

Public Key Encryption Algorithms for Digital Information Exchange

Iqbal Singh

Deptt. of Computer Science, Bhagwant University, Rajasthan, India

Abstract

In this article I have discussed Public Key Encryption, its use in information Encryption, Digital Signature and Key Agreement. This paper discusses some public key algorithms such as RSA, DH, ECDH, DSA, ECDSA, ECC and their explanation. Here I have discussed mathematical arithmetic for these algorithms.

Keywords

Public Key Encryption, Digital Signature, Key Agreement, RSA, ECC, DH, ECDH, DSA, ECDSA, Modular Arithmetic, Cryptography.

I. Introduction

In private key (symmetric key) cryptography, data exchanged can be protected by encryption. In this data is encrypted by encryption algorithms using Keys. Only users having same key can encrypt/decrypt the data [12]. For secured communication between two parties they required to exchange secret key between them. Success of this scheme lies in secrecy of the shared key. Length of the key depends on algorithm used. Due to huge networks interconnecting many intermediate locations, secure key exchange is generally not possible. Public Key Encryption involves pair of public/private keys and cryptographic operations. Public key is distributed online to all users but only the specific user knows the private key. Any unauthorised user who has access to exchanged public information will not be able to calculate shared secret unless that user knows the private key [12]. Public key encryption can be applied in Digital Signatures, Data Encryption and Key Agreement.

II. Mathematical Expression

Cryptographic operations are expressed mathematically. Public and private keys are related by mathematical function called one way function. In public key cryptography the public key is calculated using private key. Obtaining of private key from the public key is a reverse operation. If the private key is obtained from the public key and other public data, then the public key algorithm for the particular key is said to be cracked. The difficulty of reverse operation increases as the key size increases. So public key algorithms operate on sufficiently large numbers to make the reverse operation practically impossible and thus make the system secure.

III. Key Agreement

Key agreement is a method in which the devices/users communicating in the network establishes a shared secret between them by exchanging their public keys. Both the devices on receiving the other device's public key perform key generation operation using its private key to obtain the shared secret [15]. The public keys are generated using private key and other shared constants. Suppose P be the private key of a device and $U(P, C)$ be the public key. Since public key is generated using private key, the representation $U(P, C)$ shows that the public key contain the components of private key P and some constants C where C is known by all the device taking part in the communication. Consider two devices A and B. Let P_A and $U_A(P_A, C)$ be the private

key and public key of device A, and P_B and $U_B(P_B, C)$ be the private key and public key of device B respectively. Both device exchanges their public keys. Device A, having got the public key of B, uses its private key to calculate shared secret

$$K_A = \text{Generate_Key}(P_A, U_B(P_B, C))$$

Device B, having got the public key of A, uses its private key to calculate the shared secret

$$K_B = \text{Generate_Key}(P_B, U_A(P_A, C))$$

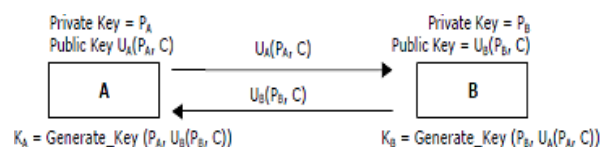


Fig. 1: Calculating shared secret by A and B

The key generation algorithm 'Generate_Key' will be such that the generated keys at the device A and B will be the same, that is shared secret $K_A = K_B = K(P_A, P_B, C)$. Since it is practically impossible to obtain private key from the public key. Any third party, having access only to the public keys $U_A(P_A, C)$ and $U_B(P_B, C)$, will never be able to obtain the shared secret K .

IV. Data Encryption

Encryption is a process in which the sender encrypts the message in such a way that only the intended recipient will be able to decrypt the message. Consider an example in which a device B whose private key and public key are P_B and U_B respectively. Since U_B is public key all devices will be able to get it. For any device that needs to send the message 'Msg' in a secured way to device B, it will encrypt the data using B's public key to obtain the cipher text 'Ctx'. The encrypted message, cipher text, can only be decrypted using B's private key. On receiving the message, the B decrypts it using its private key P_B . Since only B knows its private key P_B none other including A can decrypt the message [16].

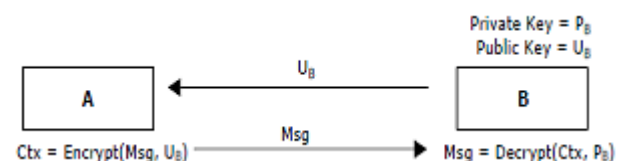


Fig. 2: Encrypted data send by A and its decryption by B

It is important that device A receives the correct public key from device B, i.e. no middleman must change the public key. Digital Certificate helps to deliver the public key in authenticated method.

V. Digital Signature

In this technique, message can be signed by a device using its private key to ensure authenticity of the message. Any device that has got the access to the public key of the signed device can verify the signature. Thus the device receiving the message can ensure that the message is indeed signed by the intended device and is not modified during the transit. If any the data or signature is modified, the signature verification fails [13]. Take a case in

which device A need to ensure the authenticity of its message, the device A signs its message using its private key P_A . The device A will then send the message 'Msg' and signature 'Sgn' to device B. The device B, on receiving the message, can verify the message using A's public key U_A and there by ensuring that the message is indeed sent by A and is also not altered during the transit. Since only the device A knows its private P_A key, it is impossible for any other device to copy the signature.

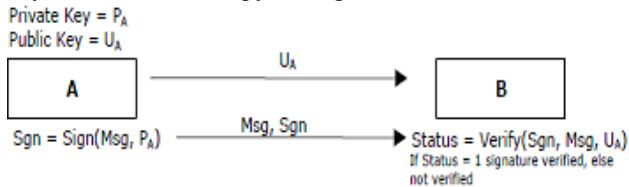


Fig. 3: Message exchange through Digital Signature

A. Digital Certificate

Consider two devices X and Y establishing a shared secret. Both devices exchange their public keys. The devices calculate the shared secret using their private key and the other device's public key. Now consider an intermediate point M through which all the communication happens. If M captures Y's public key and sends M's public key instead with Y's identity, then X will end up in establishing shared secret with M and will communicate with M thinking that it is communicating with Y. This happened because there is no way for X to verify that the received public key is indeed that of Y. Here is now the Digital Certificate comes to play. For data transfer in a network there is an authority trusted by all devices. This Trusted Certificate Authority (CA) [14] signs the public keys and the unique identifiers of all devices. These signed data (public key, IDs etc.) along with the signature arranged in a standard format is called as the certificate. All the devices that take part in secured and trusted communication have to obtain a certificate from the trusted authority.

Now the device X and Y exchanges their respective certificate instead of public key. These certificates are verified using CA's public key. Even if the intermediate point M modifies the public key or any other data in any of the certificate, the certificate verification will fail. Some problem still exists. How to get the public key of the CA in a realistic way? The public key of the CA needs to be obtained by some other trusted method. For example, in cases of secure Internet browsing the certificate of CA installed in the device along with the web browser.

The device that requires a certificate will send the certificate request to the CA. The request contains the device data such as device ID and device public key. The CA first finds the digest of the device data and CA specific data using a hash algorithm. CA then signs the hash, using its private key and combines the data and signature in a standard format to form a certificate and is given to the device. The CA usually does some background check to ensure the device is not hostile before issuing the certificate. An example of a standard digital certificate format is X.509 certificates [7].

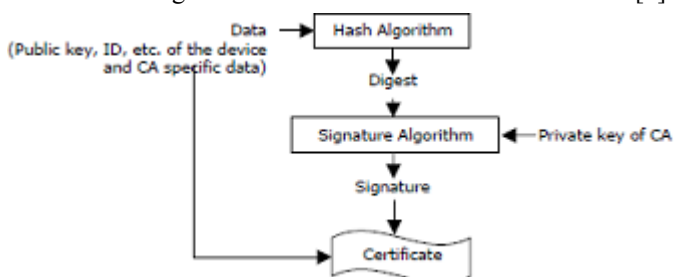


Fig. 4: Formation of Certificate

On receiving a certificate, a device extracts the data from the certificate, checks the ID and other data in the certificate. The signature in the certificate is verified using CA's public key.

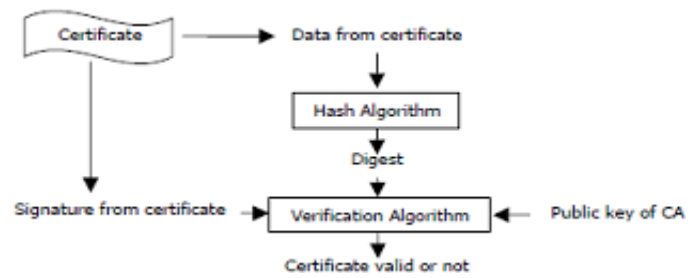


Fig. 5: Verification procedure of certificate

B. Certificate Hierarchy

Due to distributed location of the devices that take part in communication, a central certificate authority may not suffice to issue and maintain the certificate of all the devices. Certificate hierarchy is the solution in this case. Trusted root CA will give permission to intermediate CAs to give certificate to the communicating devices by issuing the certificate to these intermediate CAs. These intermediate CAs then issue certificates to the device. There can be multiple levels of certificate hierarchy in which the intermediate CAs will give permission to other CA to issue certificate to the communicating devices.

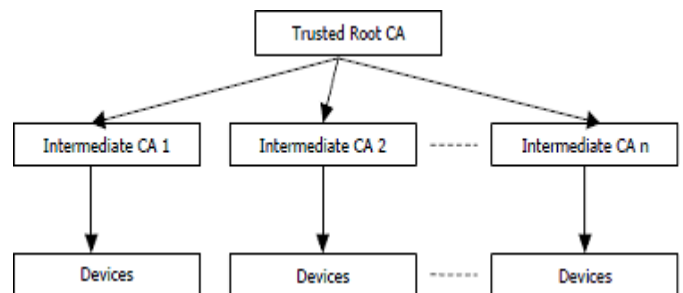


Fig. 5: Certificate Hierarchy

If a device X obtained a certificate from an intermediate CA, then it has along with its device certificate, the certificate of the intermediate CA, which issued the certificate to the device X. Such device will have certificates of all intermediate CAs up to the root. Consider another device Y taking part in communication with device X. Device Y on receiving the X's certificate may request X the certificate of intermediate CA who issued certificate to X. The device Y may end up in asking all the certificate of intermediate CAs till the root CA for successful verification of X's device certificate. The device Y must at least has the self-signed root CA certificate obtained by a trusted means to successfully authenticate X's certificate.

C. Protection of Public Keys

The public keys are generally stored as certificates and the root CA public key is stored as self signed certificate. It is important that the public keys/certificates should be stored securely such that any third party cannot modify it. If Root CA certificate is modified then the attacker can make any certificate acceptable by the device thus defeating the certificate and secured communication.

D. Authentication

In one way authentication, only one party needs to provide its certificate to prove its authenticity and hence to create secured

channel. Once the secured channel is created the other party authenticates/authorize itself by providing a 'user name and password'. This type of authentication is generally used in client-server architecture. In this case the server maintains the database of all the authorized users. Only if the username and password are matched, the server gives out the secured information. In two way authentication both the devices need to have a Digital Certificate from a trusted authority. Both devices exchange their respective certificates for authentication during handshake of key agreement protocol. This happens typically in peer-peer communication.

VI. Modular Arithmetic

Modular arithmetic is the commonly used arithmetic in public key cryptography. Modular arithmetic deals only with integers. Modular arithmetic over a number n involves arithmetic operations on integers between 0 and $n-1$, where n is called the modulus. If the number happens to be out of this range in any of the operation the result, r , is wrapped around in to the range 0 and $n-1$ by repeated subtraction of the modulus n from the result r . This is equivalent in taking the remainder of division operation r/n .

For e.g. for modulo 23 arithmetic

$n=23$, Let $a=15$, $b=20$

$(a+b) \bmod n = (15+20) \bmod 23 = 35 \bmod 23 = 12$

Since the result of $a+b=35$ which is out of the range $[0,22]$, the result is wrapped around in to the range $[0, 22]$ by subtracting 23 till the result is in range $[0, 22]$.

$a \bmod b$ is thus explained as remainder of division a/b . Subtraction and multiplication can also be explained similarly. A negative number is added repeatedly with n till it can be represented in the range $[0, n-1]$

The modular division $a/b \bmod p$ is defined as $a*b^{-1} \bmod p$. b^{-1} is the multiplicative inverse of b .

Multiplicative inverse of number b with respect to $\bmod p$ is defined as a number b^{-1} such that $b*b^{-1} \bmod p = 1$.

A. Congruent Relation:

Modular arithmetic is a congruent relation. Congruence is shown by the symbol ' \equiv '. For a modulus n two numbers a and b are said to be congruent if:

$a \bmod n = b \bmod n$ i.e.

$a \equiv b \pmod{n}$ if, $a \bmod n = b \bmod n$

For example consider the modulus 7 i.e. $n = 7$

Then the numbers 2, 9, 16, 23 etc are congruent to each other, since

$(2 \bmod 7) = (9 \bmod 7) = (16 \bmod 7) = (23 \bmod 7)$ etc

B. Properties of Modular Arithmetic

P1. $a \equiv b \bmod n$ implies $a-b=k*n$, where k is an integer

P2. $\bmod n + \bmod n \equiv a+b \pmod{n}$, also true for other operators '-', '*' and '/'

P3. $a+b \equiv b+a \pmod{n}$, also true for other operators '-', '*' and '/'

P4. $a \equiv a \bmod n$

P5. $a \equiv b \bmod n$ implies $b \equiv a \bmod n$

P6. Fermat's little theorem, if M and p are coprime then $M^{p-1} \equiv 1 \pmod{p}$

P7. if p and q are coprime and also if $a \equiv b \pmod{p}$ and $a \equiv b \pmod{q}$ then $a \equiv b \pmod{pq}$

VII. Algorithms

This section discusses a few public key algorithms and will also gives an explanation on how these algorithms work. The algorithms covered in this section are:

- Key Agreement Algorithms – RSA, DH, ECDH

- Encryption Algorithms – RSA
- Signature Algorithms – RSA, DSA, ECDSA

A. Rivest, Shamir, Adleman Public-Key Encryption - RSA

RSA is a public key algorithm that is used for Encryption, Signature and Key Agreement. RSA typically uses keys of size 1024 to 2048. The RSA standard is specified RFC 3447, RSA Cryptography Specifications Version 2.1[3]. Overviews of RSA algorithms are given below.

1. RSA Encryption

(i) Parameter Generation

- R1. Select two prime numbers p and q .
- R2. Find $n=p*q$, Where n is the modulus that is made public. The length of n is considered as the RSA key length.
- R3. Choose a random number ' e ' as a public key in the range $0 < e < (p-1)(q-1)$ such that $\gcd(e, (p-1)(q-1)) = 1$.
- R4. Find private key d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$.

(ii) Encryption

Consider the device A that needs to send a message to B securely.

- R5. Let e be B's public key. Since e is public, A has access to e .
- R6. To encrypt the message M , represent the message as an integer in the range $0 < M < n$.
- R7. Cipher text $C = M^e \bmod n$, where n is the modulus.

(iii) Decryption

- R8. Let C be the cipher text received from A.
- R9. Calculate Message $M = C^d \bmod n$, where d is B's private key and n is the modulus.

2. RSA Key Agreement

Since public key cryptography involves mathematical operation on large numbers, these algorithms are considerably slow compared to the symmetric key algorithm. They are so slow that it is infeasible to encrypt large amount of data. Public key encryption algorithm such as RSA can be used to encrypt small data such as 'keys' used in private key algorithm. RSA is thus used as key agreement algorithm.

(i) Key Agreement Algorithm

For establishing shared secret between two device A and B

- R10. Generate a random number, key, at device A.
- R11. Encrypt key by RSA encryption algorithm using B's public key and pass the cipher text to B
- R12. At B decrypt the cipher text using B's private key to obtain the key.

3. RSA Signature

RSA Signature is similar to RSA encryption except that the private key is used for signing and public key is used for verification.

(i) Parameter Generation

The parameter generation process is same as that in RSA Encryption.

(ii) Signing

Consider the device A that needs to sign the data that it sends to B.

- R13. Let d be A's private key
- R14. To sign a data M , represent the data as an integer in

the range $0 < M < n$

R15. Signature $C = M^d \bmod n$

(iii) Verification

R16. Let M be the message and C be the signature received from A

R17. Calculate $M' = C^e \bmod n$, where e is A 's public key. Since e is public, B has access to e

R18. If $M' = M$, the signature is verified, else failed.

(iv) One-Way Function in RSA

Consider the key generation equation R4, $ed \equiv 1 \pmod{(p-1)(q-1)}$ and $n = p \cdot q$

Where e is the public key d is the private key. p and q are kept private but n is made public. Since e is public, anybody who has access to p and q could easily generate the private key d using the above equation R4. The security of RSA depends on the difficulty to factorize n to obtain the prime numbers p and q . n is easily obtained by multiplying p and q but the reverse operation of factorizing n to obtain prime numbers p and q is practically impossible if p and q are sufficiently large numbers.

(v) Mathematical Explanation for RSA

From parameter generation equation R4

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

From the encryption equation R7

$$\text{Cipher text } C = M^e \bmod n$$

From the decryption equation R9

$$\text{Message } M = C^d \bmod n$$

Combining above two equations $M = (M^e \bmod n)^d \bmod n$, using equation HX1

$$M = M^{ed} \bmod n$$

Similarly by combining signature and verification equation R15 and R17 we get

$$M = M^{ed} \bmod n$$

So to prove the correctness of RSA, it has to prove that

$$M = M^{ed} \bmod n, \text{ if}$$

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

From the above equation $ed \equiv 1 \pmod{(p-1)(q-1)}$ and property P1 it follows that

$$ed - 1 = K(p-1)(q-1), \text{ which can also be written as}$$

$$ed - 1 = k(p-1), \text{ and } \text{----} [\text{RX1}]$$

$$ed - 1 = k'(q-1) \text{ ----} [\text{RX2}]$$

Where K , k and k' are positive integers

Since any integer is congruent to itself it can be written as

$$M^{ed} \equiv M^{ed} \pmod{p}, \text{ i.e.}$$

$$M^{ed} \equiv M^{ed-1} * M \pmod{p},$$

Using equation RX1 the above equation can be written as

$$M^{ed} \equiv M^{k(p-1)} * M \pmod{p}, \text{ ----} [\text{RX3}]$$

Since p is prime, any integer M can either be a co-prime with p or a multiple of p .

Case 1: If M and p are coprime, then from Fermat's little theorem

$$M^{p-1} \equiv 1 \pmod{p}, \text{ or}$$

$$M^{k(p-1)} \equiv 1^k \pmod{p}, \text{ i.e.}$$

$$M^{k(p-1)} \equiv 1 \pmod{p} \text{ ----} [\text{RX4}]$$

From equations RX3 and RX4

$$M^{ed} \equiv M \pmod{p}$$

Case 2: If M is a multiple of p , then M^{ed} will also be a multiple of p , i.e.

$M \bmod p = 0$, also $M^{ed} \bmod p = 0$, thus from congruence relation,

$$M^{ed} \equiv M \pmod{p}$$

Similarly using RX2 it can be proved that $M^{ed} \equiv M \pmod{p}$ for above two cases.

Since p and q are prime numbers they are coprime to each other. Therefore by using property P7 the above two equations can be combined as

$$M^{ed} \equiv M \pmod{p \cdot q}, \text{ by property P5}$$

$$M \equiv M^{ed} \pmod{p \cdot q}$$

Since M is chosen in the range 0 and $(p \cdot q - 1)$

$$M = M^{ed} \bmod p \cdot q, \text{ i.e. } M = M^{ed} \bmod n$$

B. Diffie-Hellman Key Agreement - DH

Diffie-Hellman is a key agreement algorithm that helps two devices to agree on a shared secret between them without the need to exchange any secret/private information. The DH standard is specified RFC 2631[4]. An overview of the algorithm is given below.

1. Key Agreement Algorithm

For establishing shared secret between two devices A and B , both devices agree on public constants p and g . Where p is a prime number and g is the generator less than p .

D1. Let a and b be the private keys of the devices A and B respectively, Private keys are random number less than p .

D2. Let $g^a \bmod p$ and $g^b \bmod p$ be the public keys of devices A and B respectively

D3. A and B exchanged their public keys.

D4. The end A computes $(g^b \bmod p)^a \bmod p$ that is equal to $g^{ba} \bmod p$.

D5. The end B computes $(g^a \bmod p)^b \bmod p$ that is equal to $g^{ab} \bmod p$.

D6. Since $K = g^{ba} \bmod p = g^{ab} \bmod p$, shared secret = K .

(i) Mathematical Explanation

From the properties of modular arithmetic P2

$$a \bmod n * b \bmod n = a * b \pmod{n}$$

Which can be written as

$$(a_1 \bmod n) * (a_2 \bmod n) * \dots * (a_k \bmod n) = a_1 * a_2 * \dots * a_k \pmod{n},$$

if $a_i = a$, where $i = 1, 2, 3, \dots, k$

$$(a \bmod n)^k = a^k \bmod n, \text{ therefore } \text{----} [\text{HX1}]$$

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p \text{ and}$$

$$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$$

For all integers $g^{ab} = g^{ba}$

Therefore shared secret $K = g^{ab} \bmod p = g^{ba} \bmod p$

Since it is practically impossible to find the private key a or b from the public key $g^a \bmod p$ or $g^b \bmod p$, it is not possible to obtain the shared secret K for a third party.

(ii) One-Way Function

For device A , Let a be the private key and $x = g^a \bmod p$ is the public key,

Here $x = g^a \bmod p$ is one-way function. The public key x is obtained easily in the forward operation, but finding ' a ' given x , g and p is the reverse operation and takes exponentially longer time and is practically impossible. This is known as discrete logarithm problem [11].

(iii) Elliptic Curve Diffie Hellman – ECDH

ECDH, a variant of DH, is a key agreement algorithm. For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters [17].

An overview of ECDH is given below.

(iv) Key Agreement Algorithm

For establishing shared secret between two device A and B

- E1. Let d_A and d_B be the private key of device A and B respectively, Private keys are random number less than n , where n is a domain parameter.
- E2. Let $Q_A = d_A * G$ and $Q_B = d_B * G$ be the public key of device A and B respectively, G is a domain parameter
- E3. A and B exchanged their public keys
- E4. The end A computes $K = (x_K, y_K) = d_A * Q_B$
- E5. The end B computes $L = (x_L, y_L) = d_B * Q_A$
- E6. Since $K=L$, shared secret is chosen as x_K

(v) Mathematical Explanation

To prove the agreed shared secret K and L at both devices A and B are the same From E2, E4 and E5

$$K = d_A * Q_B = d_A * (d_B * G) = (d_B * d_A) * G = d_B * (d_A * G) = d_B * Q_A = L$$

Hence $K = L$, therefore $x_K = x_L$

Since it is practically impossible to find the private key d_A or d_B from the public key Q_A or Q_B , it's not possible to obtain the shared secret for a third party.

D. Elliptic Curve Digital Signature Algorithm - ECDSA

For sending a signed message from A to B, both have to agree up on Elliptic Curve domain parameters. Sender A have a key pair consisting of a private key d_A (a randomly selected integer less than n , where n is the order of the curve, an elliptic curve domain parameter) and a public key $Q_A = d_A * G$ (G is the generator point, an elliptic curve domain parameter)[18]. An overview of ECDSA process is defined below.

Signing

Consider the device A that signs the data M that it sends to B.

- E7. Let d_A be A's private key
- E8. Calculate $m = \text{HASH}(M)$, where HASH is a hash function, such as SHA-1
- E9. Select a random integer k such that $0 < k < n$
- E10. Calculate $r = x_1 \bmod n$, where $(x_1, y_1) = k * G$
- E11. Calculate $s = k^{-1}(m + d_A * r) \bmod n$
- E12. The signature is the pair (r, s)

Verification

E13. Let M be the message and (r, s) be the signature received from A

E14. Let Q_A be A's public key. Since Q_A is public, B has access to it.

- E15. Calculate $m = \text{HASH}(M)$
- E16. Calculate $w = s^{-1} \bmod n$
- E17. Calculate $u_1 = m * w \bmod n$ and $u_2 = r * w \bmod n$
- E18. Calculate $(x_1, y_1) = u_1 * G + u_2 * Q_A$
- E19. The signature is valid if $x_1 = r \bmod n$, invalid otherwise

Mathematical Explanation

From the verification equation E19, the signature is valid if $x_1 = r \bmod n$ ---- [EX1]

But from E18, x_1 is the x-coordinate of equation $u_1 * G + u_2 * Q_A$
From E10 r is the x-coordinate of equation $k * G$

Thus to prove equation EX1, It has to prove that

$$u_1 * G + u_2 * Q_A = k * G$$

Substituting the value of u_1 and u_2 from E17 the first part of the above equation

$$u_1 * G + u_2 * Q_A = (m * w \bmod n) * G + (r * w \bmod n) * Q_A$$

But $Q_A = d_A * G$, therefore

$$u_1 * G + u_2 * Q_A = (m * w \bmod n) * G + (r * w * d_A \bmod n) * G, \text{ i.e.}$$

$$u_1 * G + u_2 * Q_A = (w * (m + r * d_A) \bmod n) * G$$

But from E16, $w = s^{-1} \bmod n$, i.e. $s^{-1} \equiv w \bmod n$ therefore

$$u_1 * G + u_2 * Q_A = (s^{-1} * (m + r * d_A) \bmod n) * G \text{ ---- [EX2]}$$

but from equation E11

$$s = K^{-1}(m + d_A * r) \bmod n, \text{ i.e. } k \equiv s^{-1}(m + d_A * r) \bmod n$$

Substituting k in EX2

$$u_1 * G + u_2 * Q_A = k * G$$

Therefore $x_1 = r \bmod n$

E. Digital Signature Algorithm - DSA

DSA is a public key algorithm that is used for Digital Signature. The DSA standard is specified FIPS 186-2, Digital Signature Standard [2]. An overview of the algorithm is given below.

(i) Parameter Generation

- S1. Choose a 160-bit prime q .
- S2. For an integer z , choose an L -bit prime p , such that $p = qz + 1$, $512 \leq L \leq 1024$, and L is divisible by 64.
- S3. Choose h , where $1 < h < p-1$ such that $g = hz \bmod p > 1$.
- S4. Choose a random number x , where $0 < x < q$.
- S5. Calculate $y = g^x \bmod p$.
- S6. Public key is (p, q, g, y) . Private key is x .

(ii) Signing

Consider the device A that sign the data M that it sends to B.

- S7. Let x be A's private key and (p, q, g, y) be A's public key.
- S8. Generate a random per-message value k , where $0 < k < q$.
- S9. Calculate $r = (g^k \bmod p) \bmod q$.
- S10. Calculate $s = (K^{-1}(M + x * r)) \bmod q$, where M is the hash SHA1 of the message
- S11. The signature is (r, s) .

(iii) Verification

S12. Let M be the message and (r, s) be the signature received from A

S13. Let (p, q, g, y) be A's public key. Since (p, q, g, y) is public, B has access to it.

S14. Calculate $w = s^{-1} \bmod q$.

S15. Calculate $u_1 = (M * w) \bmod q$, where M is the hash SHA1 of the message.

S16. Calculate $u_2 = (r * w) \bmod q$.

S17. Calculate $v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$.

S18. The signature is valid if $v = r$, invalid otherwise.

(iv) Mathematical Explanation

From S18, Signature is valid if $v = r$, to prove the correctness of the algorithm it has to prove that $v = r$, if signature is valid.

Form S17, $v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$.

But from S5, $y = g^x \bmod p$, i.e. $y \equiv g^x \bmod p$, i.e. using equation HX1, $y^{u_2} \equiv g^{x * u_2} \bmod p$

Therefore

$$v = ((g^{u_1} * g^{x * u_2}) \bmod p) \bmod q. \text{ ---- [DX1]}$$

$$\text{But from S16, } g^{x * u_2} = g^{x * (r * w \bmod q)} = g^{(x \bmod q) * (r * w \bmod q)}$$

$$\text{From P2 } (x \bmod q) * (r * w \bmod q) \equiv (x * r * w) \bmod q$$

$$\text{Using P1 } (x \bmod q) * (r * w \bmod q) = (x * r * w) \bmod q + k * q$$

Where k is an integer, Therefore

$$g^{x.u2} = g^{(x^*r^*w) \bmod q + k^*q} \text{ i.e.}$$

$$g^{x.u2} = (g^{(x^*r^*w) \bmod q}) * (g^{k^*q}) \text{ ---- [DX2]}$$

From S3, $g = h^z \bmod p$,

Using HX1 and S2, $g^a \equiv h^{az} \equiv h^{p-1} \pmod{p}$

By P6, Fermat's little theorem $h^{p-1} \equiv 1 \pmod{p}$, i.e. $g^a \equiv 1 \pmod{p}$

Substituting the value of $g^a \equiv 1 \pmod{p}$ in DX2

$$g^{x.u2} = (g^{(x^*r^*w) \bmod q}) \text{ ---- [DX3]}$$

But from S15, $g^{u1} = g^{(M^*w \bmod q)} \text{ ---- [DX4]}$

Substituting DX3 and DX4 in DX1

$$v = ((g^{wM \bmod q + xrw \bmod q}) \bmod p) \bmod q \text{ i.e.}$$

$$v = ((g^{w(M+xr) \bmod q}) \bmod p) \bmod q \text{ ---- [DX5]}$$

From S10, in signature algorithm

$$s = (k^{-1}(M+x^*r)) \bmod q \text{ i.e.}$$

$$k \equiv s^{-1}(M+x^*w) \bmod q \text{ ---- [DX6]}$$

But from S14, $w = s^{-1} \bmod q$ i.e.

$$s^{-1} \equiv w \bmod q$$

Therefore equation DX6 can be written as

$$k \equiv w(M+x^*w) \bmod q$$

Since $k < q$, $k = w(M+x^*w) \bmod q \text{ ---- [DX7]}$

Combining DX5 and DX7 and using S9

$$V = ((g^k) \bmod p) \bmod q = r \text{ i.e. } v = r$$

(v) One-Way function in DSA

Consider the equation S5, $y = g^x \bmod p$, where x is the private key but y , g and p are public. Calculating y from g , x and p is a forward operation but obtaining x from the given y , g and p is the reverse operation and hence finding x is impossible for large numbers. This is known as discrete logarithm problem [11].

F. Elliptic Curve Cryptography – ECC

Elliptic curve cryptography (ECC) is relatively new technology compared to other public key cryptography such as RSA. Elliptic key operates on smaller key size. A 160-bit key in ECC is considered to be as secured as a 1024 bit key in RSA. ECC operates on the points in the elliptic curve $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$. The above equation of elliptic curve is in real coordinate. To make elliptic curve operation efficient and accurate the elliptic curve can be defined in finite fields. Elliptic curve in two finite fields, prime field and binary field, are defined by standard. In prime field operation the elliptic curve equation is modified as $y^2 \bmod p = x^3 + ax + b \bmod p$, where $4a^3 + 27b^2 \bmod p \neq 0$. The ECC standards are specified in SEC, Standards for Efficient Cryptography. [5]

1. Domain Parameters

There are certain public constants that are shared between parties involved in secured and trusted ECC communication. This includes curve parameter a , b , a generator point G in the chosen curve, the modulus p , order of the curve n and the cofactor h . There are several standard domain parameters defined by SEC, Standards for Efficient Cryptography [6].

2. Point Multiplication

Point multiplication is the central operation in ECC. In point multiplication a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve.

$$\text{i.e. } k^*P = Q$$

Point multiplication is achieved by two basic elliptic curve operations

- Point addition: adding two points J and K using elliptic curve equation to obtain another point L i.e., $L = J + K$.
- Point doubling: adding a point J to itself using elliptic curve equation to obtain another point L i.e. $L = 2J$.

Here is a simple example of point multiplication.

Let P be a point on an elliptic curve. Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve. i.e. to find $Q = k^*P$.

If $k = 23$ then $k^*P = 23^*P = 2(2(2(2P) + P) + P) + P$.

In the ECC explanations given below upper case letter indicates a point in the elliptic curve and the lower case letter indicates a scalar

3. One Way Function in ECC

The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Let P and Q be two points on an elliptic curve such that $k^*P = Q$, where k is a scalar. Q can be easily obtained from P and k but given P and Q , it is computationally infeasible to obtain k , if k is sufficiently large. k is the discrete logarithm of Q to the base P .

VIII. Conclusion

Public key cryptography is an innovation and is an unavoidable part of almost all security protocol and application. Being able to negotiate a shared secret between two devices online without the need of any exchange of secret data created a breakthrough in secure network/internet communication. Though theoretically it is possible to find the shared secret from the available public information, it will take exponentially longer time making it practically impossible. It is the belief in age-old mathematics, that finding an easy method for reverse process of one-way function is unlikely, keeps the public key cryptography going.

References

- [1] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, 1996, "Handbook of Applied Cryptography", CRC Press.
- [2] FIPS PUB 186-2, (2000), "Digital Signature Standard (DSS)", January 2000, [Online] Available: http://csrc.nist.gov/publications/fips/fips186_2/fips186-2-change1.pdf
- [3] RSA Laboratories, (2002), "PKCS#1 v2.1: RSA Cryptography Standard", [Online] Available: <http://www.rsa.com/rsalabs/node.asp?id=2125>
- [4] RFC 2631 (1999), "Diffie-Hellman Key Agreement Method", [Online] Available : <http://tools.ietf.org/html/rfc2631>
- [5] Certicom (2000), "Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography", Version 1.0, [Online] Available : http://www.secg.org/download/aid-385/sec1_final.pdf
- [6] Certicom (2000), "Standards for Efficient Cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters", Version 1.0, [Online] Available : http://www.secg.org/download/aid-386/sec2_final.pdf
- [7] ITU (2008), "Recommendation X.509", [Online] Available: <http://www.itu.int/rec/T-REC-X.509-200508-I>
- [8] Anoop MS (2007), "Elliptic Curve Cryptography - An Implementation Guide", [Online] Available : <http://hosteddocs.ittoolbox.com/AN1.5.07.pdf>
- [9] Openssl (2009), "Cryptography and Open Source Toolkit for SSL/TLS" [Online] Available : <http://www.openssl.org>
- [10] Certicom (2010), "Elliptic Curve Cryptography" [Online] Available : http://www.certicom.com/index.php?action=ecc_tutorial,home
- [11] RSA Laboratories (2011), "2.3.7 What is the discrete logarithm problem?" [online]. Available: <http://www.rsa.com/rsalabs/node.asp?id=2193>
- [12] Jonathan Katz, Yehuda Lindell (2008), "Introduction to Modern Cryptography", CHAPMAN & HALL/CRC,

- [13] US ESIGN Act of 2000, ELECTRONIC SIGNATURES IN GLOBAL AND NATIONAL COMMERCE ACT, [online]. Available: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=106_cong_public_laws&docid=f:publ229.106.pdf, Last accessed on june 28, 2011
- [14] Ran Canetti (2004), "Universally Composable Signature, Certification, and Authentication". CSFW, [Online] Available: <http://eprint.iacr.org/2003/239>
- [15] A. Shamir (1979), "How to share a secret." In Communications of the ACM, volume 22, pp. 612–613, ACM,
- [16] V.V.Yashchenko (2002). "Cryptography: an introduction". AMS Bookstore. p.6. ISBN 0-8218-2986-6
- [17] NIST (2006), Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", [Online] Available : http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf last accessed on 25 june, 2010
- [18] Certicom Research (2009), "Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography", Version 2.0. [Online] Available : <http://www.secg.org/download/aid-780/sec1-v2.pdf>



Iqbal Singh is B.Com, MCA. He is pursuing his Ph.D. in Computer Science from department of Computer Science, Bhagwant University, Rajasthan, India. Worked as software developer for 5 years in .NET technologies and as Assistant Professor of computer Science at G.G.S. Khalsa College for Women, Jhar Sahib (Panjab).