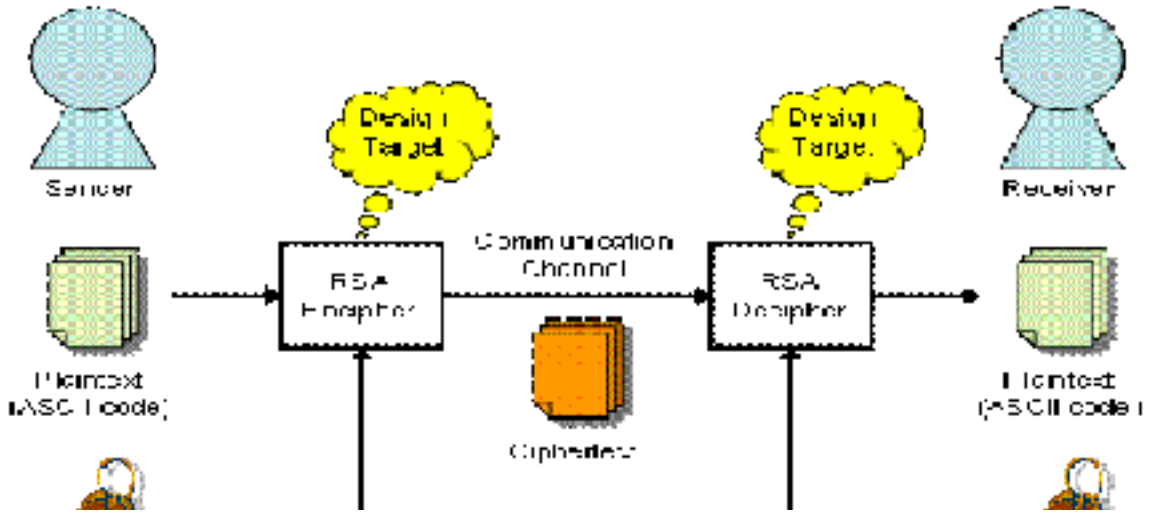
RSA Algorithm in Cryptography

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and Private key is kept private.

An example of asymmetric cryptography :

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.



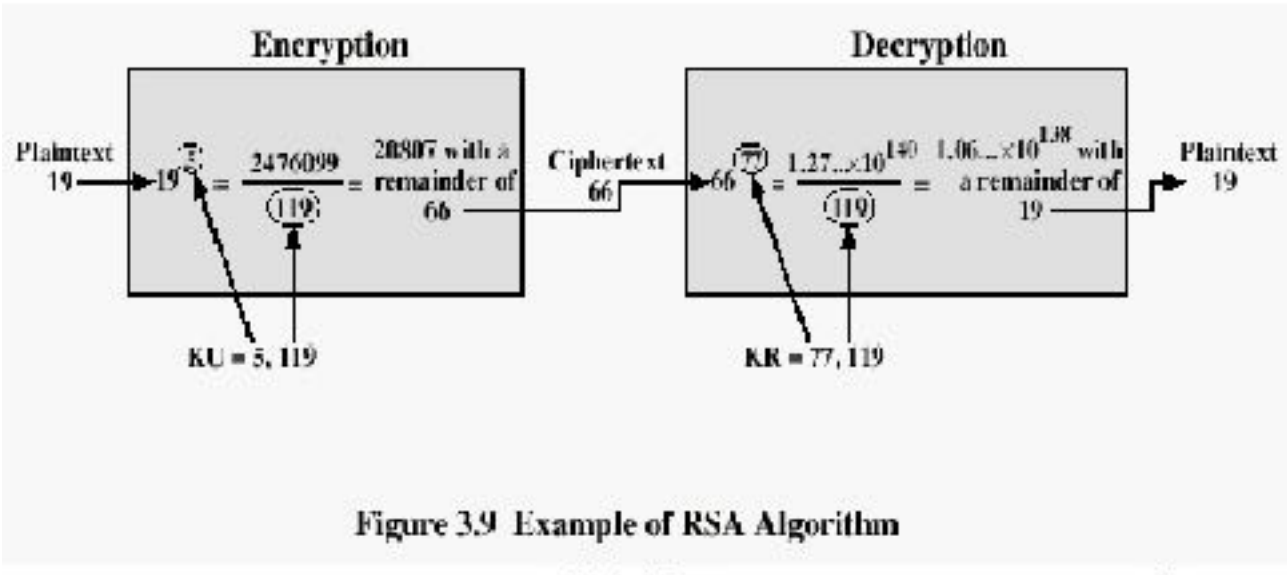


Figure 3.9 Example of RSA Algorithm

Symmetric Key Cryptosystem

Symmetric encryption, also referred to as conventional encryption or single key encryption was the only type of encryption in use prior to the development of public-key encryption in 1976.
The symmetric encryption scheme has five ingredients (see Figure 1):

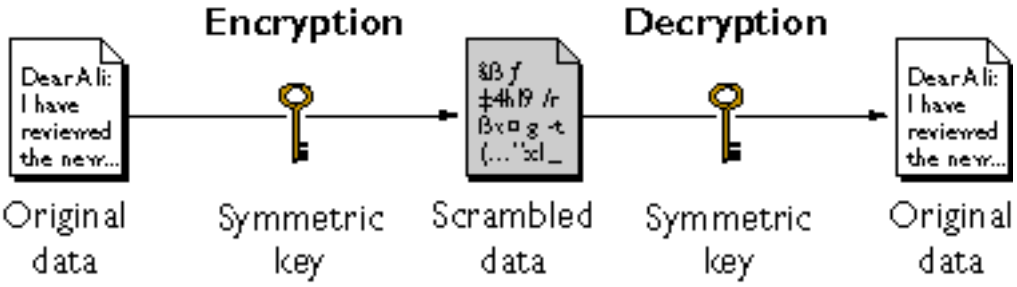
1. Plaintext: This is the original intelligible message or data that is fed to the algorithm as input.

2. Encryption algorithm: The encryption algorithm performs various sub-stitutions and permutations on the plaintext (see the examples of the sub- stitution and permutation ciphers in Lecture 8).

3. Secret Key: The secret key is also input to the encryption algorithm. The exact substitutions and permutations performed depend on the key used, and the algorithm will produce a different output depending on the specific key being used at the time.

4. Ciphertext: This is the scrambled message produced as output. It de-pends on the plaintext and the key. The ciphertext is an apparently random stream of data, as it stands, is unintelligible.

5. Decryption Algorithm: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext."



SHA-256 Cryptographic Hash Algorithm

A hash is not 'encryption' – it cannot be decrypted back to the original text (it is a 'one-way' cryptographic function, and is a fixed size for any size of source text).
This makes it suitable when it is appropriate to compare 'hashed' versions of texts, as opposed to decrypting the text to obtain the original version.
Such applications include hash tables, integrity verification, challenge handshake authentication, digital signatures, etc.

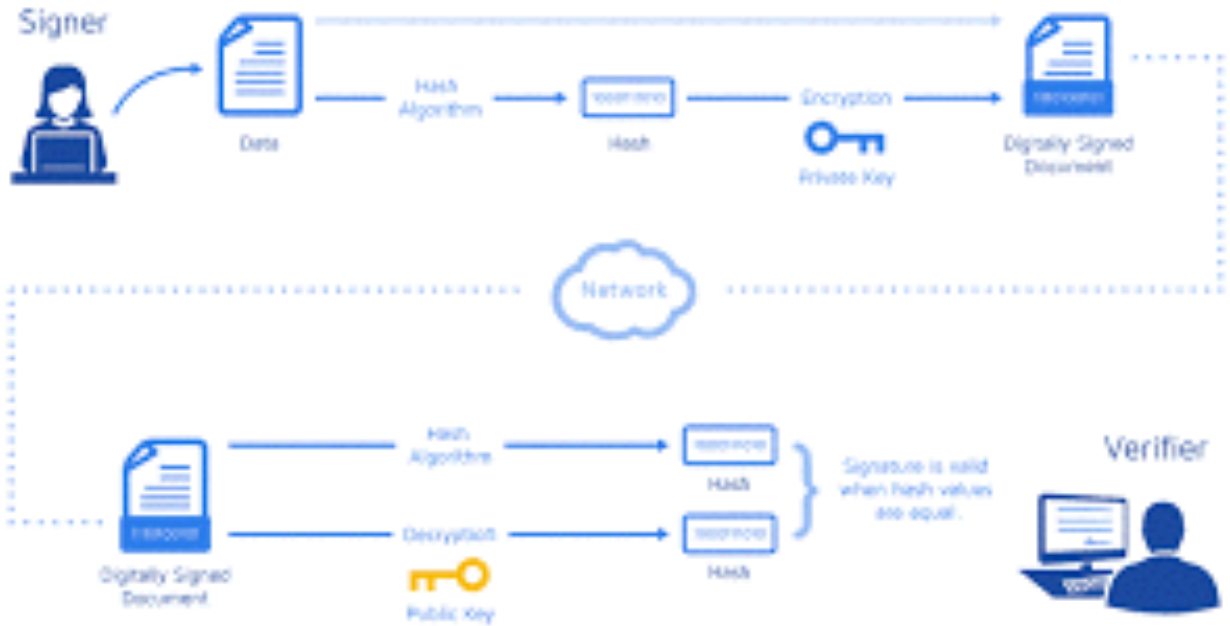
- '*challenge handshake authentication*' (or 'challenge hash authentication') avoids transmissing passwords in 'clear' – a client can send the hash of a password over the internet for validation by a server without risk of the original password being intercepted
- *anti-tamper* – link a hash of a message to the original, and the recipient can re-hash the message and compare it to the supplied hash: if they match, the

message is unchanged; this can also be used to confirm no data-loss in transmission

- *digital signatures* are rather more involved, but in essence, you can sign the hash of a document by encrypting it with your private key, producing a digital signature for the document. Anyone else can then check that you authenticated the text by decrypting the signature with your public key to obtain the original hash again, and comparing it with their hash of the text.

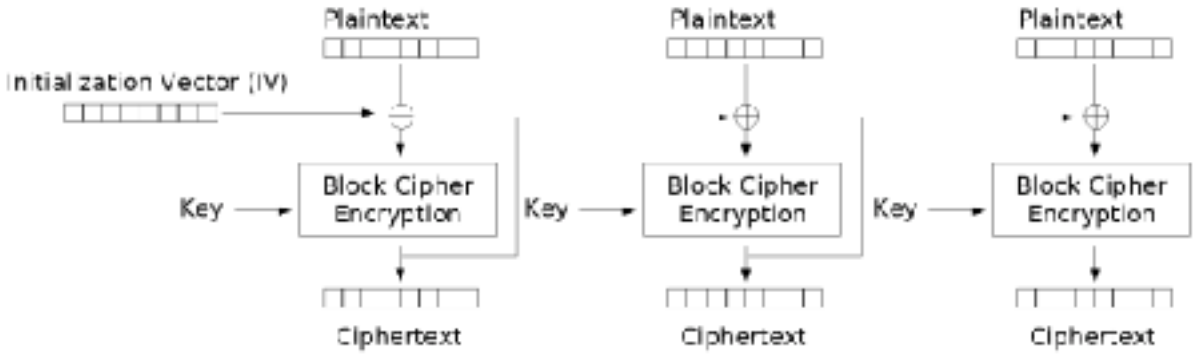
Digital Signature

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message. Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party. Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer. In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.



CBC MODE

CBC or Cipher Block Chaining is a complete other way of connecting blocks together. What is does is instead of just processing each block separately, every block will be XOR'ed with the encrypted previous block. This effectively means that every block depends on the output of the previous block.



Cipher Block Chaining (CBC) mode encryption