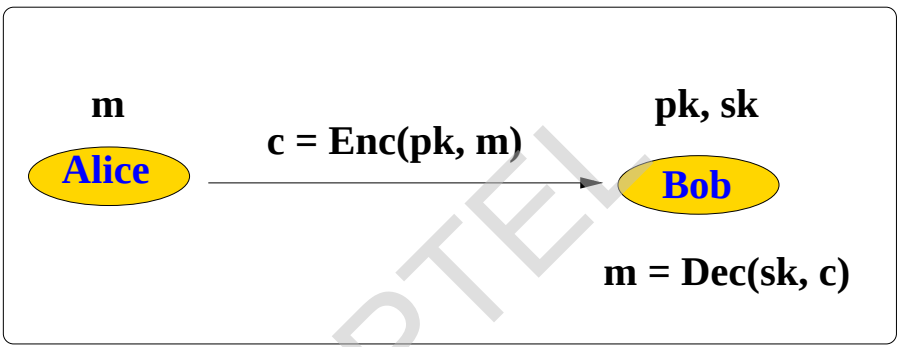
Functional Encryption (Introduction)

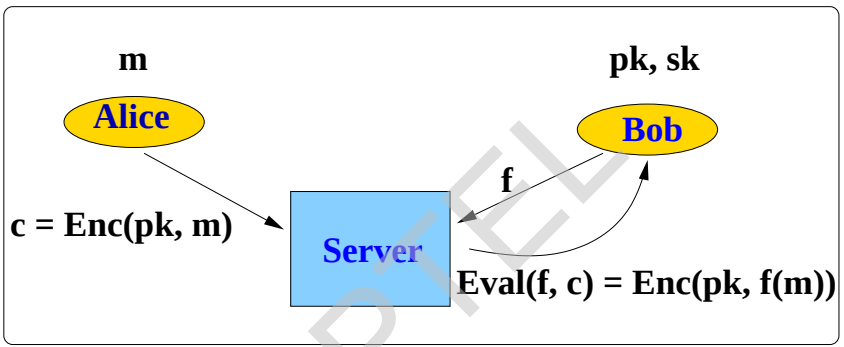
Public Key Encryption (PKE)



Drawback:

- Decryption is "all" or "nothing" affair!

Homomorphic Encryption (HE)



Drawback:

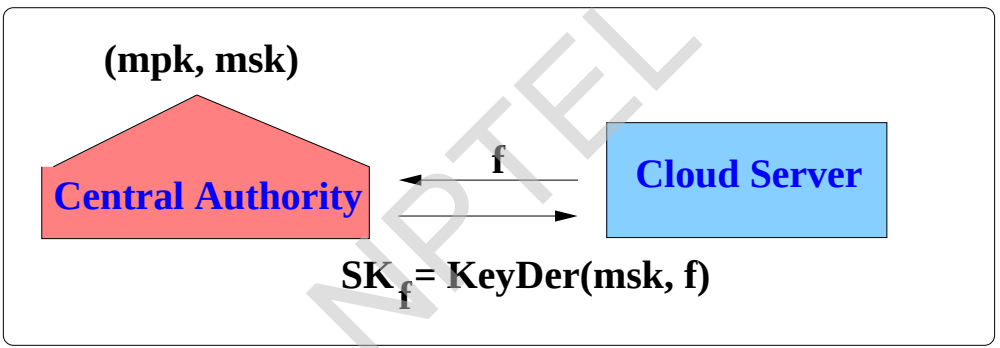
- Interaction with Bob!

Example

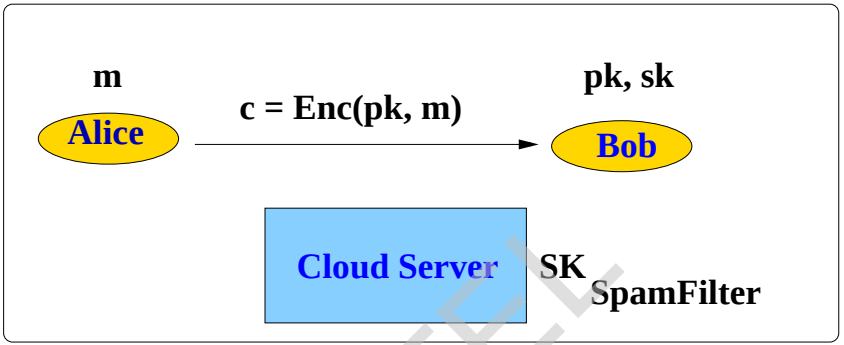
- f is + operator, $c_1 = \mathsf{Enc}(\mathsf{pk}, m_1)$ and $c_2 = \mathsf{Enc}(\mathsf{pk}, m_2)$ $\mathsf{Eval}(f, c_1, c_2) = \mathsf{Enc}(\mathsf{pk}, f(m_1, m_2)) = \mathsf{Enc}(\mathsf{pk}, m_1 + m_2)$
- Useful to outsource private computations (cloud computing)
- **Partially homomorphic cryptosystems** - RSA, ElGamal, GM, Paillier
- **Fully homomorphic encryption** - supports arbitrary computation on ciphertexts (lattice-based cryptography)

Functional Encryption (FE)

(Delegates decryption capabilities)



Functional Encryption (FE)



$$\textit{if}\ \mathsf{Eval}(\mathsf{SK}_{SpamFilter}, c) = \mathsf{True}$$
$$\textit{then}\ \text{“Move to the Spam Folder”}$$

Advantages

- Decryption does not require interaction with Bob!
- Fine-Grained Access Control of Decryption Capabilities!

FE: Credit Card Transaction Alert

- Credit Card Tranaction Alert ($\mathsf{SK}_{\mathsf{Alert}}$)

$$\textit{if}\ \mathsf{Eval}(\mathsf{SK}_{\mathsf{Alert}}, c) = \mathsf{True}$$
$$\textit{then}\ \text{“Fire an Alarm”}$$

Alert: Transactions over Rs. 1.0 Lakhs

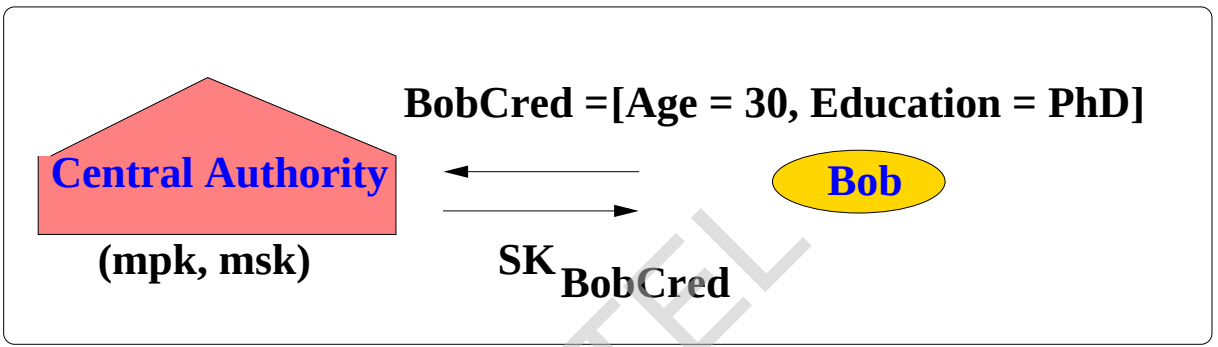
FE: Credit Card Fraud Investigation

- Credit Card Fraud Investigation ($\mathsf{SK}_{f_{\mathsf{Auditing}}}$)

$$\textit{if}\ \mathsf{Eval}(\mathsf{SK}_{f_{\mathsf{Auditing}}}, c) = \mathsf{True}$$
$$\textit{then}\ \text{“Fire an Alarm”}$$

f_{Auditing}: Transactions over Rs. 1.0 Lakhs which took place in November and originated from Kolkata.

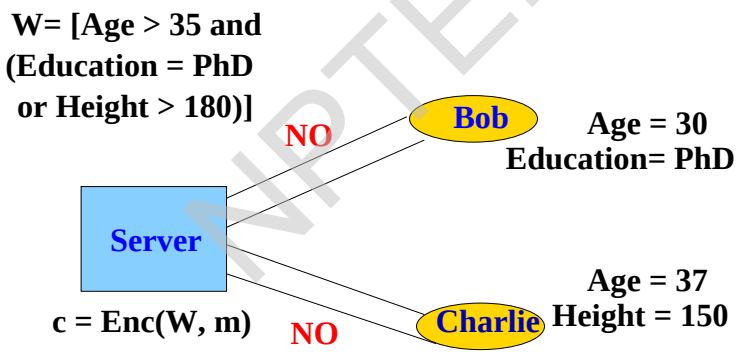
FE: Online dating



Bob has specific attributes and will receive a secret key that can only decrypt profiles for which the attributes match the dating preferences.

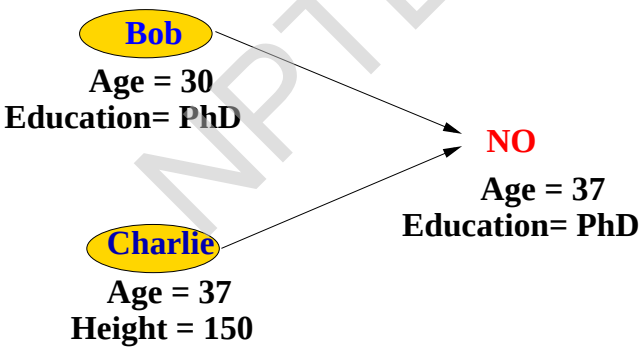
FE: Online dating

- profile m is encrypted under the dating preferences (access structure) $W = [$ Age > 35 and Education $=$ PhD or Height $> 180)]$



FE: Online dating - Collusion Resistance

- profile m is encrypted under the dating preferences (access structure) $W = [$ Age > 35 and (Education $=$ PhD or Height $> 180)]$
- primitive should withstand collusion attack



Current Lines of Work

- Efficient functional encryption for access control
- Functional encryption for all circuits
- Efficient constructions for expressive functionalities

FE: Definition

A Functional Encryption (FE) scheme for the functionality $\mathcal{F}$ consists of the following algorithms:

$$(\mathsf{mpk}, \mathsf{msk}) \longleftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$$

$$\mathsf{SK}_f \longleftarrow \mathsf{KeyDer}(\mathsf{msk}, f)$$

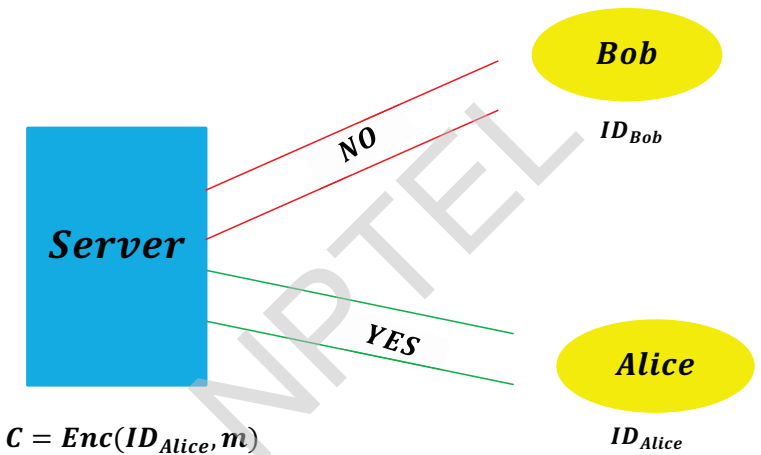
$$\mathsf{CT} \longleftarrow \mathsf{Enc}(\mathsf{mpk}, m)$$

$$f(m) \longleftarrow \mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT})$$

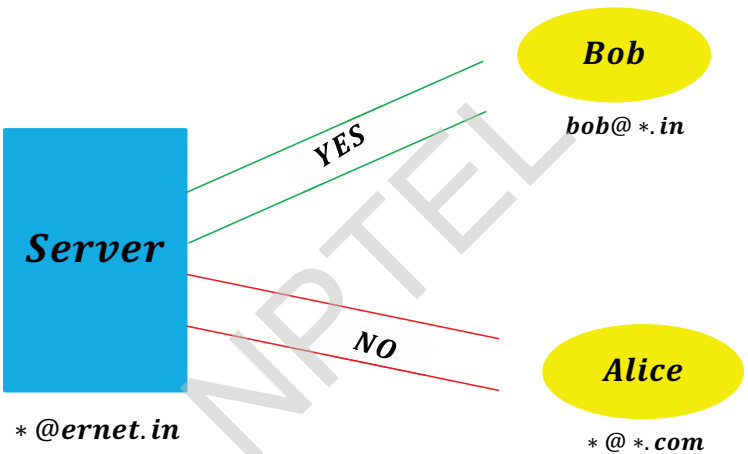
Examples of Functionalities

- (Hierarchical) Identity-Based Encryption
- Fuzzy Identity-Based Encryption
- Attribute-Based Encryption
- Predicate Encryption
- etc.

Identity-Based Encryption (IBE)



Generalized Hierarchical IBE (HIBE)



Implementation Attacks

- Implementation attacks take on a different approach to the above for discovering the secret key.

- Instead of attacking the mathematical properties of the algorithm these form of attacks (also known as side channel attacks) take advantage of the physical phenomena that occurs when a cryptographic algorithm is implemented in hardware.

- Four side channel attacks are listed in the FIPS standard 140-2 "Security Requirements for Cryptographic Modules", **Power Analysis**, **Timing Analysis**, **Fault Induction** and **TEMPEST**.

- Here we will be interested mainly in Differential Power Analysis (DPA) as it applies to DES however we will have a brief look at Timing attacks.

Differential Power Analysis

- Power Analysis is a relatively new concept but has proven to be quite effective in attacking smartcards and similar devices.
- The smartcard is very susceptible to this form of attack mainly because it applies little or no power filtering due to its small size.
- It was first demonstrated by Ernst Bovelander in 1997 but a specific attack strategy was not given.
- A year later it was brought to the general public's attention by Paul Kocher and the Cryptographic Research team in San Francisco.
- Kocher et al. provided an attack strategy that would recover the secret key from cryptographic systems running the DES algorithm.

- This caused great concern amongst the smartcard community and a search for an effective countermeasure began.

- To date a limited number of countermeasures have been proposed and none are fully effective.

- The attacks work equally well on other cryptographic algorithms as shown by Thomas Messerges et al. who presented a great deal of supplementary research on the subject.

- Power analysis involves an analysis of the pattern of power consumed by a cryptographic module as it performs its operations.

- The purpose of this pattern analysis is to acquire knowledge about causal operations that is not readily available through other sources.

- The power consumption will generally be different for each operation performed (and even for the same operations with different data values).

- One of the causes of these variations is the transistor technology used to implement the module.

- The transistors act as voltage controlled switches, and the power they consume varies with the type of instructions being processed.

- For example, a conditional branch instruction appears to cause a lot of noticeable fluctuations according to Kocher, and should therefore be avoided if possible where secret keys are concerned.

- An example of a setup for a power analysis attack is shown in figure 3.

- For smartcards and similar devices, the power can be measured across a $10 - 50\Omega$ resistor in series with the power or ground line of the specific device.

- The resistor should be small enough so as not to interfere with the operation of the circuit itself, but large enough to give easily observable voltage fluctuations. It is better to put the resistor in series with the ground of the device.

- If the power line is used then two scope probes would be needed and the resultant waveforms substracted.

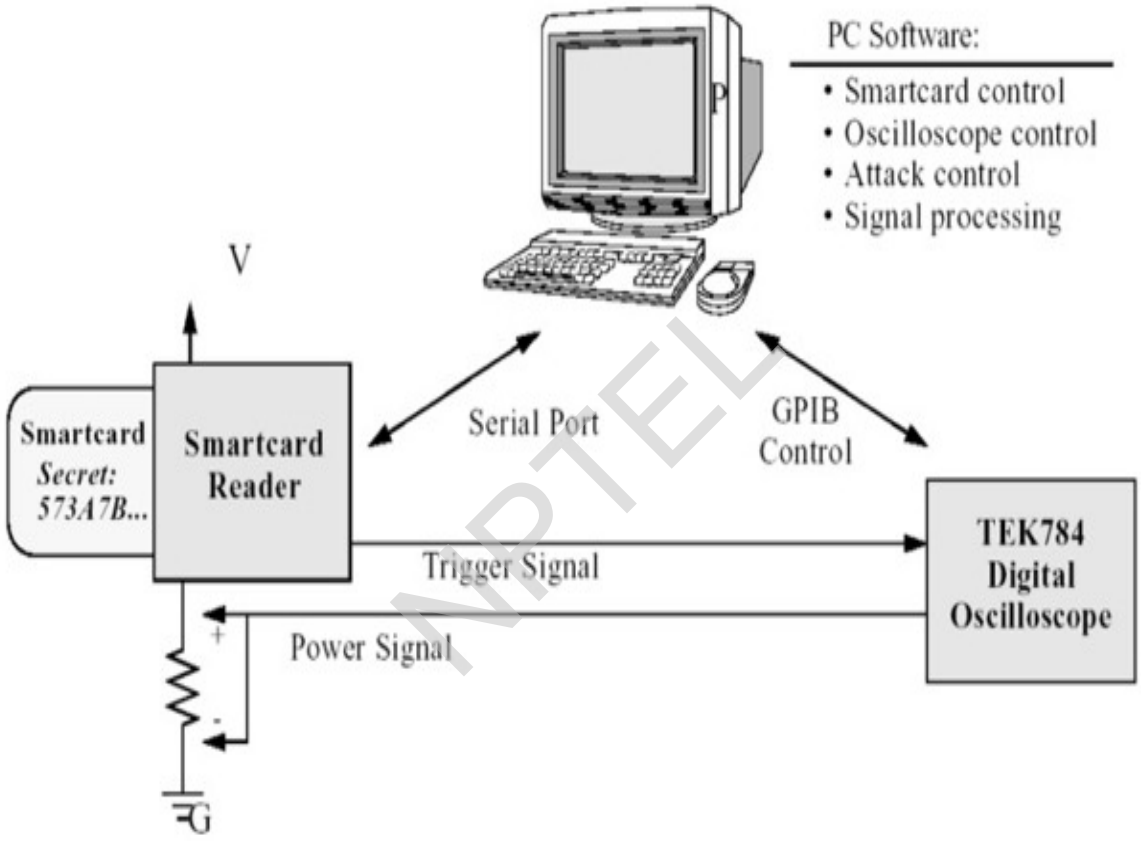


Figure 3: An example setup for a Differential Power Analysis attack on a smartcard.

- Although the setup in figure 3 will suffice for a smartcard it will generally not be this simple for a complex cryptographic accelerator which probably draws its power from the peripheral component interconnect (PCI) backplane of a computer.

- Ideally, the attacker would wish to get as close as possible to the actual chip performing the operations if a high signal to noise ratio (SNR) is to be obtained.

- This might be more difficult than it first appears as information on which of the boards numerous chips is actually running the algorithm may not be readily available.

- Even if it were, the power pin of the chip would have to be physically separated from the board to perform the attack and then reattached once complete (if the attack were to go unnoticed).

- Most tamper resistant devices would not permit this from happening.
- An example of a possible setup is shown in figure 4.
- In this case a PCI extender board is used to measure the power fluctuations.
- The actual cryptographic board slots into the extender board and therefore the power the cryptographic board draws from the PCI backplane has to flow through the extender board which can be fitted with some points that allow for measurement of the power.
- These can be home made or easily purchased.

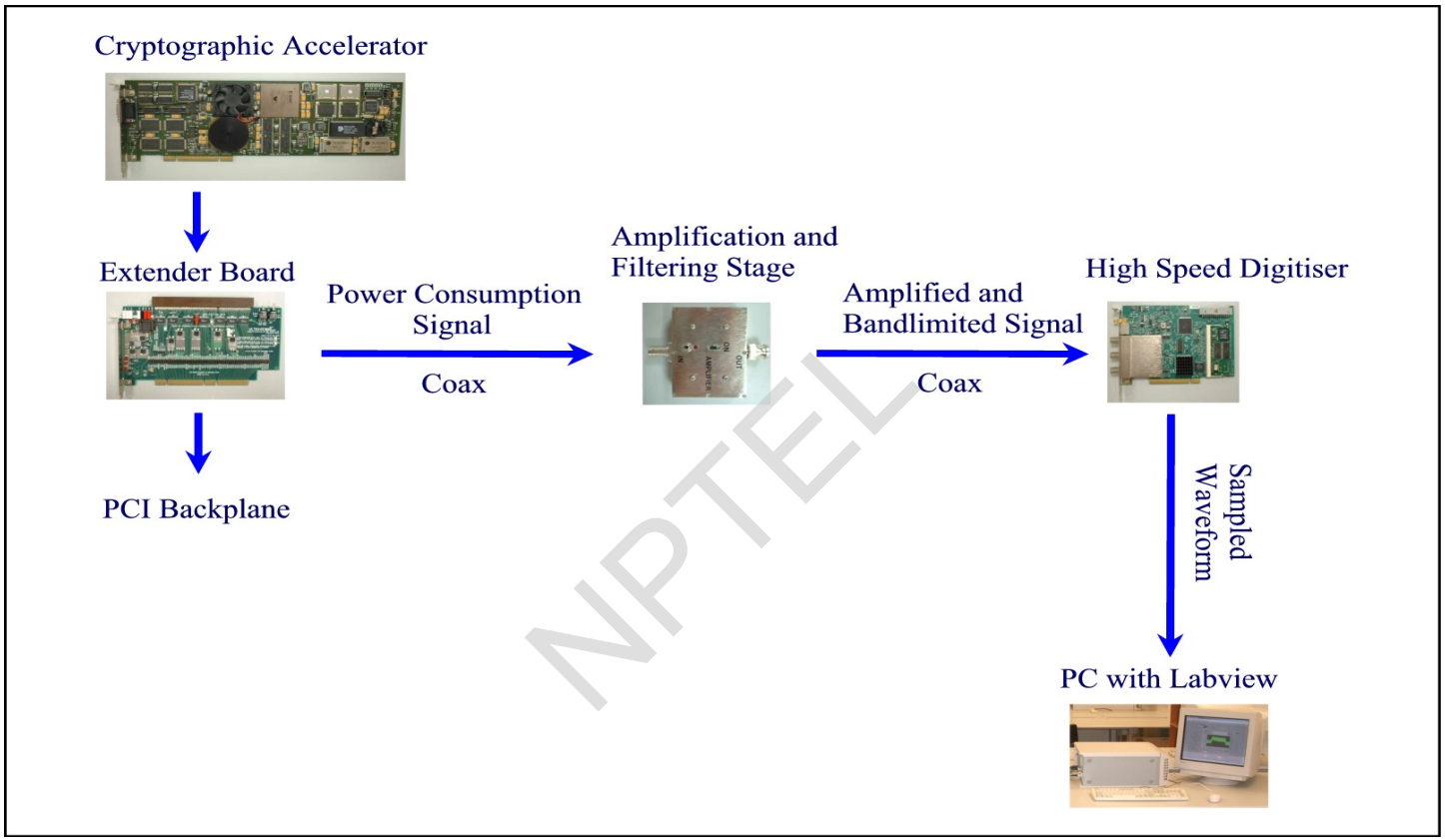


Figure 4: An example setup for a Differential Power Analysis attack on a high speed cryptographic accelerator.

- Assuming a setup such as those in figures 3 and 4 in which the algorithm being executed is the Data Encryption Standard (DES) the attack can proceed as follows.

- A method must be devised to produce a random set of J plaintext inputs that can be sent to the cryptosystem for encryption.

- This method must be automated as the number of random plaintext inputs will be quite large. Generally this will be the job of the PC however on more complex cryptosystems it may be possible to upload new firmware that will do the trick.

- On receiving these plaintext inputs, pi_j, $1 \leq j \leq J$, the board will begin to run its algorithm and draw varying amounts of power.

- These power fluctuations can be sampled using a digital sampling oscilloscope which should be capable of sampling at about 20-30 times the clock frequency being used.

- There are two main reasons for this:

  1. Possible that we might have multiple operations occuring in each clock cycle. Also, operation of interest might only last a small fraction of the clock cycle.
  2. The more samples you have per cycle the less chance of noise caused by a misalignment of samples.

- The waveforms observed for each pi_j can be represented as a matrix wf_{jk}, where $1 \leq k \leq K$.

- The subscripts j and k are used to identify the plaintext number causing

the waveform and the time sample point within that particular waveform, respectively.

- A second column matrix, co_j, can also be used to represent the ciphertext output.

- In practice, each row of wf_{jk} would probably be stored as a separate file for ease of processing.

- Having captured each power waveform and ciphertext output, a function known as a **partitioning function**, D(.), must now be defined.

- This function will allow division of the matrix wf_{jk} into two sub-matrices $wf0_{pk}$ and $wf1_{qk}$ containing P and Q rows respectively, with $1 \leq p \leq P$ and $1 \leq q \leq Q$ where $P + Q = J$.

- Provided that the inputs pi_j were randomly produced, then $P = Q = J/2$ as $J \rightarrow \infty$ (i.e. the waveforms will be divided equally between the two sets).

- The partitioning function allows the division of wf_{jk} because it calculates the value of a particular bit, at particular times, during the operation of the algorithm.

- If the value of this bit is known, then it will also be known whether or not a power bias should have occurred in the captured waveform.

- For a 1, a bias should occur, and for a 0 it shouldn't.

- Separating the waveforms into two separate matrices (one in which the bias occurred and another in which it didn't) will allow averaging to reduce the noise and enhance the bias (if it occurred).

- For randomly chosen plaintexts, the output of the D(.) function will equal either a 1 or 0 with probability $\frac{1}{2}$ (this is just another statement of the fact that $P = Q = J/2$ as $J \rightarrow \infty$).

- An example of a partitioning function is:

$$D(C_1, C_6, K_{16}) = C_1 \oplus SBOX1(C_6 \oplus K_{16}) \tag{8}$$

- Where SBOX1(.) is a function that outputs the target bit of S-box 1 in the last round of DES (in this case it's the first bit), C_1 is the one bit of co_j that is exclusive OR'ed with this bit, C_6 is the 6 bits of co_j that is exclusive OR'ed with the last rounds subkey and K_{16} is the 6 bits of the last round's subkey that is input into S-box 1.

- The value of this partitioning function must be calculated at some point throughout the algorithm.

- So, if the values C_1, C_6 and K_{16} can be determined, it will be known whether or not a power bias occurred in each waveform.

- It is assumed that the values C_1 and C_6 can be determined and the value of the subkey K_{16} is the information sought.

- To find this, an exhaustive search needs to be carried out. As it is 6 bits long, a total of $2^6 = 64$ subkeys will need to be tested.

- The right one will produce the correct value of the partitioning bit for every plaintext input.

- However, the incorrect one will only produce the correct result with probability $\frac{1}{2}$.

- In this case, the two sets $wf0_{pk}$ and $wf1_{qk}$ will contain a randomly

distributed collection of waveforms which will average out to the same result (Provided the plaintext inputs are randomly chosen).

- The differential trace (discussed below) will thus show a power bias for the correct key only.

- Of course it means that 64 differential traces are needed but this is a vast improvement over a brute force search of the entire 56 bit key.

- Mathematically, the partitioning of wf_{jk} can be represented as

$$wf0_{pk} = \{wf_{jk} | D(.) = 0\} \tag{9}$$

and

$$wf1_{qk} = \{wf_{jk} | D(.) = 1\} \tag{10}$$

- Once the matrices $wf0_{pk}$ and $wf1_{qk}$ have been set up, the average of each is then taken producing two waveforms $awf0_k$ and $awf1_k$ both consisting of K samples.

- By taking the averages of each, the noise gets reduced to very small levels but the power spikes in $wf1_{pk}$ will be reinforced.

- However, averaging will not reduce any periodic noise contained within the power waveforms and inherent to the operations on the cryptographic board.

- This can largely be eliminated by subtracting $awf0_{pk}$ from $awf1_{qk}$ (this can be thought of as demodulating a modulated signal to reveal the "baseband", where the periodic noise is the "carrier").

- The only waveform remaining will be the one with a number of bias

points identifying the positions where the target bit was manipulated.

- This trace is known as a **differential trace**, ΔD_k.

- Again, in mathematical terms, the above can be stated as

$$awf0_k = \frac{1}{P} \sum_{wf_{jk} \in wf1} wf_{jk} = \frac{1}{P} \sum_{p=1}^{P} wf0_{pk} \tag{11}$$

and

$$awf1_k = \frac{1}{Q} \sum_{wf_{jk} \in wf0} wf_{jk} = \frac{1}{Q} \sum_{q=1}^{Q} wf1_{qk} \tag{12}$$

- The differential trace ΔD_k is then obtained as

$$\Delta D_k = awf1_k - awf0_k \tag{13}$$

- The last five equations can now be condensed into one:

$$\Delta D_k = \frac{\sum_{j=1}^{J} D(.) wf_{jk}}{\sum_{j=1}^{J} D(.)} - \frac{\sum_{j=1}^{J} (1 - D(.)) wf_{jk}}{\sum_{j=1}^{J} (1 - D(.))} \tag{14}$$

- As $J \rightarrow \infty$, the power biases will average out to a value ϵ which will occur at times k_D - each time the target bit D was manipulated.

- In this limit, the averages $awf0_k$ and $awf1_k$ will tend toward the expectation $E\{wf0_k\}$ and $E\{wf1_k\}$, and equations 13 and 14 will

converge to

$$E\{wf1_k\} - E\{wf0_k\} = \epsilon, \qquad \text{at times} \quad k = k_D \tag{15}$$

and

$$E\{wf1_k\} - E\{wf0_k\} = 0, \qquad \text{at times} \quad k \neq k_D \tag{16}$$

- Therefore, at times $k = k_D$, there will be a power bias ϵ visible in the differential trace. At all other times, the power will be independent of the target bit and the differential trace will tend towards 0.

- The above will only work if the subkey guess was correct. For all other guesses the partitioning function will separate the waveforms randomly, and equations 15 and 16 will condense to

$$E\{wf1_k\} - E\{wf0_k\} = 0, \qquad \forall k \tag{17}$$

- As mentioned above, 64 differential traces are needed to determine which key is the correct one.

- Theoretically, the one containing bias spikes will allow determination of the correct key however, in reality the other waveforms will contain small spikes due to factors such as non-random choices of plaintext inputs, statistical biases in the S-boxes and a non-infinite number of waveforms collected.

- Generally however, the correct key will show the largest bias spikes and can still be determined quite easily.

- The other 42 bits from the last round's subkey can be determined by applying the same method to the other 7 S-boxes.

- A brute force search can then be used to obtain the remaining 8 bits of the 56 bit key.

- NOTE: The same J power signals can be used for each S-box as the different D functions re-order them accordingly.

Timing Attacks

- A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.

- We can explain the attack using the modular exponentiation algorithm shown in figure 5, but the attack can be adapted to work with any implementation that does not run in fixed time.

- In this algorithm, modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit.

```
square_and_mul(b, e, m)
{
    d = 1;
    for (k = N-1 downto 0)
    {
        d = (d × d) mod m;
        if (e[k] == 1)
        {
          d = (d × b) mod m;
        }
    Return d;
}
```


Figure 5: Square and Multiply algorithm for Computing $b^e \bmod (m)$ where e is N bits long.

- As Kocher points out in his paper, the attack is simplest to understand in an extreme case.

- Suppose the target system uses a modular multiplication function that is very fast in almost all cases but in a few cases takes musch more time than an entire average modular exponentiation.

- The attack proceeds bit by bit starting with the leftmost bit $e[N-1]$. Suppose that the first j bits are known (to obtain the entire exponent, start with $j = 0$ and repeat the attack until the entire exponent is known).

- For a given ciphertext, the attacker can complete the first j iterations of the **for** loop.

- Operation of subsequent steps depends on the unkown exponent bit.

- If the bit is set d = (d × b) mod m will be executed.

- For a few values of b and d, the modular multiplication will be extremely slow, and the attacker knows which these are.

- Therefore, if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1.

- If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0.

- In practice, modular exponentiation implementations do not have such extreme timing variations, in which the execution time of a single iteration can exceed the mean execution time of the entire algorithm.

- Nevertheless, there is enough variation to make this attack practical.

- Although the timing attack is a serious threat, there are simple counter measures that can be used including the following:
  - **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
  - **Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher point out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.
  - **Blinding:** Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analsys essential to the timing

attack.

- RSA Data Security incorportates a blinding feature into some of its products. The private-key operation $M = C^d \bmod n$ is implemented as follows:

  1. Generate a secret random number r between 0 and $n - 1$.
  2. Compute $C' = C(r^e) \bmod n$, where e is the public exponent.
  3. Comput $M' = (C')^d \bmod n$ with the ordinary RSA implementation.
  4. Compute $M = M'r^{-1} \bmod n$ (where r^{-1} is the multiplicative inverse of $r \bmod n$). It can be demonstrated that this is the correct result by observing that $r^{ed} \bmod n = r \bmod n$.

- RSA Data Security reports a 2 to 10% performance penalty for blinding.

THE SECURE SOCKETS LAYER (SSL)

The Secure Sockets Layer (SSL)

- Due to the fact that nearly all businesses have websites (as well as government agencies and individuals) a large enthusiasm exists for setting up facilities on the Web for electronic commerce.
- Of course there are major security issues involved here that need to be addressed.
- As businesses begin to see the threats of the Internet to electronic commerce, the demand for secure web pages grows.
- A number of approaches to providing Web security are possible.

- The various approaches are similar in many ways but may differ with respect to their scope of applicability and relative location within the TCP/IP protocol stack.
- For example we can have security at the IP level making it transparent to end users and applications.
- However another relatively general-purpose solution is to implement security just above TCP.
- The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS).
- Here we look at SSL which was originated by Netscape.

Overview

- As mentioned, the Secure Sockets Layer (SSL) is a method for providing security for web based applications.
- It is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols as illustrated in figure 1.
- It can be seen that one layer makes use of TCP directly. This layer is known as the **SSL Record Protocol** and it provides basic security services to various higher layer protocols.

- An independent protocol that makes use of the record protocol is the **Hypertext Transfer Protocol** (HTTP) protocol.
- Another three higher level protocols that also make use of this layer are part of the SSL stack. They are used in the management of SSL exchanges and are as follows:
  1. Handshake Protocol.
  2. Change Cipher Spec Protocol.
  3. Alert Protocol.

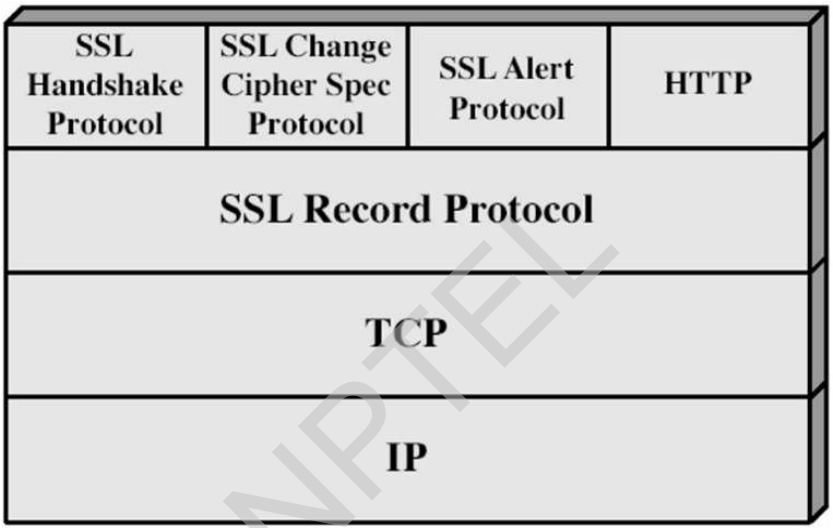


Figure 1: SSL protocol stack.

SSL Record Protocol

- This protocol provides two services for SSL connections:
  1. Confidentiality - using conventional encryption.
  2. Message Integrity - using a Message Authentication Code (MAC).
- In order to operate on data the protocol performs the following actions (see figure 2):
- It takes an application message to be transmitted and fragments it into manageable blocks. These block are $2^{14} = 16,384$ bytes or less.

- These blocks are then optionally compressed which must be lossless and may not increase the content length by more than 1024 bytes.
- A message authentication code is then computed over the compressed data using a shared secret key. This is then appended to the compressed (or plaintext) block.
- The compressed message plus MAC are then encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed $2^{14} + 2048$. A number of different encryption algorithms are permitted.
- The final step is to prepend a header.

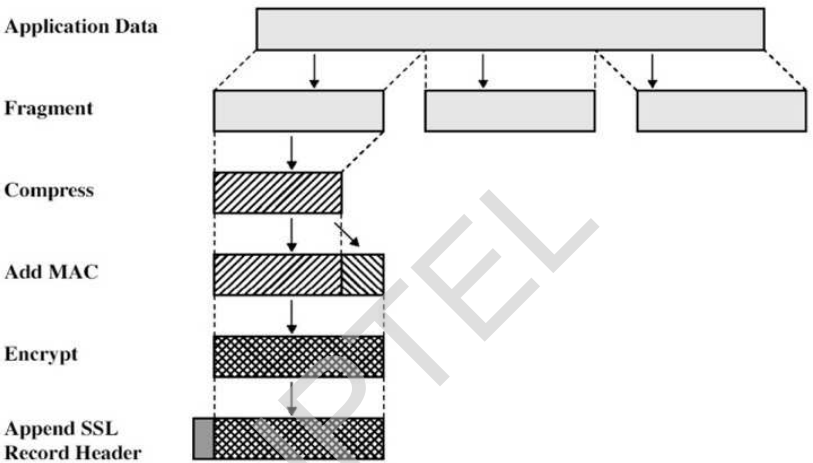


Figure 2: SSL Record Protocol Operation.

- The header consists of the following fields:
  - Content type (8 bits) - The higher layer protocol used to process the enclosed fragment.
  - Major Version (8 bits) - Indicates major version of SSL in use. For SSLv3, the value is 3.
  - Minor Version (8 bits) - Indicates minor version in use. For SSLv3, the value is 0.
  - Compressed Length (16 bits) - The length in bytes of the compressed (or plaintext) fragment.

- The overall format is shown in figure 3.

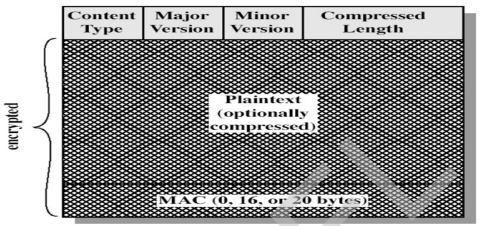


Figure 3: SSL record format.

- The "content type" above is one of four types; the three higher level protocols given above that make use of the SSL record, and a fourth known as "application_data".

Change Cipher Spec Protocol

- This consists of a single message which consists of a single byte with the value 1.
- This is used to cause the pending state to be copied into the current state which updates the cipher suite to be used on this connection.

Alert Protocol

- This protocol is used to convey SSL-related alerts to the peer entity.
- It consists of two bytes the first of which takes the values 1 (warning) or 2 (fatal).
- If the level is fatal SSL immediately terminates the connection.
- The second byte contains a code that indicates the specific alert.

Handshake Protocol

- This is the most complex part of SSL and allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.
- This protocol is used before any application data is sent.
- It consists of a series of messages exchanged by the client and server, all of which have the format shown in figure 5.

- Each message has three fields:
  1. Type (1 byte): Indicates one of 10 messages such as "hello_request" (see figure 4).
  2. Length (3 bytes): The length of the message in bytes.
  3. Content(≥ 0 byte): The parameters associated with this message such version of SSL being used.

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Figure 4: SSL Handshake protocol message types.

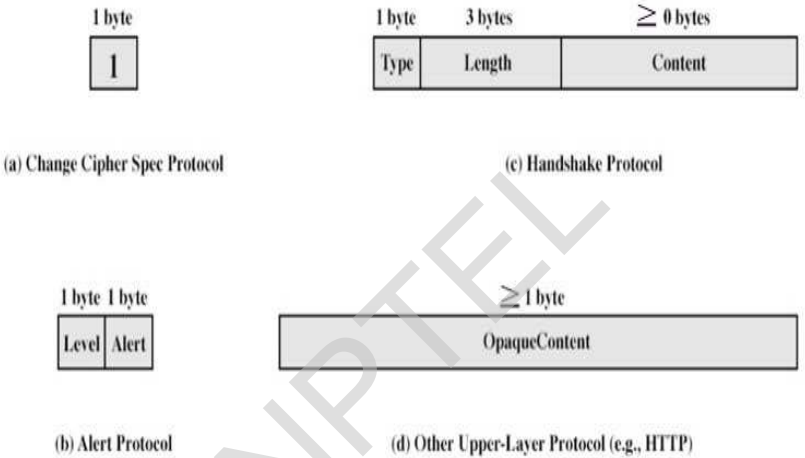


Figure 5: SSL record protocol payload.

Four Phases of Hanshake protocol

1. Establish security capabilities including protocol version, session ID, cipher suite, compression method and initial random numbers. This phase consists of the client_hello and server_hello messages which contain the following (for the client):
   - Version: The highest SSL version understood by client
   - Random: 32-bit timestamp and 28 byte nonce.
   - Session ID: A variable length session identifier.

- CipherSuite: List of cryptoalgorithms supported by client in decreasing order of preference. Both key exchange and CipherSpec (this includes fields such as CipherAlgorithm, MacAlgorithm, CipherType, HashSize, Key Material and IV Size) are defined.
- Compression Method: List of methods supported by client.

2. Server may send certificate, key exchange, and request certificate it also signals end of hello message phase. The certificate sent is one of a chain of X.509 certificates. The

server_key exchange is sent only if required. A certificate may be requested from the client if needs be by certificate_request.

3. Upon receipt of the server_done message, the client should verify that the server provided a valid certificate, if required, and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. The client sends certificate if requested (if none available then it sends a no_certificate alert instead). Next the client sends client_key_exchange message . Finally, the client may

send certificate verification.

4. Change cipher suite and finish. The secure connection is now setup and the client and server may begin to exchange application layer data.

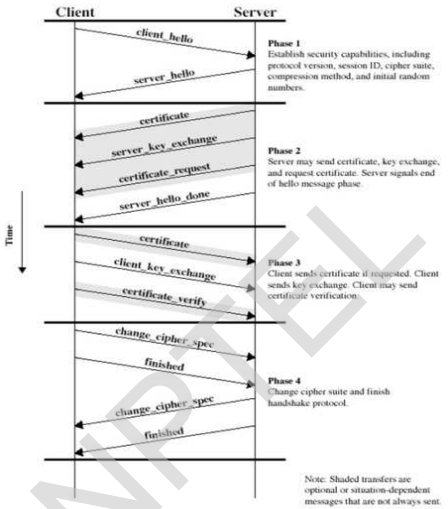


Figure 6: Handshake protocol action.

Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP)

- With the explosively growing reliance on electronic mail for every conceivable purpose, there grows a demand for authentication and confidentiality services.

- Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extension (S/MIME).

- The latter is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security.

- Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organisational use, while PGP will remain the choice for personal e-mail security for many users.

Background

- PGP is largely the effort of a single person, Phil Zimmermann.

- It provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

- In essence what Zimmermann has done is the following:

  1. Selected the best cryptographic mechanisms (algorithms) as building blocks.
  2. Integrated these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy to use commands.

3. Made the package and its source code freely available via the Internet, bulletin boards, and commercial networks such as America On Line (AOL).
4. Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible low cost commercial version of PGP.

- From its beginnings about 15 years ago, PGP has grown explosively and is now very widely used. A number of reasons are sited for such growth:

  1. It is available free worldwide in versions that run on many different platforms, Windows, UNIX, Mac etc. In addition the commercial version satisfies those who want vendor support.
  2. It is based on algorithms that have survived extensive public review and are considered secure. Specifically, the package includes RSA, DSS and Diffie-Hellman for public-key encryption; AES, CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.

3. It has a wide range of applicability, from corporations that wish to select and enforce a standardised scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet.
4. It was not developed by, nor is it controlled by, any government or standards organisation. For those with an instinctive distrust of "the establishment", this makes PGP attractive. In the last few years commercial versions have become available.
5. PGP is now on an Internet standards track (RFC 3156). Nevertheless, PGP still has an aura of an anti-establishment endeavor.

Operational Description

- PGP consists of the following five services:

  1. Authentication
  2. Confidentiality
  3. Compression
  4. E-mail compatibility
  5. Segmentation

- Table 1 shows a summary of these services and the algorithms used to implement them.

- Each service will be discussed in turn and then we will look at the problem of key management.

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.
Segmentation	—	To accommodate maximum message size limitations, PGP performs segmentation and reassembly.

Figure 1: Summary of PGP services.

Authentication

- Figure 2a illustrates the digital signature service provided by PGP.
- The hash function used is SHA-1 which creates a 160 bit message digest. EP (DP) represents public encryption (decryption) and the algorithm used can be RSA or DSS (recall that the DSS can only be used for the digital signature function and unlike RSA cannot be used for encryption or key exchange).
- The message may be compressed using and algorithm called **ZIP**. This is represented by "Z" in the figure.
- The combination of SHA-1 and RSA provides an effective digital signature scheme.

- Due to the strength of RSA the recipient is assured that only the possessor of the matching private key can generate the signature.

- Because of the strength of SHA-1 the recipient is assured that no one else could generate a new message that matches the hash code and hence, the signature of the original message.

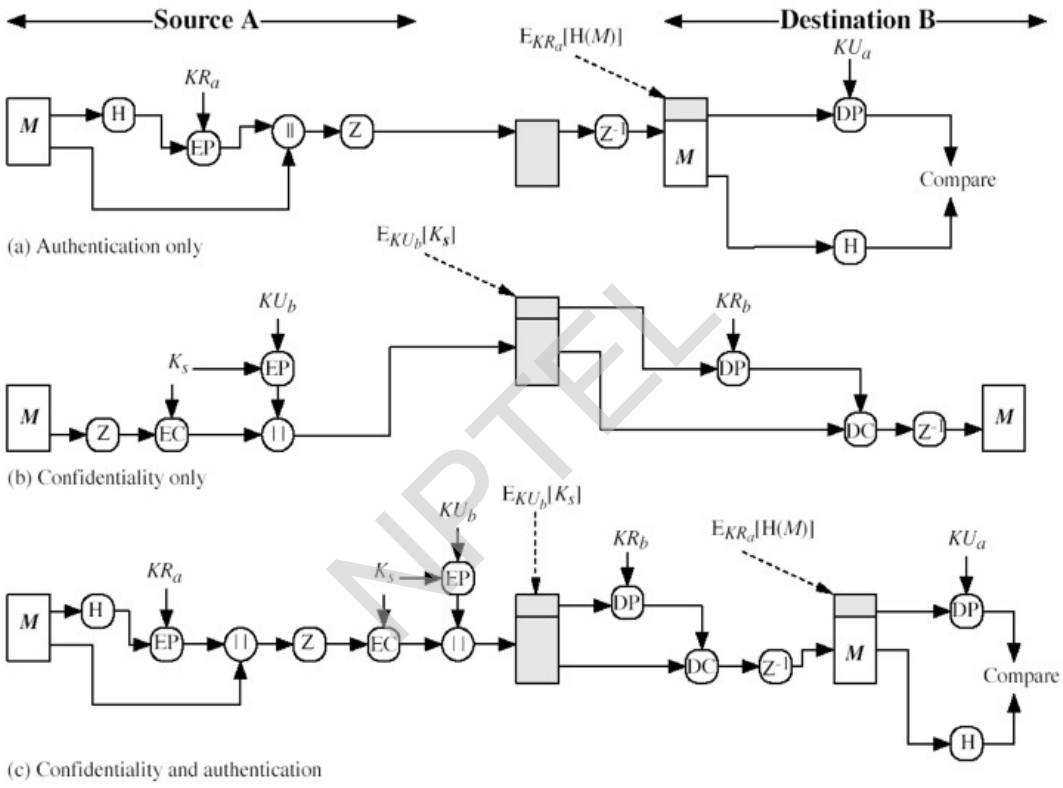


Figure 2: PGP cryptographic functions.

Confidentiality

- Another basic service provided by PGP is confidentiality which is provided by encrypting messages to be transmitted or to be stored locally as files.
- In both cases, the user has a choice of AES, CAST-128, IDEA or 3DES in 64 bit cipher feedback (CFB) mode.
- The symmetric key is used only once and is created as a random number with the required number of bits.
- It is transmitted along with the message and is encrypted using the recipients public key.
- Figure 2c illustrates the sequence:

1. The sender generates a message and a random number to be used as a session key for this message only.
2. The message is encrypted using AES, CAST-128, IDEA or 3DES with the session key.
3. The session key is encrypted with RSA (or another algorithm known as EIGamal) using the recipients public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.
6. As mentioned before, public key encryption is a lot more computationally intensive than symmetric encryption.
7. For this reason both forms are used as public key encryption solves the key distribution problem.
8. Message is encrypted using symmetric key cryptography whereas only the key is encrypted using the public key algorithm.

Confidentiality and Authentication

- As figure 2c illustrates, both services may be used for the same message.
- First, a signature is generated for the plaintext message and prepended to the message.
- Then the plaintext message plus signature is encrypted using AES (or CAST-128, IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).

- This sequence is preferable to the opposite: encrypting the message and then generating a signature of the encrypted message.

- It is generally more convenient to store a signature with a plaintext version of a message.

- Furthermore, for purposes of third party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.

Compression

- As a default, PGP compresses the message after applying the signature but before encryption.

- This has the benefit of saving space both for e-mail transmission and for file storage.

- The placement of the compression algorithm, indicated by Z for compression and Z^{-1} for decompression in figure 2 is critical:

1. The signature is generated before compression for two reasons:
   (a) It is preferable to sign an uncompressed message so it is free of the need for a compression algorithm for later verification.
   (b) Different version of PGP produce different compressed forms. Applying the hash function and signature after compression would constrain all PGP implementation to the same version of the compression algorithm.
2. Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-mail compatibility

- Many electronic mail systems only permit the use of blocks consisting of ASCII text.

- When PGP is used, at least part of the block to be transmitted is encrypted. This basically produces a sequence of arbitrary binary words which some mail systems won't accept.

- To accommodate this restriction PGP uses and algorithm known as **radix64** which maps 6 bits of a binary data into and 8 bit ASCII character.

- Unfortunately this expands the message by 33% however, with the compression algorithm the overall compression will be about one third (in general).

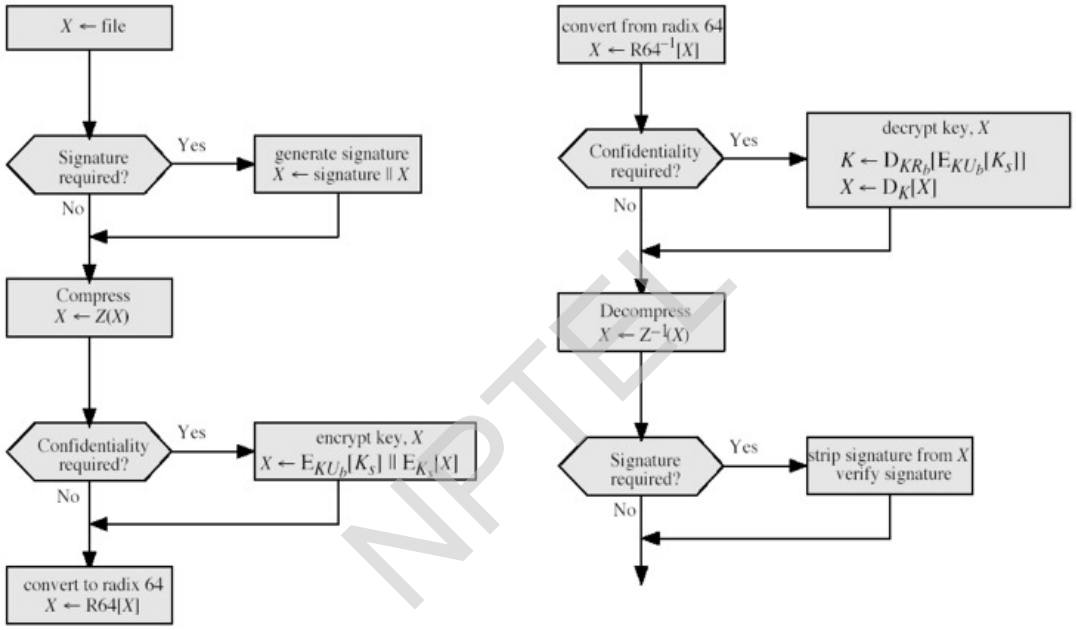
Segmentation

- E-mail facilities are often restricted to a maximum message length.
- For example, many of the facilities accessible throughout the Internet impose a maximum length of 50,000 octets.
- Any message longer than that must be broken up into smaller segments, each of which is mailed separately.
- To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to sent via e-mail.

- The segmentation is done after all the other processing, including the radix-64 conversion.

- Thus the session key component and signature component appear only once, at the beginning of the first segment.

- At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the steps illustrated in figure 3.



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

Figure 3: Transmission and Reception of PGP messages.

Cryptographic Keys and Key Rings

- PGP makes use of four types of keys:

  1. One-time session symmetric keys
  2. Public keys
  3. Private keys
  4. Passphrase based symmetric keys

- Three separate requirements can be identified with respect to these keys:

  1. A means of generating unpredictable session keys is needed
  2. We would like to allow a user to have multiple public-key/private-key pairs. As a result there is not a one-to-one correspondence between users and their public keys. Thus, some means is needed for identifying particular keys.
  3. Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

Session key generation

- Each session key is associated with a single message and is used only for the purpose of encryption and decrypting that message.
- Recall that message encryption/decryption is done with a symmetric encryption algorithm.
- Assuming it is a 128 bit key that is required, the random 128 bit numbers are generated using CAST-128.

- The input to the random number generator consists of as 128-bit key (this is a random number using the keystroke input from the user) and two 64-bit blocks that are treated as plaintext to be encrypted.

- Using CFB mode two 64-bit cipher text blocks are produced and concatenated to form the 128 bit session key.

- The algorithm that is used is based on the one specified in ANSI X12.17.

Key Identifiers

- As mentioned it is possible to have more than one public/private key pair per user.

- Each one therefore needs an ID of some kind. The key ID associated with each public key consists of its least significant 64 bits.

- That is, the key ID of public key KU_a is ($\mathrm{KU}_a \bmod 2^{64}$). This is a sufficient length that the probability of duplicate key IDs is very small.

- A key ID is also used for the PGP digital signature as the sender may use one of a number of private keys to encrypt the message digest and the recipient must know which one was used. A more detailed look at the format of a transmitted message is shown in figure 4.

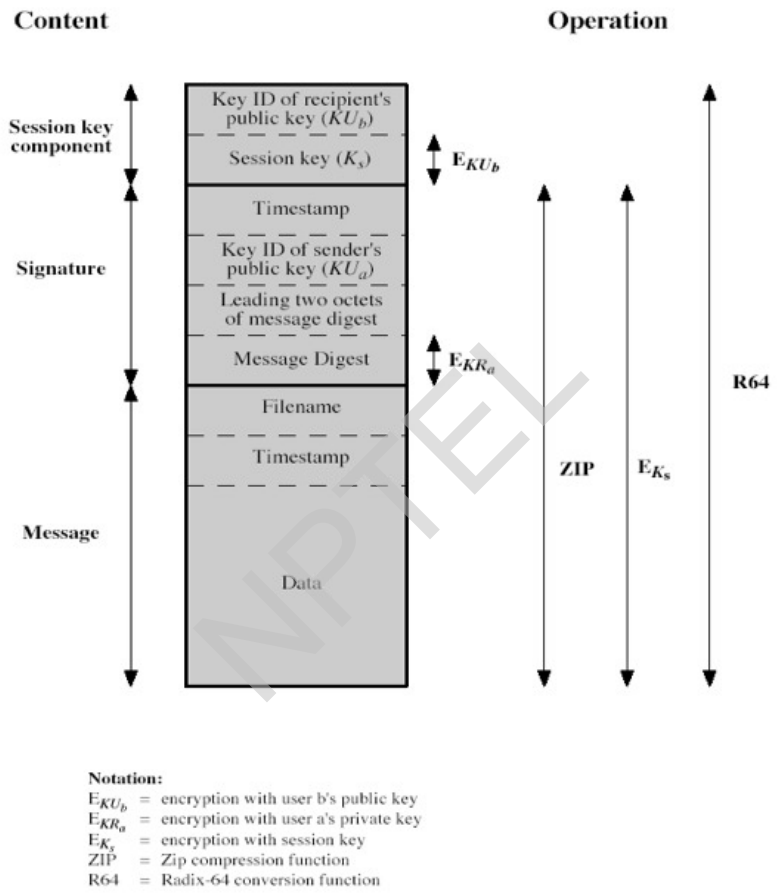


Figure 4: General format of PGP message (from A to B).

Key Rings

- Key IDs are critical to the operation of PGP.
- From figure 4 it can be seen that two key IDs are included in any PGP message that provides both confidentiality and authentication.
- These keys need to be stored and organised in a systematic way for efficient and effective use by all parties.
- The scheme used in PGP is to provide a pair of data structures at each node, one to store the public/private key pairs owned by that node and one to store the public keys of other users known at this node.
- These data structures are referred to, respectively as the private-key ring and the public key ring. They can be seen in figure 5.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	KU_i	trust_flag_i	User i	trust_flag_i		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

Figure 5: General structure of private and public-key rings.

- We can view the ring as a table where each row represents one of the public/private key pairs owned by this user. Each row contains the following:
  - **Timestamp:** The date/time when this key pair was generated.
  - **Key ID:** The least significant 64 bits of the public key for this entry.
  - **Public Key:** The public-key portion of the pair.
  - **Private key:** The private-key portion of the pair.
  - **User ID:** Typically a user's e-mail address.

- The private key ring can be indexed by either User ID, key ID or both.

- However for security the value of the key is not stored in the key ring but and encrypted version of it which requires a pass phrase to decrypt.

- As with all passphrase schemes, the security of this method depends on the strength of the passphrase.

- Figure 6 show PGP message generation (without compression or radix64 conversion) using all the terms we have met (reception is similar).

- A freeware PGP version can be downloaded here:

  http://www.pgpi.org/products/pgp/versions/freeware/.

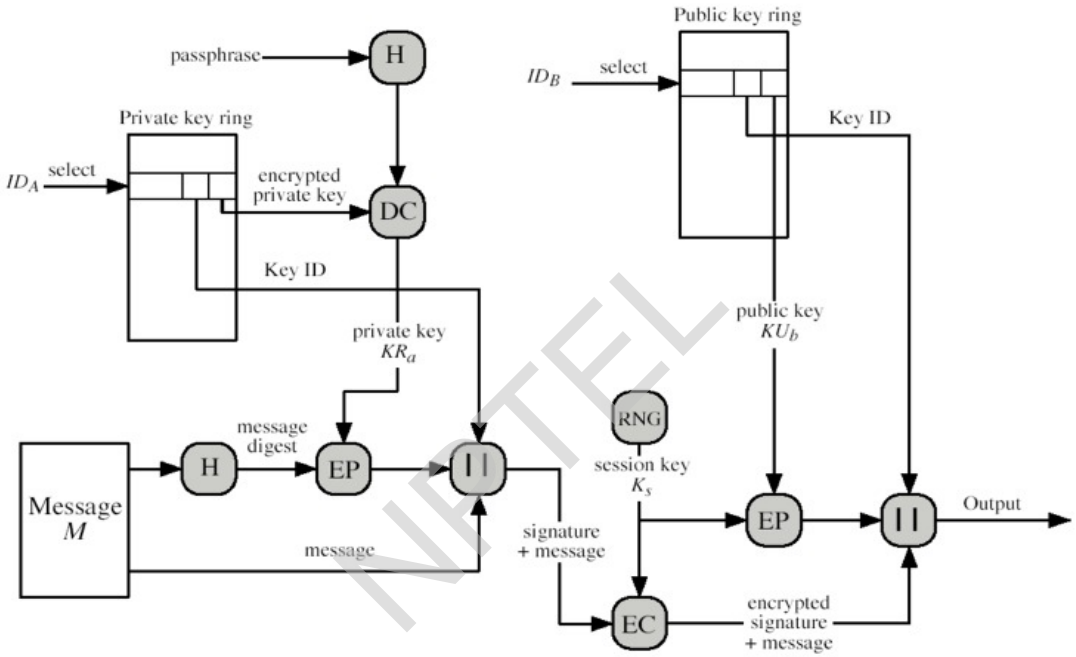


Figure 6: PGP Message generation (from User A to User B; no compression or radix64 conversion).