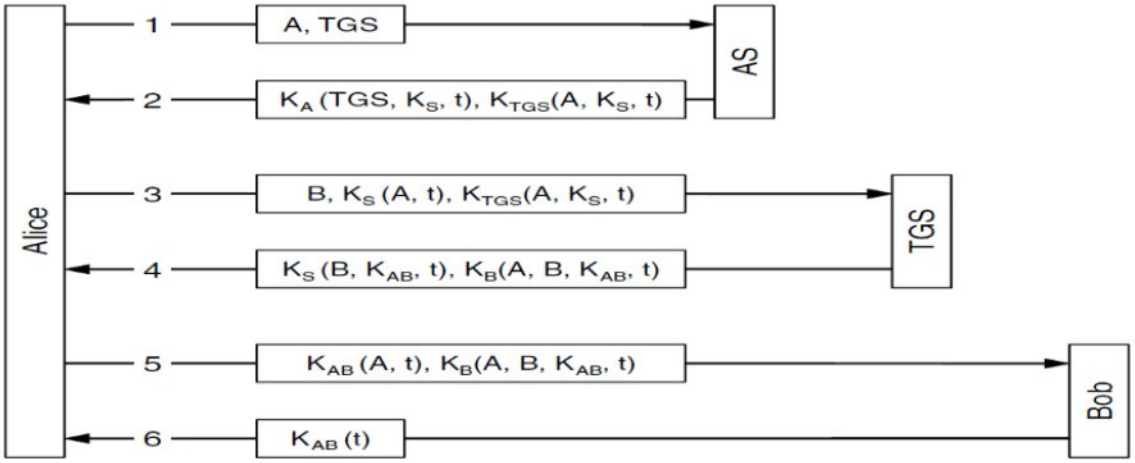


Fig 5.11 The operation of Kerberos V5

AS is similar to a KDC in that it shares a secret password with every user. The TGS's job is to issue tickets that can convince the real servers that the bearer of a TGS ticket really is who he or she claims to be.

To start a session, Alice sits down at an arbitrary public workstation and types her name. The workstation sends her name to the AS in plaintext. What comes back is a session key and a ticket, KTGS(A, KS), intended for the TGS. These items are packaged together and encrypted using Alice's secret key, so that only Alice can decrypt them. Only when message 2 arrives does the workstation ask for Alice's password. The password is then used to generate KA in order to decrypt message 2 and obtain the session key and TGS ticket inside it. At this point, the workstation overwrites Alice's password to make sure that it is only inside the workstation for a few milliseconds at most. If Trudy tries logging in as Alice, the password she types will be wrong and the workstation will detect this because the standard part of message 2 will be incorrect.

5.1.5 Authentication Using Public-Key Cryptography

Mutual authentication can also be done using public-key cryptography. To start with, Alice needs to get Bob's public key. If a PKI exists with a directory server that hands out certificates for public keys, Alice can ask for Bob's, as shown in Fig. 5.11. as message 1. The reply, in message 2, is an X.509 certificate containing Bob's public key. When Alice verifies that the signature is correct, she sends Bob a message containing her identity and a nonce.

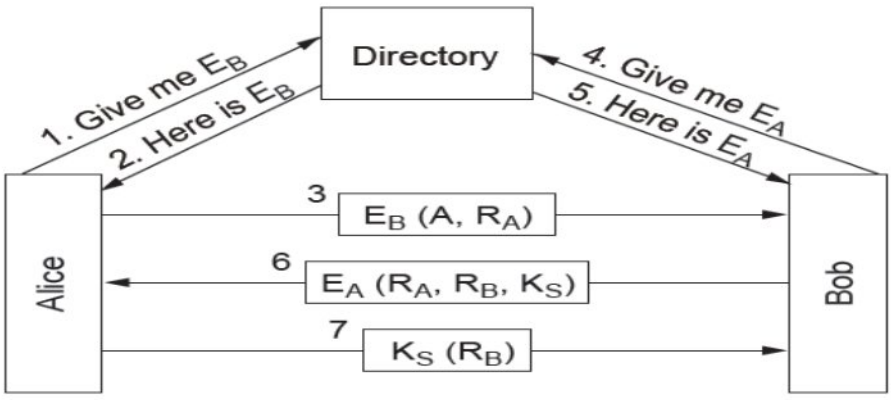


Fig 5.12 Mutual authentication using public-key cryptography

When Bob receives this message, he has no idea whether it came from Alice or from Trudy, but he plays along and asks the directory server for Alice's public key (message 4) which he soon gets (message 5). He then sends Alice a message containing Alice's RA , his own nonce, RB , and a proposed session key, KS , as message 6.

When Alice gets message 6, she decrypts it using her private key. She sees RA in it, which gives her a warm feeling inside. The message must have come from Bob, since Trudy has no way of determining RA . Furthermore, it must be fresh and not a replay, since she just sent Bob RA . Alice agrees to the session by sending back message 7. When Bob sees RB encrypted with the session key he just generated, he knows Alice got message 6 and verified RA .

She can fabricate message 3 and trick Bob into probing Alice, but Alice will see an RA that she did not send and will not proceed further. Trudy cannot forge message 7 back to Bob because she does not know RB or KS and cannot determine them without Alice's private key. She is out of luck.

5.2 EMAIL SECURITY

- email is one of the most widely used andregarded network services
- currently message contents are not secure
  - may be inspected either in transit
  - or by suitably privileged users on destination System

Email Security Enhancements

- confidentiality
  - protection from disclosure
- authentication
  - of sender of message
- message integrity
  - protection from modification
- non-repudiation of origin
  - protection from denial by sender

Pretty Good Privacy (PGP)

- widely used de facto secure email
- developed by Phil Zimmermann
- selected best available crypto algs to use
- integrated into a single program
- on Unix, PC, Macintosh and other systems
- originally free, now also have commercialversions available

PGP Operation –Authentication

1. sender creates message
2. use SHA-1 to generate 160-bit hash ofmessage

3. signed hash with RSA using sender'sprivate key, and is attached to message
4. receiver uses RSA with sender's publickey to decrypt and recover hash code
5. receiver verifies received message usinghash of it and compares with decryptedhash code

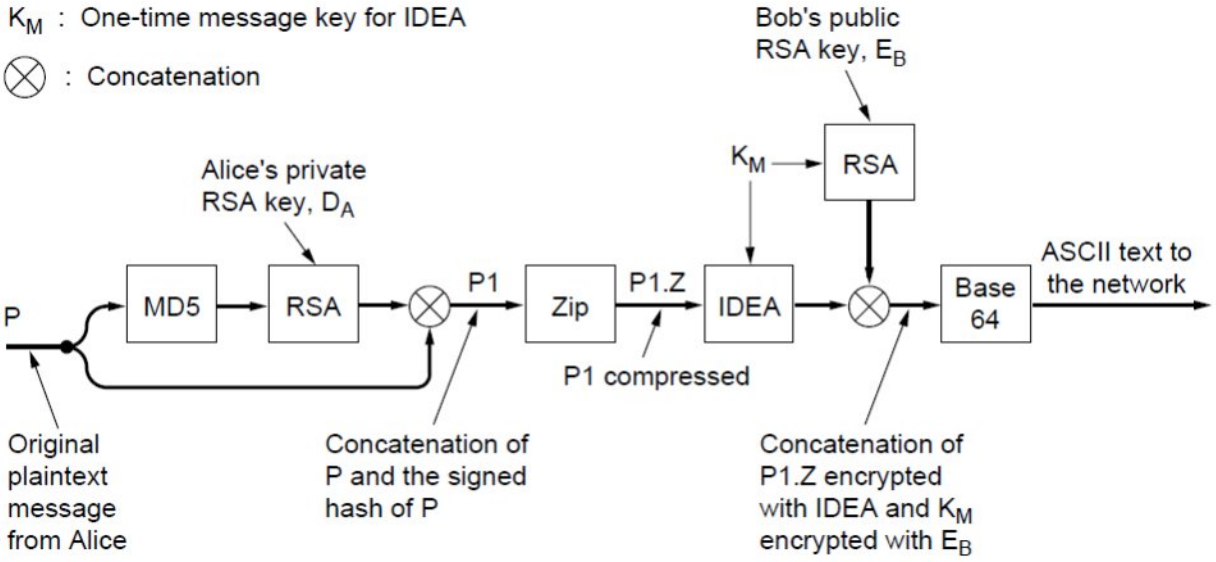


Fig 5.13 PGP in operation for sending a message

PGP Operation –Confidentiality

1. sender generates message and 128-bitrandom number as session key for it
2. encrypt message using CAST-128 / IDEA /3DES in CBC mode with session key
3. session key encrypted using RSA withrecipient's public key, & attached to msg
4. receiver uses RSA with private key todecrypt and recover session key
5. session key is used to decrypt message

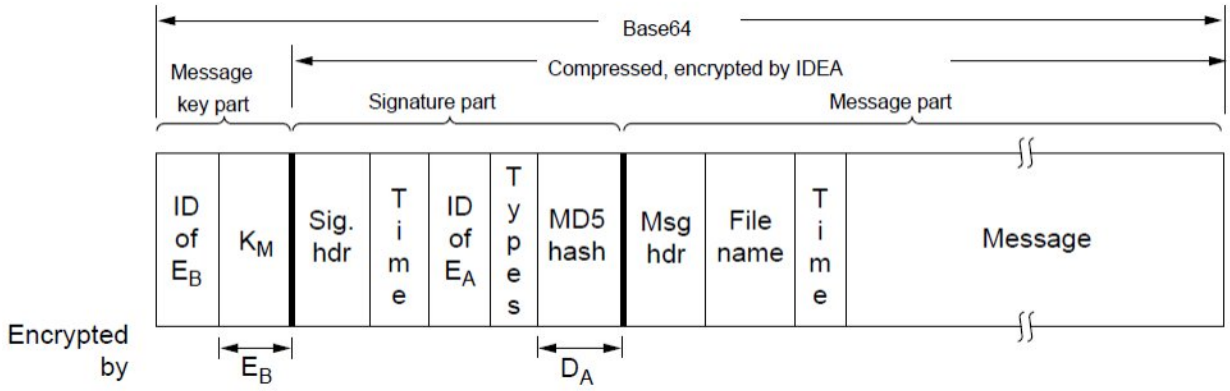


Fig 5.14 A PGP message

PGP Operation – Confidentiality& Authentication

➢ can use both services on same message

- create signature & attach to message
- encrypt both message & signature
- attach RSA/ElGamal encrypted session key

PGP Operation –Compression

- by default PGP compresses messageafter signing but before encrypting
  - so can store uncompressed message &signature for later verification
  - & because compression is non deterministic
- uses ZIP compression algorithm

PGP Operation – EmailCompatibility

- when using PGP will have binary data to send

(encrypted message etc)

- however email was designed only for text
- hence PGP must encode raw binary data intoprintable ASCII characters
- uses radix-64 algorithm
  - maps 3 bytes to 4 printable chars
  - also appends a CRC
- PGP also segments messages if too big

PGP Session Keys

- need a session key for each message
  - of varying sizes: 56-bit DES, 128-bit CAST orIDEA, 168-bit Triple-DES
- generated using ANSI X12.17 mode
- uses random inputs taken from previoususes and from keystroke timing of user

PGP Public & Private Keys

- since many public/private keys may be in use,need to identify which is actually used to encryptsession key in a message
  - could send full public-key with every message
  - but this is inefficient
- rather use a key identifier based on key
  - is least significant 64-bits of the key
  - will very likely be unique
- also use key ID in signatures

PGP Key Rings

- each PGP user has a pair of keyrings:
  - public-key ring contains all the public-keys ofother PGP users known to this user, indexedby key ID
  - private-key ring contains the public/privatekey pair(s) for this user, indexed by key ID &encrypted keyed from a hashed passphrase
- security of private keys thus depends onthe pass-phrase security

Ks = session key used in symmetric encryption scheme
PRa = private key of user A, used in public-key encryption scheme
PUa = public key of user A, used in public-key encryption scheme

EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format

PGP Key Management

- rather than relying on certificate authorities
- in PGP every user is own CA
  - can sign keys for users they know directly
- forms a "web of trust"
  - trust keys have signed
  - can trust keys others have signed if have a chain ofsignatures to them
- key ring includes trust indicators
- users can also revoke their keys

S/MIME (Secure/Multipurpose Internet Mail Extensions)

- security enhancement to MIME email
  - original Internet RFC822 email was text only
  - MIME provided support for varying contenttypes and multi-part messages
  - with encoding of binary data to textual form
  - S/MIME added security enhancements
- have S/MIME support in many mail agents
  - eg MS Outlook, Mozilla, Mac Mail etc

S/MIME Functions

- enveloped data
  - encrypted content and associated keys
- signed data
  - encoded message + signed digest
- clear-signed data
  - cleartext message + encoded signed digest
- signed & enveloped data
  - nesting of signed & encrypted entities

S/MIME CryptographicAlgorithms

- digital signatures: DSS & RSA
- hash functions: SHA-1 & MD5
- session key encryption: ElGamal& RSA
- message encryption: AES, Triple-DES, RC2/40 and others
- MAC: HMAC with SHA-1

- have process to decide which algs to use

S/MIME Messages

- S/MIME secures a MIME entity with asignature, encryption, or both
- forming a MIME wrapped PKCS object
- have a range of content-types:
  - enveloped data
  - signed data
  - clear-signed data
  - registration request
  - certificate only message

S/MIME Certificate Processing

- S/MIME uses X.509 v3 certificates
- managed using a hybrid of a strict X.509CA hierarchy & PGP's web of trust
- each client has a list of trusted CA's certs
- and own public/private key pairs & certs
- certificates must be signed by trusted CA's

Certificate Authorities

- have several well-known CA's
- Verisign one of most widely used
- Verisign issues several types of Digital IDs
- increasing levels of checks & hence trust

5.3 WEB SECURITY

- Web now widely used by business, government,individuals
- but Internet & Web are vulnerable
- have a variety of threats
- integrity
- confidentiality
- denial of service
- authentication
- need added security mechanisms

Secure Naming

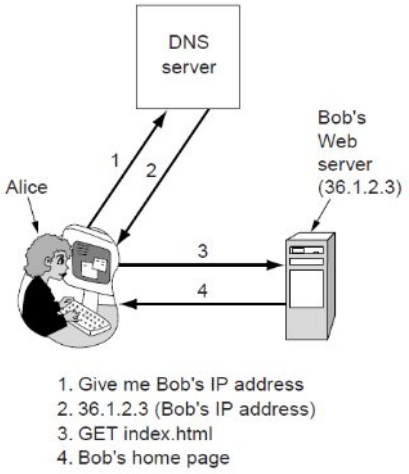


Fig 5.15 Normal situation

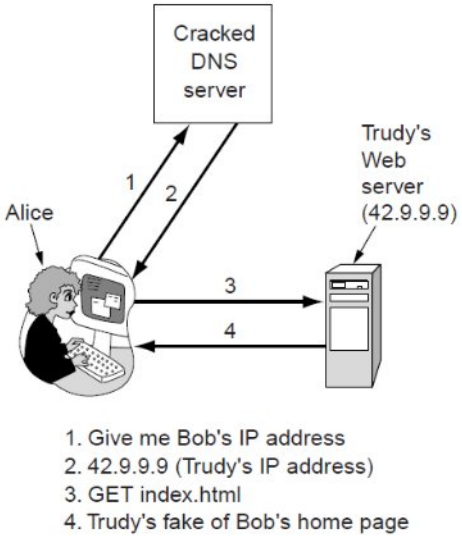


Fig 5.16 An attack based on breaking into DNS and modifying Bob's record

DNSsec fundamental services:

- Proof of where the data originated.
- Public key distribution.
- Transaction and request authentication.

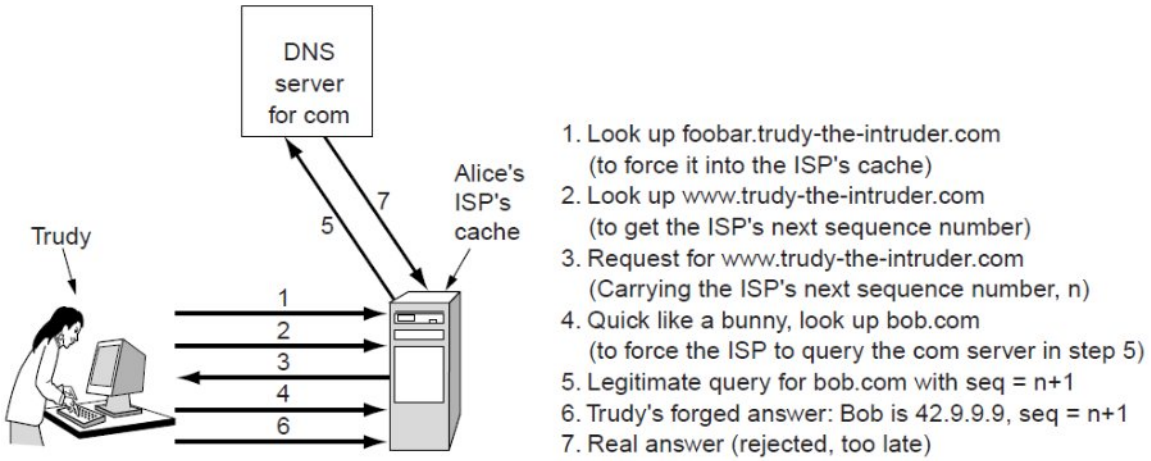


Fig 5.17 How Trudy spoofs Alice's ISP.

SSL (Secure Socket Layer)

- transport layer security service
- originally developed by Netscape
- version 3 designed with public input
- subsequently became Internet standard known asTLS (Transport Layer Security)
- uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

Application (HTTP)
Security (SSL)
Transport (TCP)
Network (IP)
Data link (PPP)
Physical (modem, ADSL, cable TV)

Fig 5.18 Layers (and protocols) for a home user browsing with SSL

Uses Public Key Scheme

- Each client-server pair uses
- 2 public keys
- one for client (browser)
  - created when browser is installed on client machine
- one for server (http server)
  - created when server is installed on server hardware
- 2 private keys
- one for client browser
- one for server (http server)

SSL Architecture

• **SSL session**
• an association between client & server
• created by the Handshake Protocol
• define a set of cryptographic parameters
• may be shared by multiple SSL connections
• **SSL connection**
• a transient, peer-to-peer, communications link
• associated with 1 SSL session

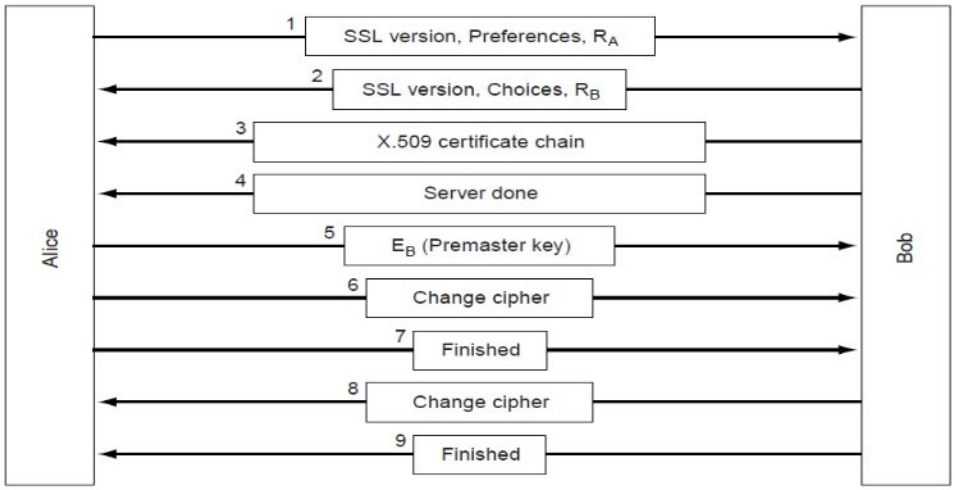


Fig 5.19 A simplified version of the SSL connection establishment subprotocol.

SSL Record Protocol

• **confidentiality**
• using symmetric encryption with a shared secret keydefined by Handshake Protocol
• IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
• message is compressed before encryption
• **message integrity**
• using a MAC (Message Authentication Code) createdusing a shared secret key and a short message

SSL Change Cipher Spec Protocol

• one of 3 SSL specific protocols which usethe SSL Record protocol
• a single message
• causes pending state to become current
• hence updating the cipher suite in use

SSL Alert Protocol

• conveys SSL-related alerts to peer entity
• severity
• warning or fatal

• specific alert
• unexpected message, bad record mac, decompression failure,handshake failure, illegal parameter
• close notify, no certificate, bad certificate, unsupportedcertificate, certificate revoked, certificate expired, certificateunknown
• compressed & encrypted like all SSL data

SSL Handshake Protocol

• allows server & client to:
• authenticate each other
• to negotiate encryption & MAC algorithms
• to negotiate cryptographic keys to be used
• comprises a series of messages in phases
• Establish Security Capabilities
• Server Authentication and Key Exchange
• Client Authentication and Key Exchange
• Finish

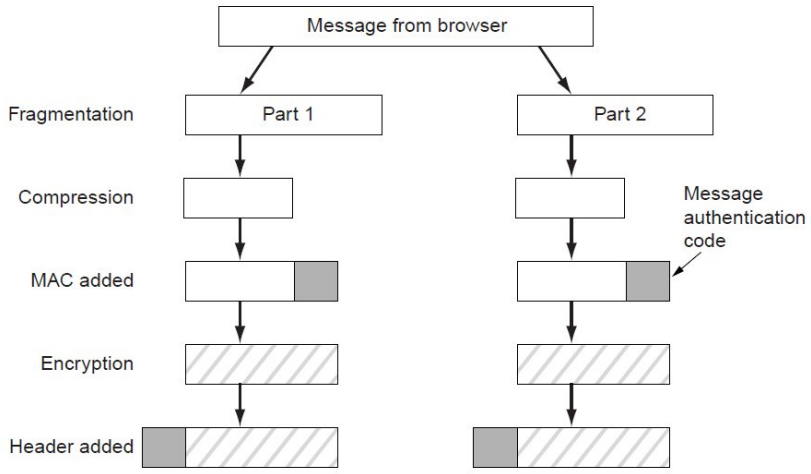


Fig 5.20 Data transmission using SSL

SSL Handshake Protocol

TLS (Transport Layer Security)

• IETF standard RFC 2246 similar to SSLv3
• with minor differences
• in record format version number
• uses HMAC for MAC
• a pseudo-random function expands secrets
• has additional alert codes
• some changes in supported ciphers
• changes in certificate negotiations
• changes in use of padding

Secure Electronic Transactions (SET)

• open encryption & security specification

• to protect Internet credit card transactions
• developed in 1996 by Mastercard, Visa etc
• not a payment system, rather a set of security protocols & formats
• secure communications amongst parties
• trust from use of X.509v3 certificates
• privacy by restricted info to those who need it

SET Transaction

1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. merchant requests payment

Dual Signature

• customer creates dual messages
• order information (OI) for merchant
• payment information (PI) for bank
• neither party needs details of other
• but **must** know they are linked
• use a dual signature for this
• signed concatenated hashes of OI & PI

Purchase Request – Merchant

1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's publicsignature key to ensure order has not beentampered with in transit & that it was signedusing cardholder's private signature key
3. processes order and forwards the paymentinformation to the payment gateway forauthorization (described later)
4. sends a purchase response to cardholder

Payment Gateway Authorization

1. verifies all certificates
2. decrypts digital envelope of authorization block to obtainsymmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtainsymmetric key & then decrypts payment block
5. verifies dual signature on payment block
6. verifies that transaction ID received from merchantmatches that in PI received (indirectly) from customer
7. requests& receives an authorization from issuer

8. sends authorization response back to merchant

Payment Capture

• merchant sends payment gateway apayment capture request
• gateway checks request
• then causes funds to be transferred tomerchants account
• notifies merchant using capture response

5.4 SOCIAL ISSUES

Privacy

Nowadays, telephone companies and Internet providers readily provide wiretaps when presented with search warrants. It makes life much easier for the policeman and there is no danger of falling off the horse. Cryptography changes all that. Anybody who goes to the trouble of downloading and installing PGP and who uses a well-guarded alien-strength key can be fairly sure that nobody in the known universe can read his e-mail, search warrant or no search warrant. Governments well understand this and do not like it. Real privacy means it is much harder for them to spy on criminals of all stripes, but it is also much harder to spy on journalists and political opponents.

Consequently, some governments restrict or forbid the use or export of cryptography. In France, for example, prior to 1999, all cryptography was banned unless the government was given the keys. France was not alone. In April 1993, the U.S. Government announced its intention to make a hardware crypto processor, the clipper chip, the standard for all networked communication. In this way, it was said; citizens' privacy would be guaranteed. It also mentioned that the chip provided the government with the ability to decrypt all traffic via a scheme called key escrow, which allowed the government access to all the keys. However, it promised only to snoop when it had a valid search warrant. Needless to say, a huge furor ensued, with privacy advocates denouncing the whole plan and law enforcement officials praising it. Eventually, the government backed down and dropped the idea.

Anonymous Remailers

PGP, SSL, and other technologies make it possible for two parties to establish secure, authenticated communication, free from third-party surveillance and interference. However, sometimes privacy is best served by not having authentication, in fact by making communication anonymous. The anonymity may be desired for point-to-point messages, newsgroups, or both.

Let us consider some examples. First, political dissidents living under authoritarian regimes often wish to communicate anonymously to escape being jailed or killed. Second, wrongdoing in many corporate, educational, governmental, and other organizations has often been exposed by whistleblowers, who frequently prefer to remain anonymously to avoid retribution. Third, people with unpopular social, political, or religious views may wish to communicate with each other via e-mail or newsgroups without exposing themselves. Fourth, people may wish to discuss alcoholism, mental illness, sexual harassment, child abuse, or being a member of a persecuted minority in a newsgroup without having to go public. Numerous other examples exist, of course.

Let us consider a specific example. In the 1990s, some critics of a nontraditional religious group posted their views to a USENET newsgroup via an anonymous remailer. This server

allowed users to create pseudonyms and send e-mail to the server, which then re-mailed or reposted them using the pseudonym, so no one could tell where the message really came from. Some postings revealed what the religious group claimed were trade secrets and copyrighted documents. The religious group responded by telling local authorities that its trade secrets had been disclosed and its copyright infringed, both of which were crimes where the server was located. A court case followed and the server operator was compelled to turn over the mapping information which revealed the true identities of the persons who had made the postings. (Incidentally, this was not the first time that a religion was unhappy when someone leaked its secrets: William Tyndale was burned at the stake in 1536 for translating the Bible into English).

A substantial segment of the Internet community was outraged by this breach of confidentiality. The conclusion that everyone drew is that an anonymous remailer that stores a mapping between real e-mail addresses and pseudonyms (called a type 1 remailer) is not worth much. This case stimulated various people into designing anonymous remailers that could withstand subpoena attacks.

These new remailers often called cypherpunk remailers, work as follows. The user produces an e-mail message, complete with RFC 822 headers (except From:, of course), encrypts it with the remailer's public key, and sends it to the remailer. There the outer RFC 822 headers are stripped off, the content is decrypted and the message is remailed. The remailer has no accounts and maintains no logs, so even if the server is later confiscated, it retains no trace of messages that have passed through it.

Many users who wish anonymity chain their requests through multiple anonymous remailers, as shown in Fig.5.13. Here, Alice wants to send Bob a really, really, really anonymous Valentine's Day card, so she uses three remailers. She composes the message, M, and puts a header on it containing Bob's e-mail address. Then she encrypts the whole thing with remailer 3's public key, E3 .(Indicated by horizontal hatching). To this she prepends a header with remailer 3's e-mail address in plaintext. This is the message shown between remailers 2 and 3 in the Fig

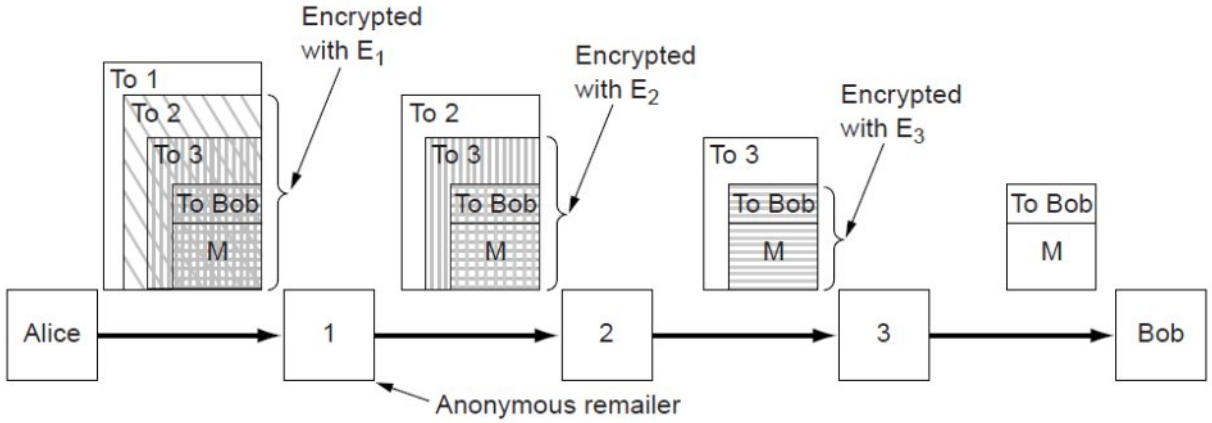


Fig 5.21 How Alice uses 3 remailers to send Bob a message

Then she encrypts this message with remailer 2's public key, E2 (indicated by vertical hatching) and prepends a plaintext header containing remailer 2's e-mail address. This message is shown between 1 and 2 in Fig.5.4.1. Finally, she encrypts the entire message with remailer 1's public key, E1 , and prepends a plaintext header with remailer 1's e-mail address. This is the message shown to the right of Alice in the Fig. and this is the message she actually transmits.

When the message hits remailer 1, the outer header is stripped off. The body is decrypted and then e-mailed to remailer 2. Similar steps occur at the other two remailers.

Although it is extremely difficult for anyone to trace the final message back to Alice, many remailers take additional safety precautions. For example, they may hold messages for a random time, add or remove junk at the end of a message, and reorder messages, all to make it harder for anyone to tell which message output by a remailer corresponds to which input, in order to thwart traffic analysis. For a description of a system that represents the state of the art in anonymous e-mail, see (Mazières and Kaashoek, 1998).

Anonymity is not restricted to e-mail. Services also exist that allow anonymous Web surfing. The user conFig.s his browser to use the anonymizer as a proxy. Henceforth, all HTTP requests go to the anonymizer, which requests the page and sends it back. The Web site sees the anonymizer as the source of the request, not the user. As long as the anonymizer refrains from keeping a log, after the fact no one can determine who requested which page.

Steganography

In countries where censorship abounds, dissidents often try to use technology to evade it. Cryptography allows secret messages to be sent (although possibly not lawfully), but if the government thinks that Alice is a Bad Person, the mere fact that she is communicating with Bob may get him put in this category, too, as repressive governments understand the concept of transitive closure, even if they are short on mathematicians. Anonymous remailers can help, but if they are banned domestically and messages to foreign ones require a government export license, they cannot help much. But the Web can.

People who want to communicate secretly often try to hide the fact that any communication at all is taking place. The science of hiding messages is called steganography, from the Greek words for "covered writing." In fact, the ancient Greeks used it themselves. Herodotus wrote of a general who shaved the head of a messenger, tattooed a message to his scalp, and let the hair grow back before sending him off. Modern techniques are conceptually the same, only they have a higher bandwidth and lower latency.

5.5 SDL BASED PROTOCOL VERIFICATION AND VALIDATION

Objectives of Protocol Verification/Validation

- Availability of Reliable and correct communication protocols
- Important Issues of communication protocols
  - Correctness: the warranty of showing of the intended behavior in any specific situation.
  - Robustness: the property of being able to work correctly under abnormal conditions
  - Performance: Utilization of the available bandwidth it is able to achieve over the physical medium.
- In performing verification, validation and testing, the Formal techniques:
  - reduce complexity
  - eliminate ambiguity
  - Prepare structured protocol.

Protocol Verification

Concentrates on verifying the following properties of a given protocol specification.

- liveness property: “Good things will happen” , they include termination requirements in sequential protocol sand recurrent properties in no terminating protocols like network operating system protocols.
- safety property: “Bad things will not happen” , they are non violation of assertion sand invariants.
- Correctness property: the warranty of showing of the intended behavior in any specific situation

Verification of Protocol

- Normal operation of the protocol:
- Proof of safety properties:
  - handling of lost frames
  - handling loss of an acknowledgement
  - no two duplicates are delivered to the receiver
- Proof liveness properties:
  - From the transition table we observe that themessages0and1are transmitted to the receiver even under the conditions of frame and acknowledgement loss and the protocol returns toits terminator state (000).
  - The terminator state is a state of a system whose occurrence means all the specified messages have been transmitted and received correctly.

Safety properties of ABP

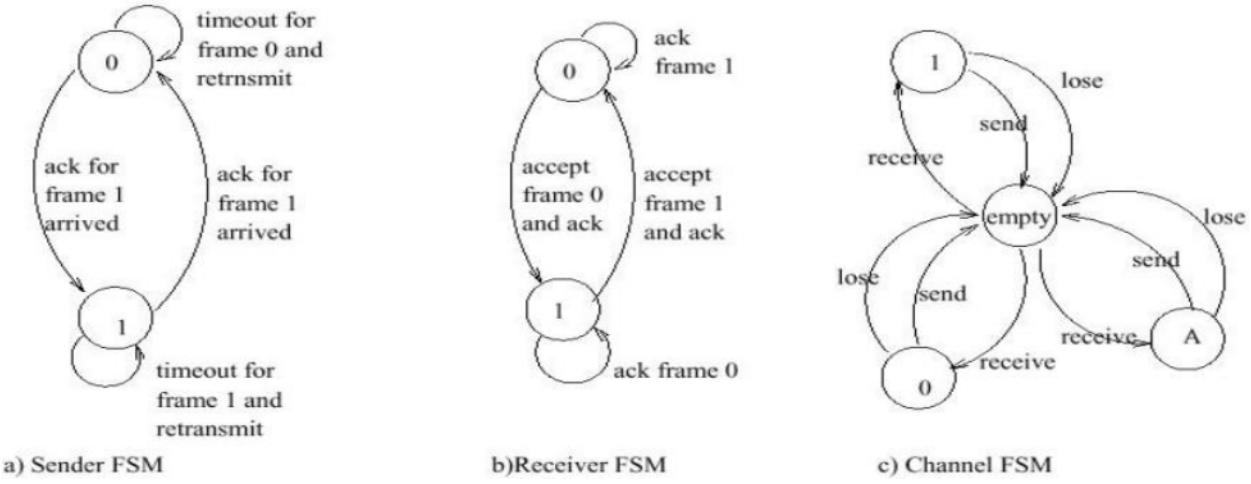
Safety properties of ABP (Alternatingbit protocol) are:

- The sender ensures that the data with the correct sequence number is sent to the receiver even though, the data is lost in the channel
- the receiver ensures that an ack is sent to the sender even if the sent ack is lost in the channel, and
- the receiver never delivers two odd packets (the packets with sequence numbers 1and1) without an intervening even packet (packet with sequence number 0)

FSM in Protocol Verification

Liveness properties

● the protocol terminates correctly



Protocol Validation

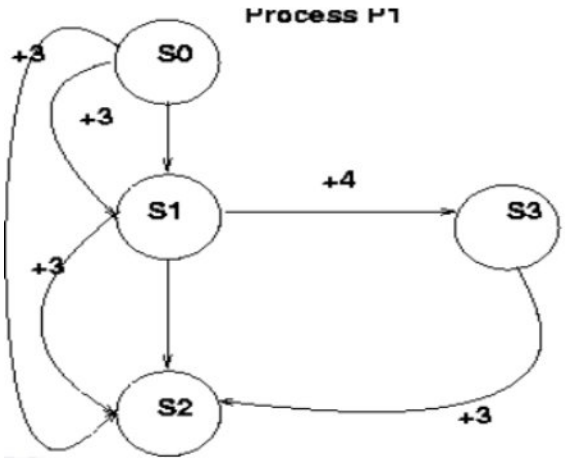
- Protocol validation is a method of checking whether the interactions of protocol entities are according to the protocol specification,
- Do indeed satisfy certain properties or conditions which may be either general or specific to the particular protocol system directly derived from the specifications.
- Validation sometimes refers to check the protocol specification such that it will not get into protocol design errors like deadlock, unspecified receptions, and livelock errors

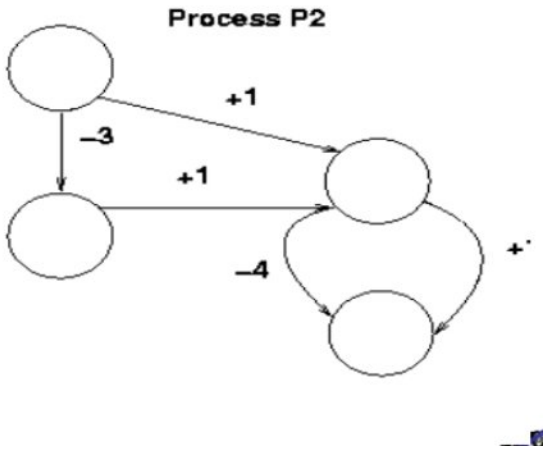
Protocol Design Errors

- State deadlocks
- Unspecified receptions
- Non-executable interactions
- Non-executable interactions
- State ambiguity of a protocol
- Unboundedness of a protocol
- Lack of adaptation in a protocol
- Livelocks

StateDeadlocks

- A state deadlock occurs when every process of a protocol has no alternative but to remain indefinitely in the same state.
- In other words, a state dead lock will prevail when no transmissions are possible from the current state of each process of a protocol, and when no messages are in transit, i.e., all channels are empty





Unspecified Reception

- An un specified reception occurs when a positive arc of the FSM of the process of a protocol that can be traverse dismissing, in other words, when a reception that can take place is not specified in the design.
- sequence of transitions from the diagram:
  - Process P1inS0(state0) sendsamessage1toprocessP2andmoves toS1 (state1).
  - P2 initiates the transition by sendingamessage3toP1andmoves fromS0toS1 state.
  - The channel P12(channel fromP1toP2) and the channel P21(channel fromP2 toP1)contains themessages1and3respectively.
  - Now P1 receives themessage3fromchannel P21andmoves fromS1toS2.
  - **P2** receives the message1 from channel P12 and moves fromS1toS2.

NonexecutableInteractions

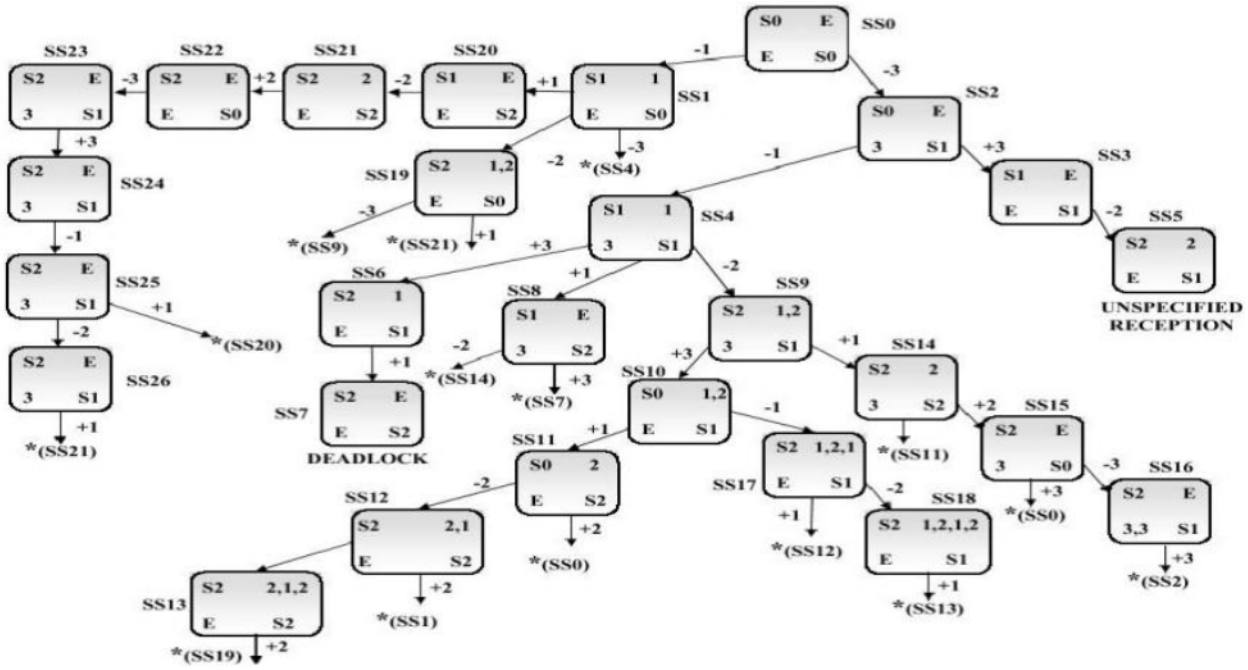
- When a design includes message transmissions and receptions that cannot occur under the normal operating conditions.
- A nonexecutable interaction is equivalent to dead-code in a computer.
- Example:
  - transition sequences of the processes as given in the diagram:
    - P1 in state S0s ends message 1 to P2 and moves to S1.
    - P2 receives the message 1 and moves to state S2.
    - P1 sends a message 2 and moves from S1 to S2.

StateAmbiguities

- A stable state pair (X,Y) is said to when a state ax of a process(say P1) and a state Yinanother process (sayP2)can be reached with both channels empty
- Monitoring stable-state pairs is useful for detecting the loss of synchronization
- A state ambiguity exists when a state of one process can coexists stably with several different states in the other process.
- consider the following execution of transitions for system given in the figure
    - P1 instate S0sends amessage1andmoves tostateS1.
    - P1whichis instateS1sendsamessage2andmoves tostateS2.
    - P2instateS0receives message1andmoves toS2.
    - P2whichis instateS2receives message2andmoves toS0

Protocol Validation Approaches

Perturbation Technique: Reachability Analysis



Reachability Analysis

Consider global state SS4 in which:

- P1is inS1state
- P2is inS1state
- Channel P12is containingmessage1, and
- Channel P21is containingmessage3.

Fromthis thefollowingtransitions arepossible:

- P1can receive the message 3 and transit to globals stateS6,
- P2 can receive the message 1and transit to SS8,
- P1can transmit a message 2 and reach SS9.

Pros&Consof ReachabilityAnalysis

The advantages of using the reachability tree(sometimes called as graphs) for protocol validation are:

- The tree generation can be easily automated, and
- Many logical errors can be detected by only examining individual global states in the reachability graph.

The disadvantages of using reachability graphs are:

- State space explosion problem;
- Does not work on unbounded protocols; and
- Many relationships among the protocol state variables, expressing the desirable logical correctness properties of the protocol, are not apparent from simply traversing the reachability tree

Fair Reachability Graphs

- Consider simple message exchange protocol where Process P1(sender) sendes 3 messages to process P2 (receiver) via a queue that has a maximum capacity of 3 messages.
- The possible sequences are shown in the figure, have the common property that both process execute the same sequence of interaction steps, that is P1 sends 3 messages and P2 receives 3 messages.
- To illustrate sequence, consider sequence 5 and 1.Sequence 5 can only be executed if at least 3 messages are allowed in the queue. Similarly sequence 1requires only single queued message (transitionS12), ie reception is followed by every transmission (transitions S12, R21, S12, R21, S12, R21).
- The sequence is treated as a series of executing steps, where each sequence consists of a protocol process sending or receiving message.
- Assumed that any transition starts and ends in a stable global system states.
- This restriction requires that protocol execution periodically results in a global statein which all transmitted messages have been received.

Sequence Number	Transitions					
1	S12	R21	S12	R21	S12	R21
2	S12	R21	R21	R21	S12	R21
3	S12	R21	S12	R21	R21	R21
4	S12	S12	R21	S12	R21	R21
5	S12	S12	S12	R21	R21	R21

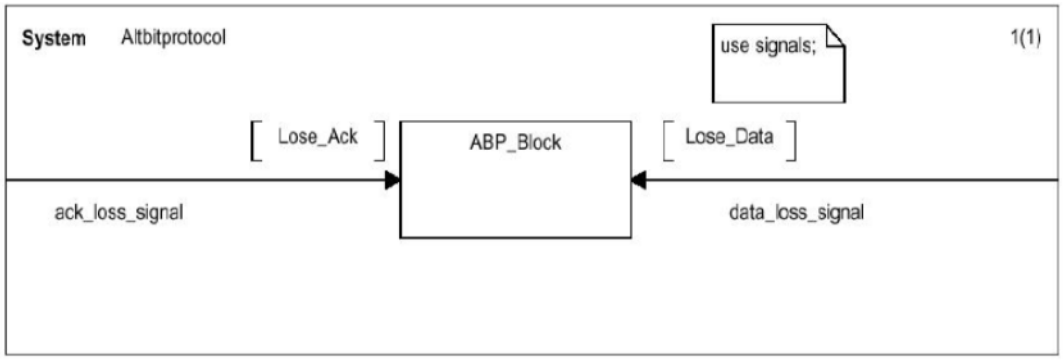
SDLBasedProtocol Verification

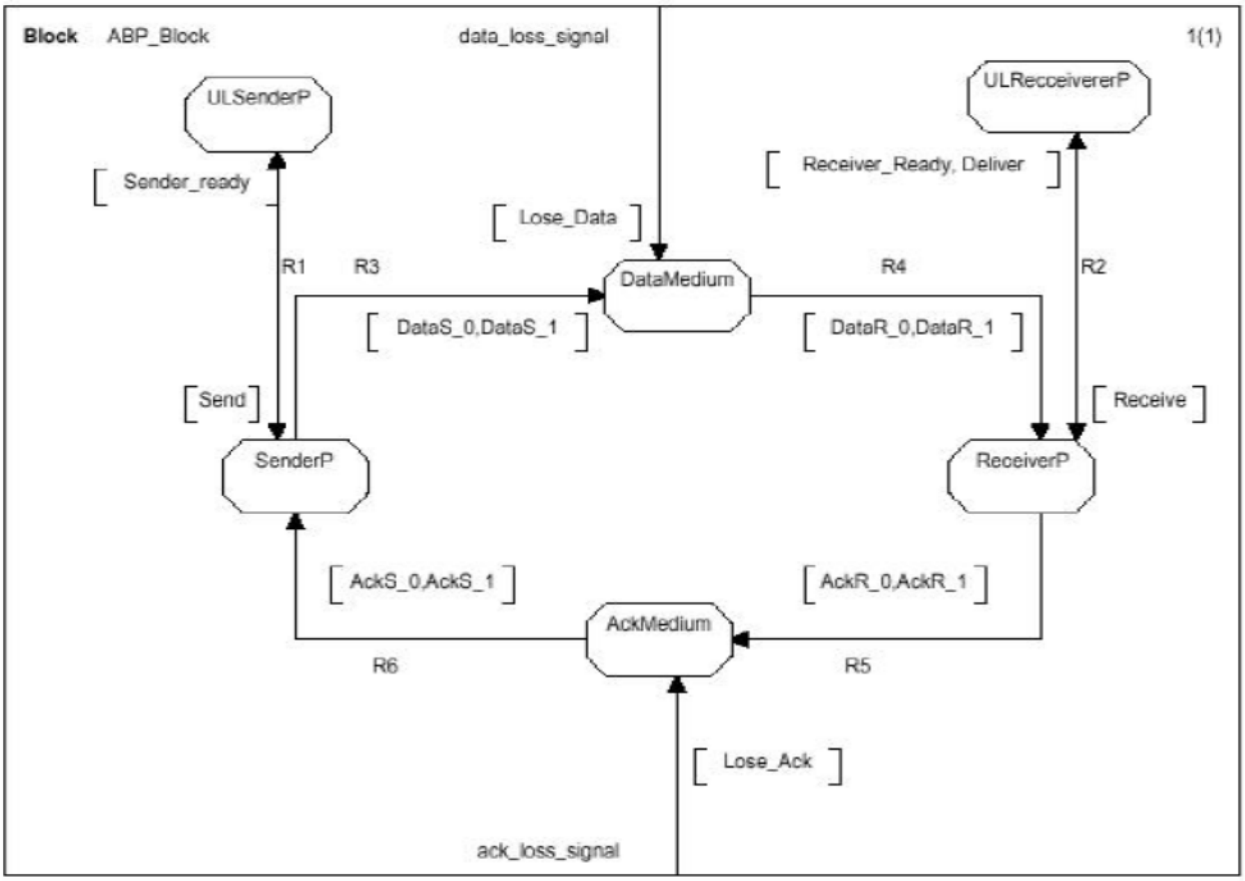
- Create the FSMs of all the entities of the protocol.
- Translate the FSMs into the SDL based specification (like system, block and process diagrams).
- Run the simulation and check that all the processes are running in the given specifications.
- Input the required signals to check for safety and liveness properties of the specified protocol.
- Generate the MSC diagrams using the SDL tool.
- Check the MSC diagrams of the test-cases to verify the safety and liveness properties.

AlternatingBit Protocol

Theblock consists of thefollowingprocesses

- UL sender P: upper layer sender process
- UL Receiver: upper layer receiver process
- Data Medium: provides service to transmit data from sender to receiver
- Sender P: sender process to transmit the frames
- Receiver P: receiver process to receive the data and deliver to upper layer process as well as acknowledge the received data.
- Ack Medium: provides service to transmit the acknowledgments received from the receiver to sender.





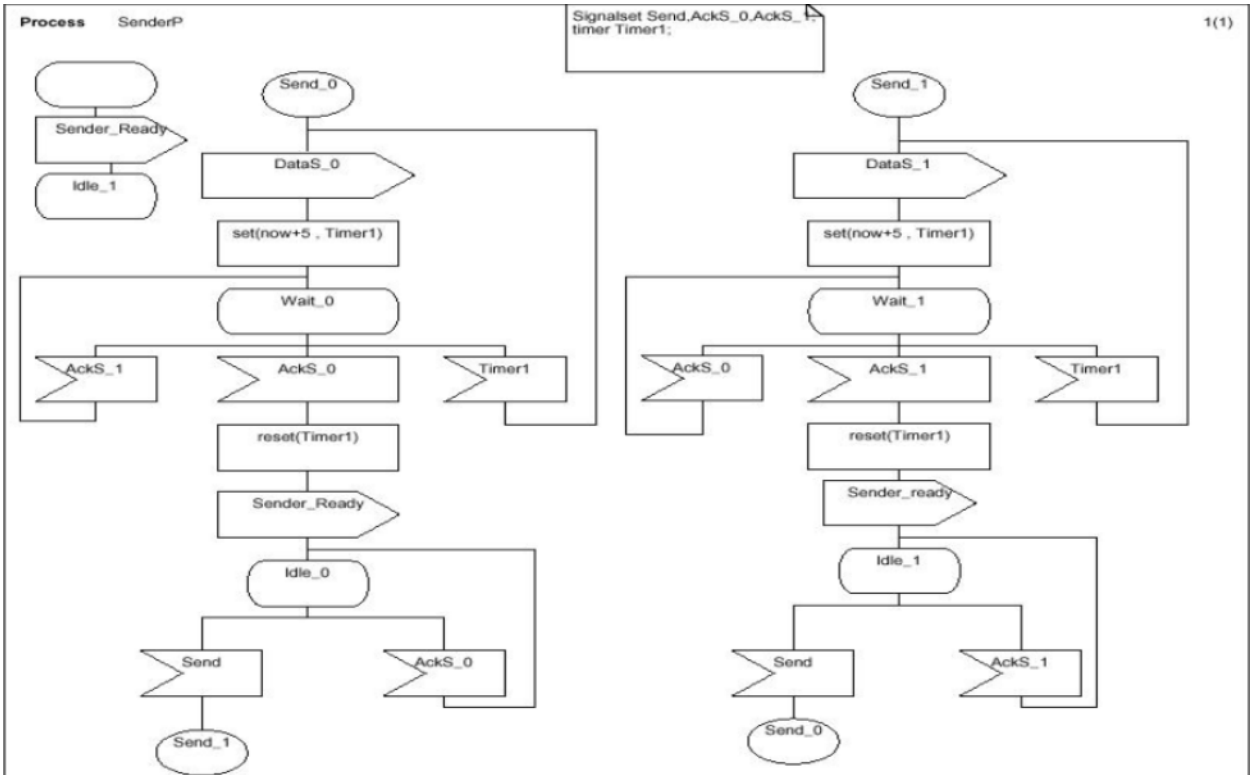
Alternating Bit Protocol

The block consists of the following processes

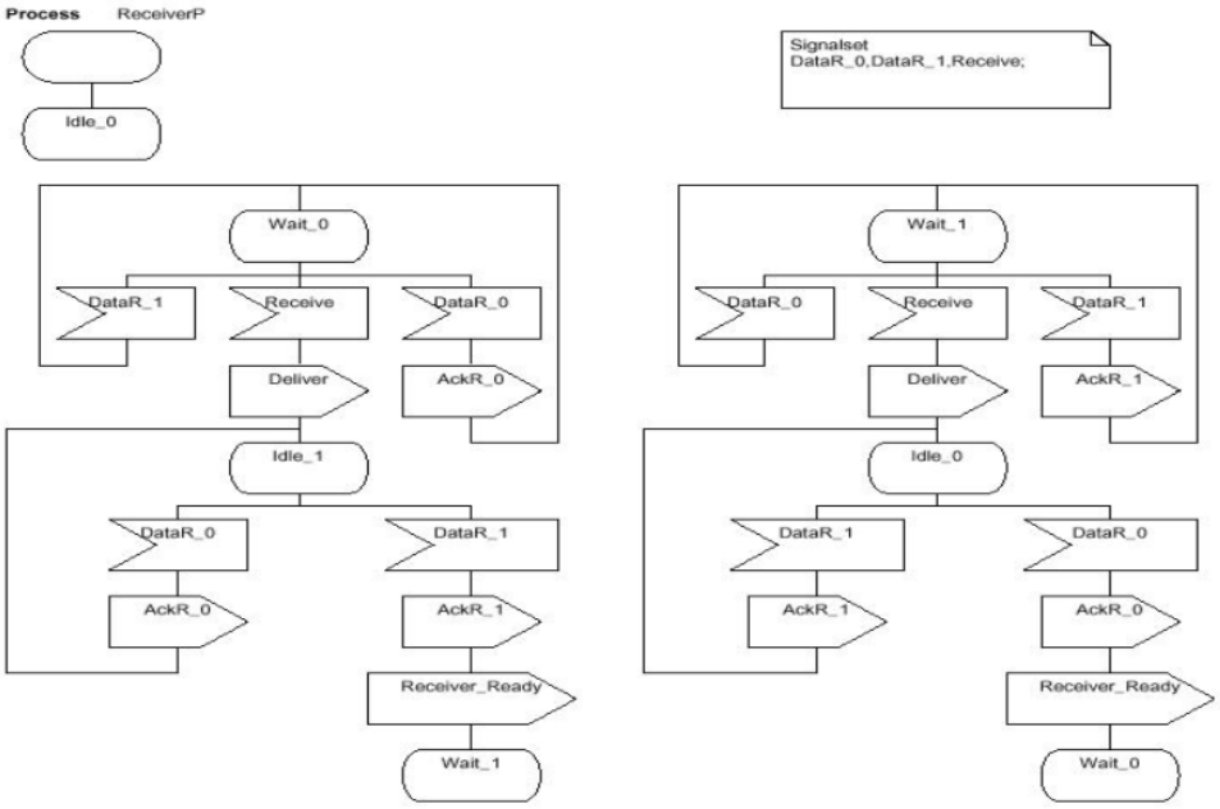
- ULsenderP: upper layer sender process
- ULReceiver: upper layer receiver process
- DataMedium: provides service to transmit data from sender to receiver

- SenderP: sender process to transmit the frames
- Receiver P: receiver process to receive the data and deliver to upper layer process as well as acknowledge the received data.
- Ack Medium: provides service to transmit the acknowledgments received from the receiver to sender

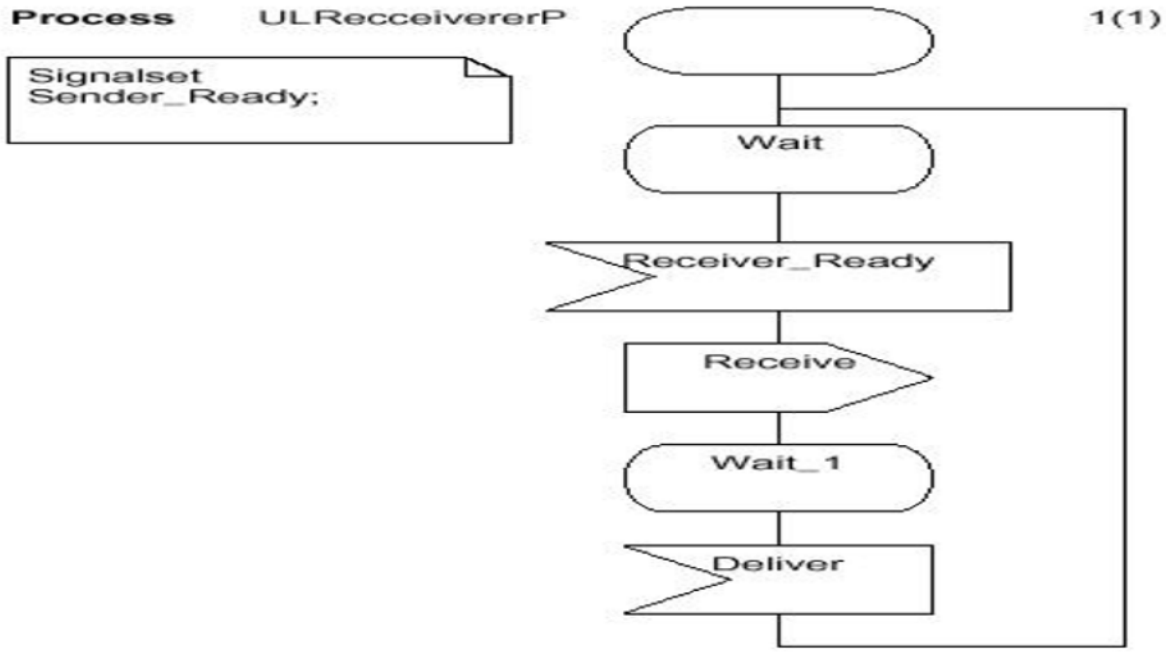
Sender Process of ABP



Receiver Process of ABP

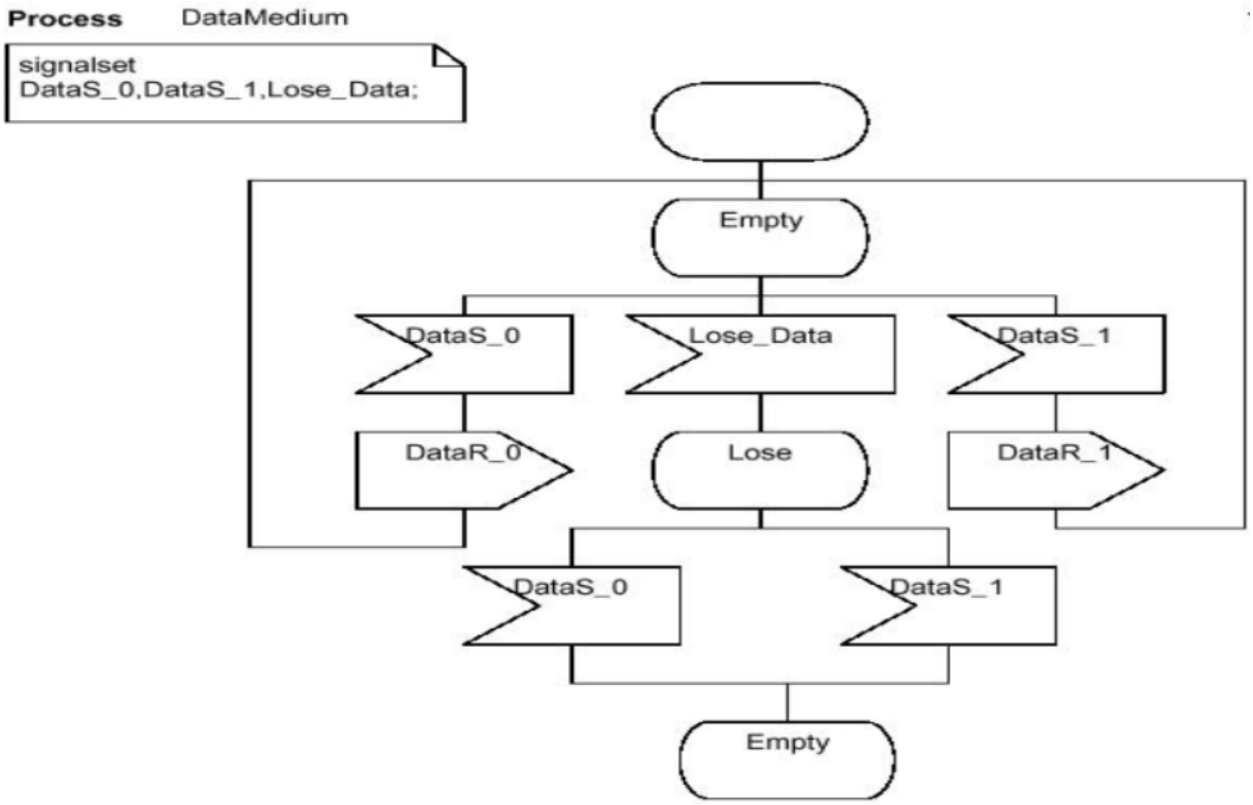


Upper Layer of receiver of ABP

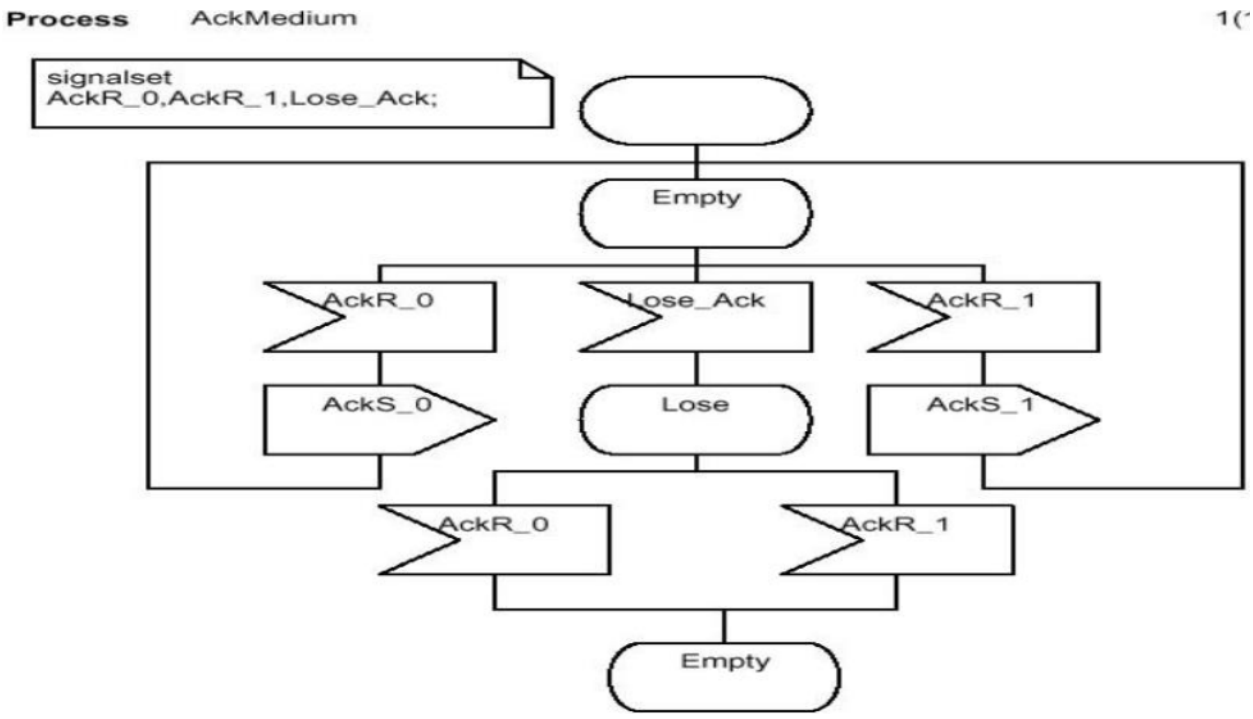


Data Medium Process

[Type text]



Ack Medium Process



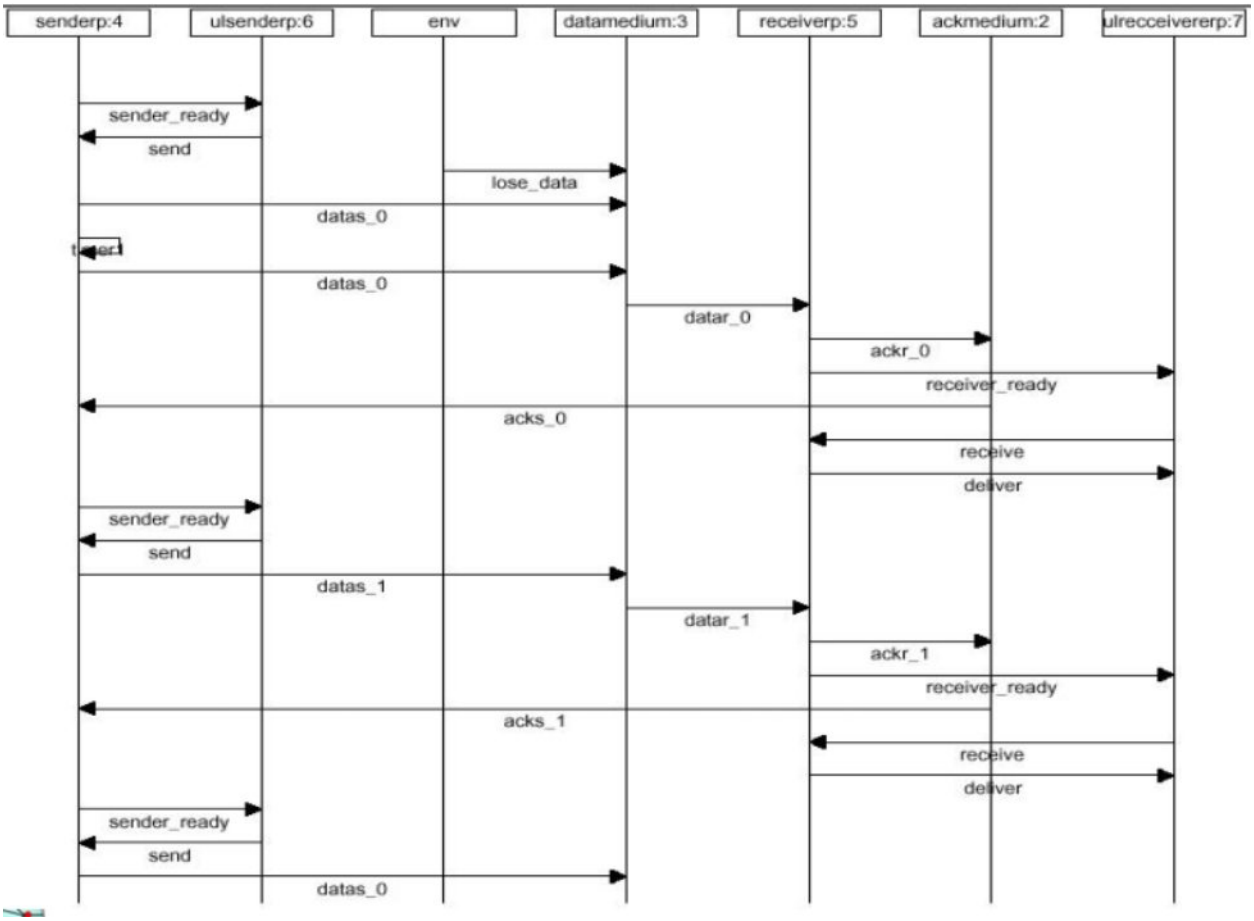
Verification of the ABP

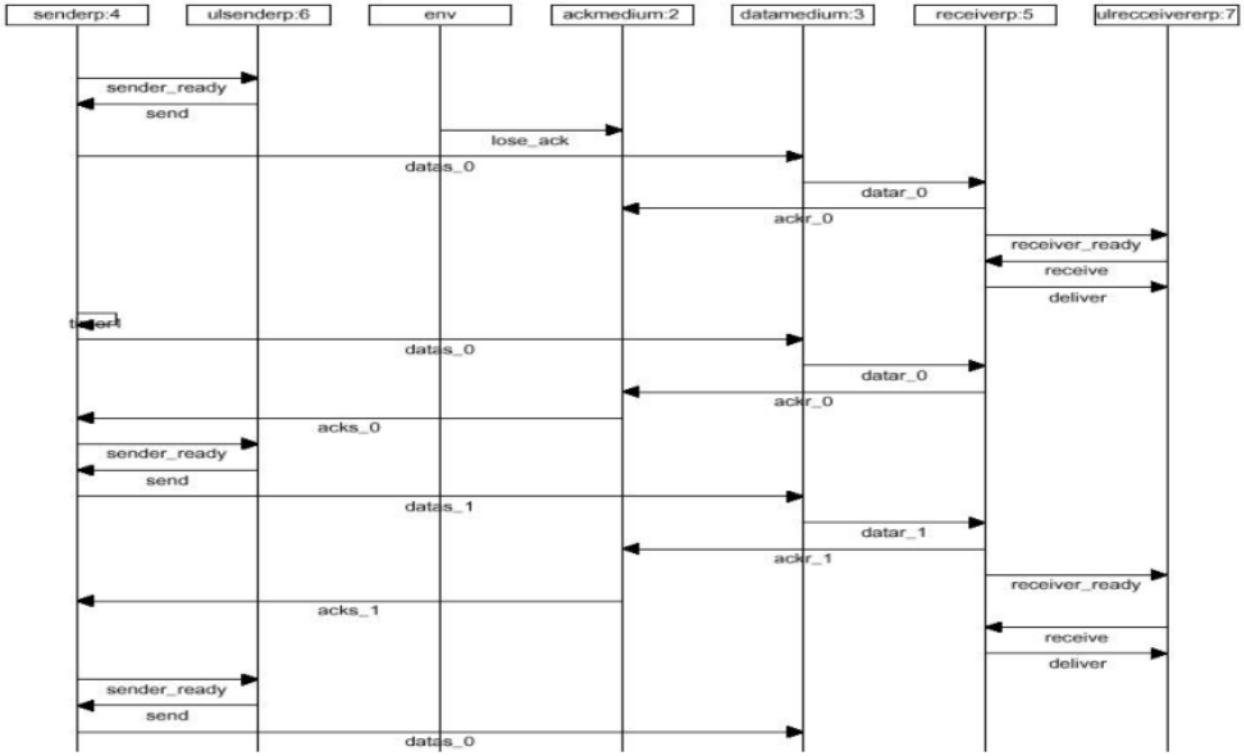
Safety properties:

To verify the ABP sends the packet with correct sequence number to the receiver, even if the medium loses the data frames. We send lose data signal to the data medium. When sender process sends data_0 packet, the data medium does not forwards it to the receiver but sender times out ultimately and retransmits the packet and reaches the destination.

To verify that ABP sends the ack signal with correct sequence number to sender, even if the medium loses the ack, we send lose_ack signal to ack_medium. Sender sends data_0, receiver send ack_0 to ack medium which loses ack making sender to timeout and resend data_0, receiver retxm. ack_0 which reaches destination.

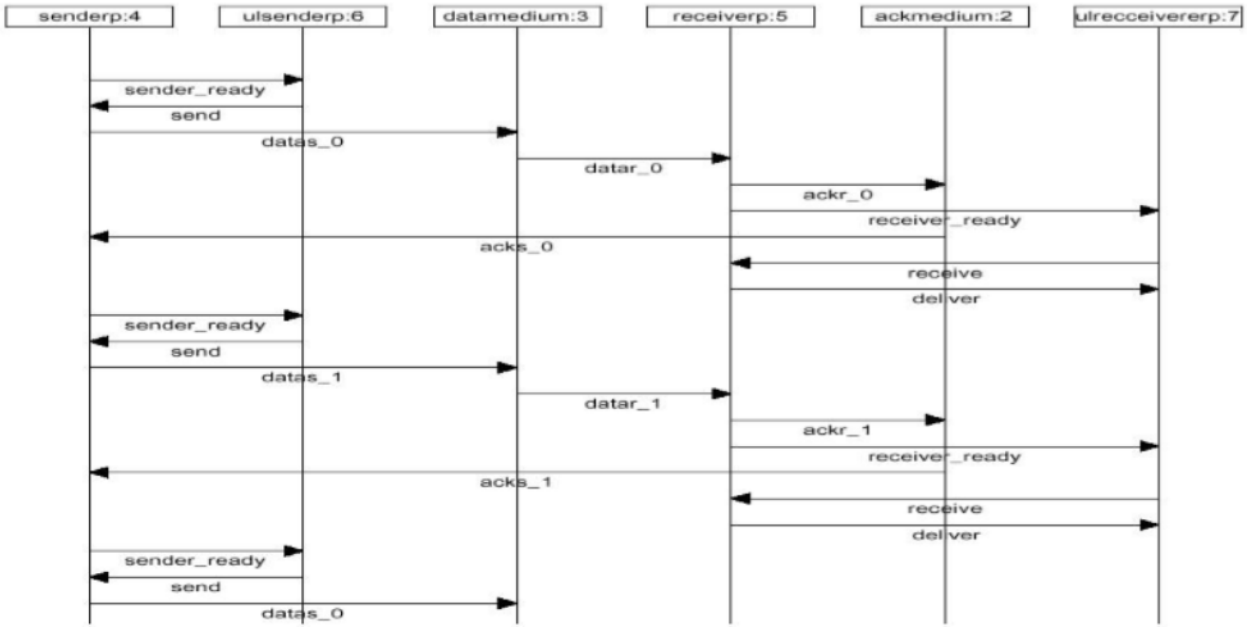
Verification of the ABP – Safety Property





Liveness property

To check that protocol terminates properly and goes into to the initial state for next operation



SDL based protocol validation

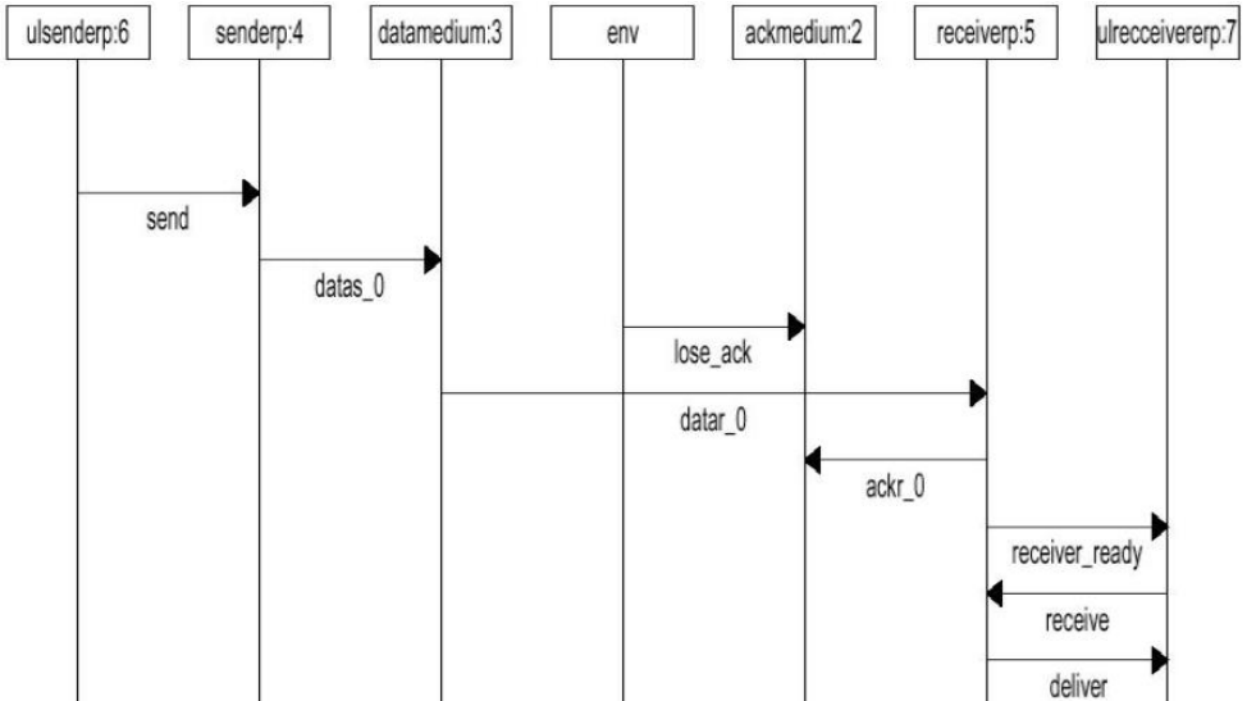
[Type text]

- Create the FSMs of all the entities of the protocol.
- Translate the FSMs into the SDL based specification.
- Run the simulation and check that all the processes are running.
- Input the required signals to the processes to check for the design errors.
- Generate the MSC diagrams using the SDL tool.
- Check the MSC diagrams of the test-cases to validate the behavior.

Validation of ABP for its deadlock design error

- Looking at MSC, ABP doesnot go into deadlock even if the medium loses data or ack .
- The protocol may have gone into dead lock if we remove retransmission after time out.
- If data or ack is lost, sender will never receive the ack signal and will be in waiting state. Also receiver remains in waiting state for next packet and protocol goes into deadlock

Validation of ABP



5.6 INTERNET PROTOCOL

- The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries.

- Its routing function enables internetworking, and essentially establishes the Internet.
- IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packetheaders.
- For this purpose, IP defines packet structures that encapsulate the data to be delivered.
- It also defines addressing methods that are used to label the datagram with source and destination information.
- Historically, IP was the connectionless datagram service in the original Transmission Control Program introduced by Vint Cerf andBob Kahn in 1974; the other being the connection-oriented Transmission Control Protocol (TCP).
- The Internet protocol suite is therefore often referred to as TCP/IP.
- The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor is Internet Protocol Version 6 (IPv6).

Functions

The Internet Protocol is responsible for addressing hosts and for routing datagrams (packets) from a source host to a destination host across one or more IP networks. For this purpose, the Internet Protocol defines the format of packets and provides an addressing system that has two functions: Identifying hosts and providing a logical location service.

Datagram construction

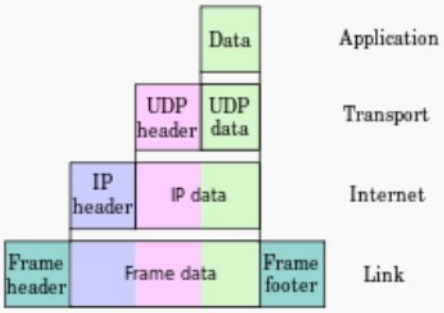


Fig 5.22 Sample encapsulation of application data from UDP to a Link protocol frame

Each datagram has two components: a header and a payload. The IP header is tagged with the source IP address, the destination IP address, and other meta-data needed to route and deliver the datagram. The payload is the data that is transported. This method of nesting the data payload in a packet with a header is called encapsulation.

IP addressing and routing

IP addressing entails the assignment of IP addresses and associated parameters to host interfaces. The address space is divided into networks and subnetworks, involving the designation of network or routing prefixes. IP routing is performed by all hosts, as well as routers, whose main function is to transport packets across network boundaries. Routers communicate with one another via specially designed routing protocols, either interior gateway protocols or exterior gateway protocols, as needed for the topology of the network.

IP routing is also common in local networks. For example, many Ethernet switches support IP multicast operations.These switches use IP addresses and Internet Group Management Protocol to control multicast routing but use MAC addresses for the actual routing.

Reliability

The design of the Internet protocols is based on the end-to-end principle. The network infrastructure is considered inherently unreliable at any single network element or transmission medium and assumes that it is dynamic in terms of availability of links and nodes. No central monitoring or performance measurement facility exists that tracks or maintains the state of the network. For the benefit of reducing network complexity, the intelligence in the network is purposely located in the end nodes of data transmission.Routers in the transmission path forward packets to the next known, directly reachable gateway matching the routing prefix for the destination address.

As a consequence of this design, the Internet Protocol only provides best effort delivery and its service is characterized as unreliable. In network architectural language, it is aconnectionless protocol, in contrast to connection-oriented modes of transmission. Various error conditions may occur, such as data corruption, packet loss, duplication and out-of-order delivery. Because routing is dynamic, meaning every packet is treated independently, and because the network maintains no state based on the path of prior packets, different packets may be routed to the same destination via different paths, resulting in out-of-order sequencing at the receiver.

Internet Protocol Version 4 (IPv4) provides safeguards to ensure that the IP packet header is error-free. A routing node calculates a checksum for a packet. If the checksum is bad, the routing node discards the packet. The routing node does not have to notify either end node, although the Internet Control Message Protocol (ICMP) allows such notification. By contrast, in order to increase performance, and since current link layer technology is assumed to provide sufficient error detection,[2] the IPv6 header has nochecksum to protect it.

All error conditions in the network must be detected and compensated by the end nodes of a transmission. The upper layer protocols of the Internet protocol suite are responsible for

resolving reliability issues. For example, a host may cache network data to ensure correct ordering before the data is delivered to an application.

Link capacity and capability

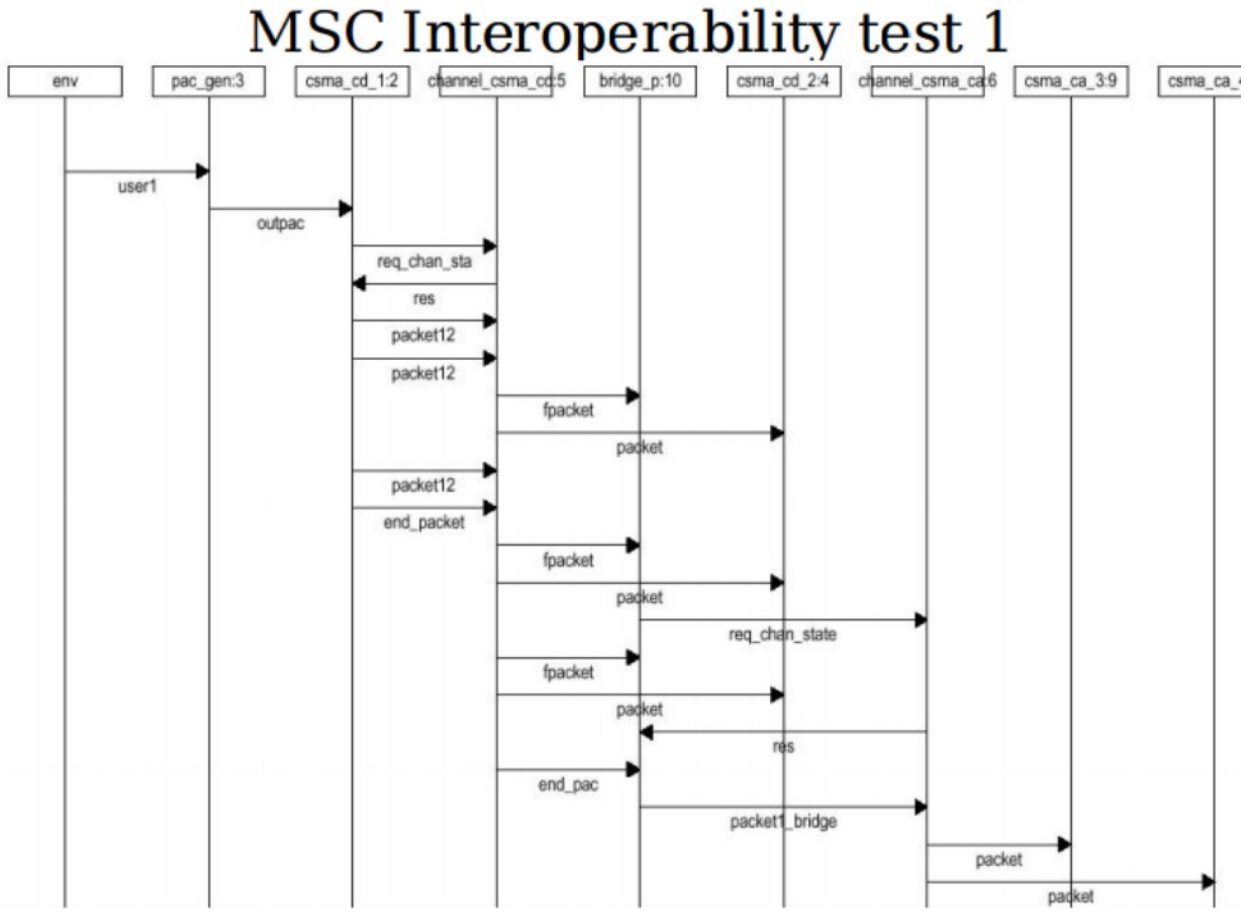
The dynamic nature of the Internet and the diversity of its components provide no guarantee that any particular path is actually capable of, or suitable for, performing the data transmission requested, even if the path is available and reliable. One of the technical constraints is the size of data packets allowed on a given link. An application must assure that it uses proper transmission characteristics. Some of this responsibility lies also in the upper layer protocols. Facilities exist to examine the maximum transmission unit (MTU) size of the local link and Path MTU Discovery can be used for the entire projected path to the destination. The IPv4 internetworking layer has the capability to automaticallyfragment the original datagram into smaller units for transmission. In this case, IP provides re-ordering of fragments delivered out of order.

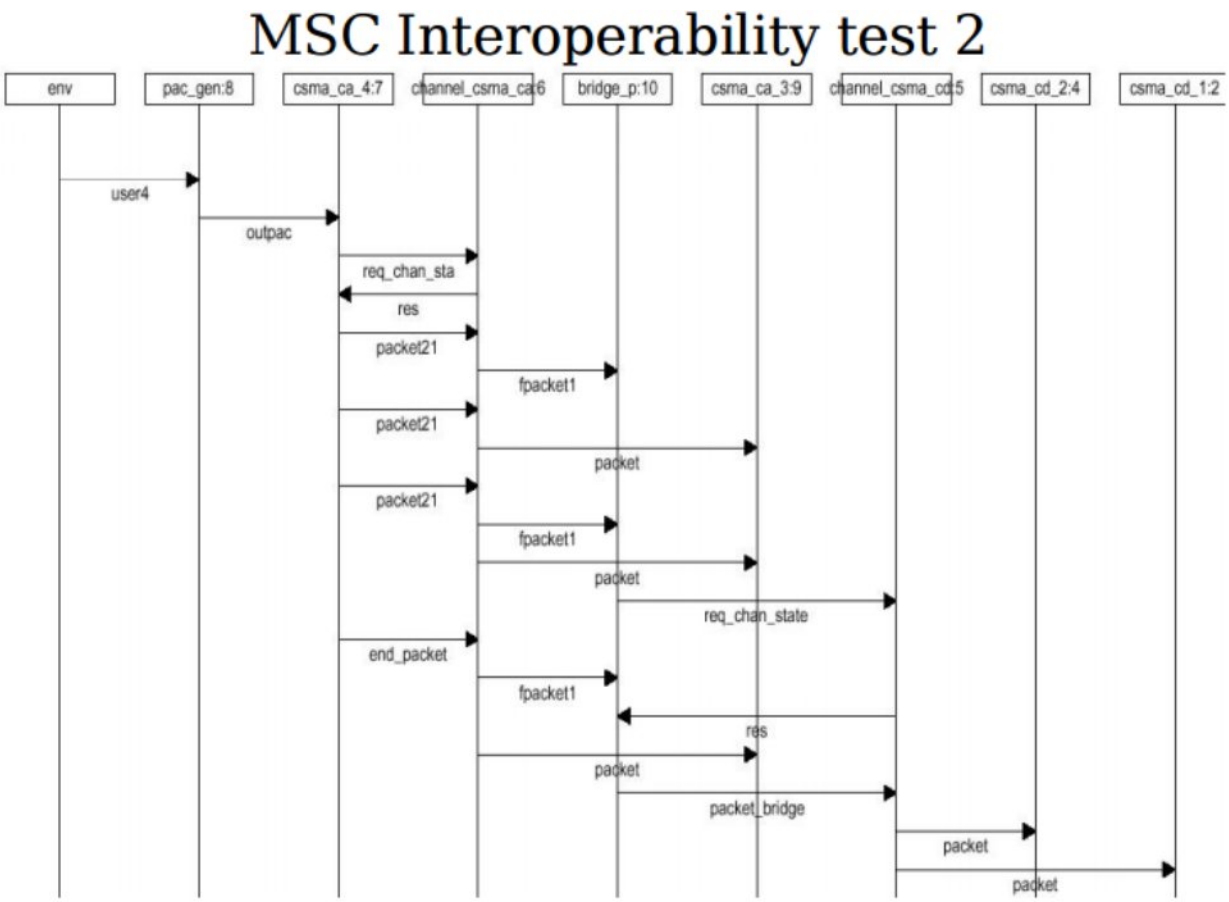
The Transmission Control Protocol (TCP) is an example of a protocol that adjusts its segment size to be smaller than the MTU. The User Datagram Protocol (UDP) and theInternet Control Message Protocol (ICMP) disregard MTU size, thereby forcing IP to fragment oversized datagrams.

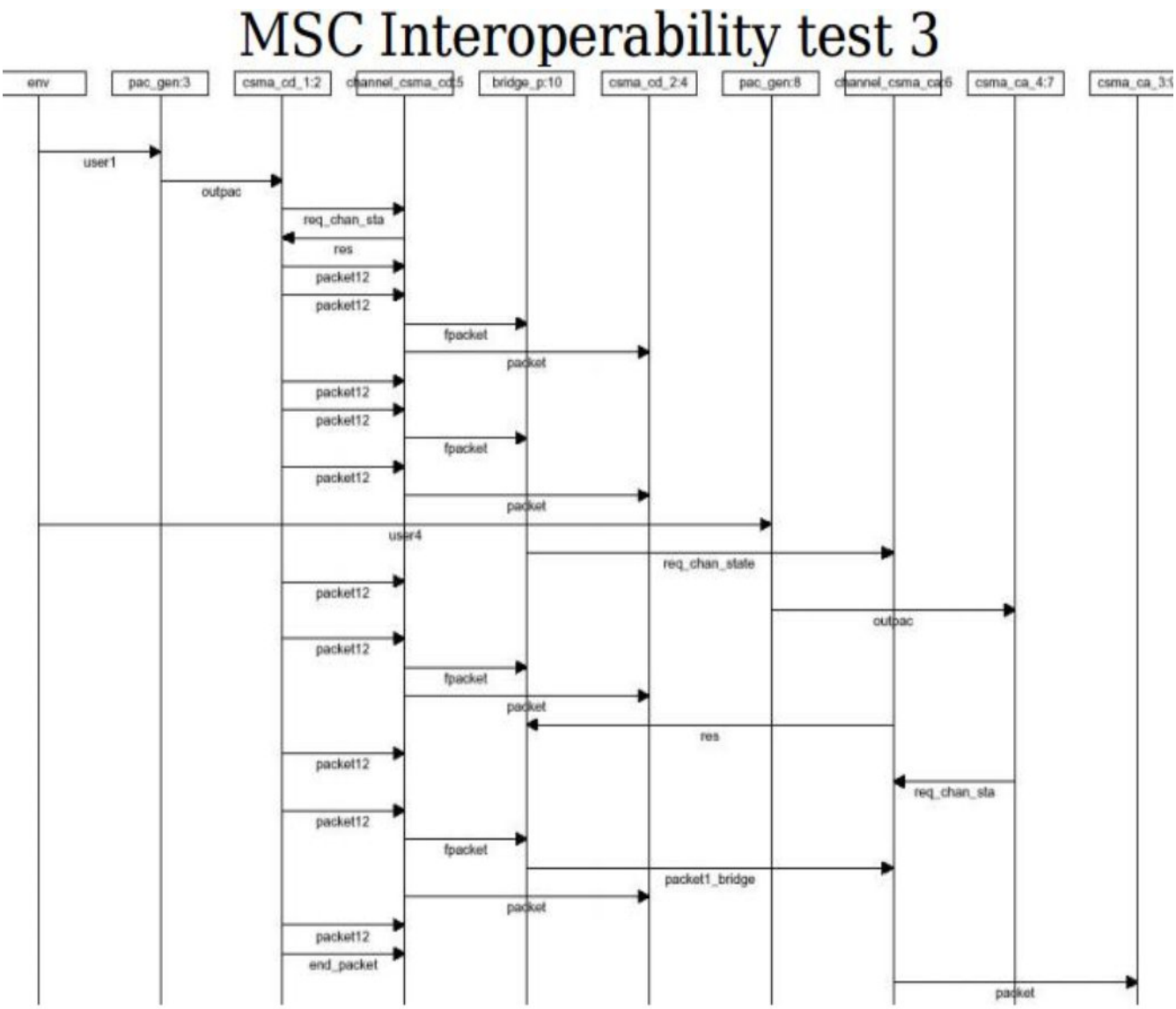
5.8 SDL BASED INTEROPERABILITY TESTING OF CSMA/CD AND CSMA/CA PROTOCOL USING BRIDGE

The procedure for interoperability testing by using SDL is as follows:

- Create FSMs for the protocols and their interoperations using a bridge.
- Create SDL diagrams of the protocols and the bridge operations.
- Run the system by giving the inputs (test sequence) from the environment such as source, destination and number of packets.
- Create the MSCs for the different kinds of inputs.
- Observe from MSCs interworking of CSMA/CA and CSMA/CD using a bridge.


MSC Interoperability test 1
env
pac_gen:3
csma_cd_1:2
channel_csma_cd:5
bridge_p:10
csma_cd_2:4
channel_csma_ca:6
csma_ca_3:9
user1
outpac
req_chan_sta
res
packet12
packet12
fpacket
packet
packet12
end_packet
fpacket
packet
req_chan_state
fpacket
packet
res
end_pac
packet1_bridge
packet
packet


MSC Interoperability test 2
env
pac_gen:8
csma_ca_4:7
channel_csma_ca:6
bridge_p:10
csma_ca_3:9
channel_csma_cd:5
csma_cd_2:4
csma_cd_1:2
user4
outpac
req_chan_sta
res
packet21
fpacket1
packet21
packet
packet21
fpacket1
packet
req_chan_state
end_packet
fpacket1
res
packet
packet_bridge
packet
packet



Results

- MSC for the movement of 3 packets from node 1 (on CSMA/CD) to node 3 (on CSMA/CA). The bridge checks the destination of the packet, if it is on the same link, then it will not forward the packet.
    - In case destination is on different link, the bridge buffers the packet until CSMA/CA channel is free.
    - Once the channel is free, packets are sent one by one to the receiver
- MSC for the movement of 3 packets from node 4 (CSMA/ CA) to node 2 (on CSMA/CD).
    - The bridge is forwarding packets reliably to the node 2
- MSC for the movement of packets in both directions simultaneously (from node 1 to 3, and node 4 to 2).

- Bridge is able to handle the data transfer reliably for both sides.

5.8 SCALABILITY TESTING

Scalability refers to the ability of communication protocol to support the network even when the network size grows without consuming much of the resources such as bandwidth and buffers. −Large networks like internet has some scalability limitation when it comes to managing individual traffic flows. Internet protocol RSVP is believed to be non scalable.

Scalability Testing of BGP:

BGP is designed to support scalability. The first item to test is the routing table capacity as the internet currently has 120,000 routes and still continues to grow. A BGP tester should generate Ipv4 routers and then validate the capacity of the local router to correctly process and store these routes. −The upper limits should be discovered and communicated to the appropriate network architects. −The current Internet routing table should also be loaded into the router so that real world scalability can be verified.

The following are the test procedures used in scalability testing of BGP.

Test case 1:

Initially a baseline set of tests was performed which determined the number of routes which could be held in a VRF (Virtual routing and forwarding) table relative to the number of VRFs that were conFig.d on the router.

Test case 2: Once the baseline set of tests has been performed the configuration on the router was altered so that each VRF had a firewall filter and a counter on its outgoing interface. This subset of tests was performed with 5, 10 and 15 filter statements and counters per VRF.

Test case 3: The third set of tests involved altering the configuration again by adding a policer to each VRF in addition to the filter and counter added in test case 2. Once this was established a similar subset of tests to test case 2 was performed

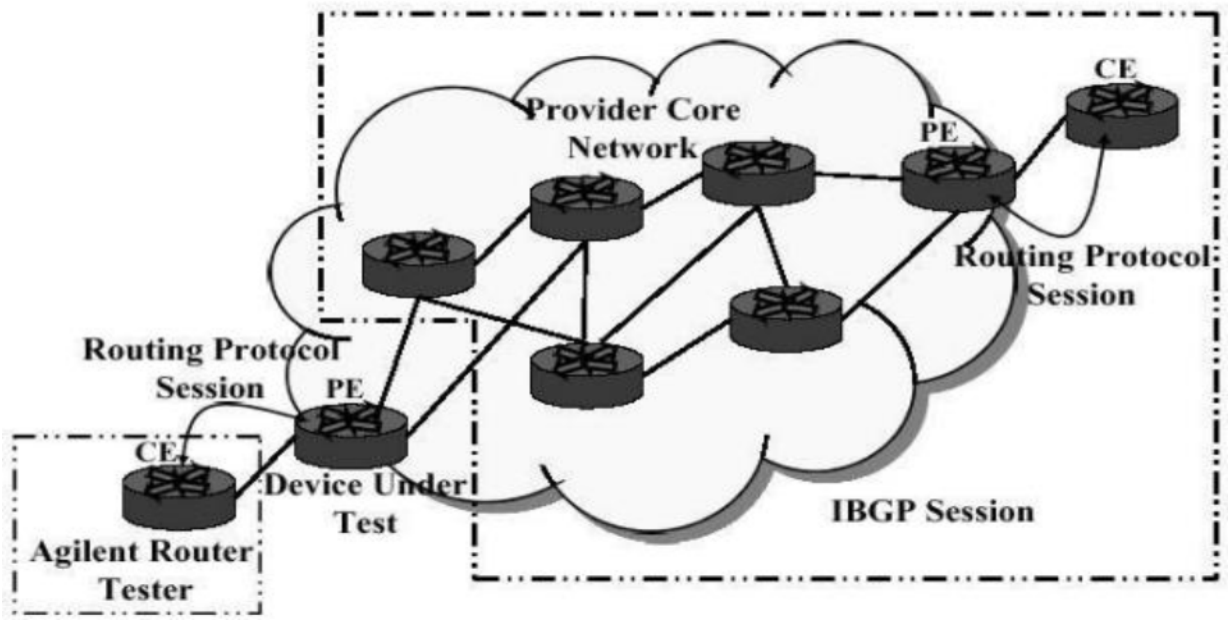


Fig 5.23 BGP layer 3 MPLS VPN network

Question for practice

UNIT V
PART A

1. What are the services provided by PGP services
2. Why E-mail compatibility function in PGP needed?
3. What are the classifications of Web security threats?
4. What are the basic issues in web security?
5. Define CSMA/CD.
6. Define CSMA/CA.
7. What is the subdivision of authentication protocol?
8. Define SDL.
9. What are the social issues in network security?
10. Define S/MIME.

PART B

1. Explain in detail various authentication protocol.
2. Discuss the Web security in networks.
3. Explain in detail about Email security in networks.
4. Explain in detail about SDL based interoperability testing of CSMA/CD and CSMA/CA protocol using Bridge
5. Explain SDL based protocol verification and validation.