User Manual for Mobile ECC 2.0

Version 2.0 as at 21Sep08

Requirement: Pentium II 266MHz or better, 192MB RAM or better, Windows XP and Microsoft .NET Framework Version 1.1 Redistributable Package (23698KB) or better from http://www.microsoft.com.

Part A : Procedure

1) Double click the icon of "Mobile_ECC.exe". Then, Figure 1 will be shown.

Fig. 1 Starting screen of Mobile ECC (mobile elliptic curve cryptosystem)

2) This software application provides five functions: **Public key generation**, **hybrid encryption**, **hybrid decryption**, **signature generation** and **signature verification**.

3) Mobile ECC has a hybrid encryption system. It uses elliptic curve cryptosystem (ECC) for two-key encryption and **Rijndael cipher** (**AES**) for one-key encryption.

ECC of Mobile ECC implements the elliptic curve of P-192 proposed by CSRC (Computer Security Resource Center, URL: http://csrc.nist.gov), NIST (National Institute of Standards and Technology, URL: http://www.nist.gov), USA. Curve P-192 is in the document of FIPS PUB (Federal Information Processing Standards Publications): FIPS 186-2 with Change Notice 1 dated October 5, 2001 announced on 15 February 2000 (URL: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf). Any update can be referred at URL: http://csrc.nist.gov/cryptval/dss.htm.

Rijndael cipher of Mobile ECC implements the 192-bit AES (Advanced Encryption Standard) proposed by CSRC, NIST, USA. It is in the document of FIPS PUB 197 (URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf) announced on 26 November 2001.

The ECC of P-192 has security strength of 96 bits. 192-bit AES has security strength of 192 bits. Hence, the hybrid cryptosystem of Mobile ECC has security strength of 96 bits. This is sufficient for the recommended minimum security strength of 80 bits. NIST proposed security level of 80-bit key to be phased out by year 2015 and used until year 2010 (URL: http://en.wikipedia.org/wiki/Key_size). It is published in Table 4 page 66 of the document of NIST SP (Special Publication) 800-57 Recommendation for Key Management - Part 1: General (Revised) announced in May 2006 (URL: http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf).

ECRYPT (European Network of Excellence for Cryptology) within the European Commission's FP6 (Sixth Framework Programme), EU, (URL: http://www.ecrypt.eu.org) listed the protection period for various security levels according to the security strength in bits in Table 7.4 page 29 of "ECRYPT Yearly Report on Algorithms and Key Lengths (2005)" (URL: http://www.ecrypt.eu.org/documents/D.SPA.16-1.0.pdf).

Table 1 Security levels and protection periods from ECRYPT, EU

Security Level	Security (bits)	Protection	Comment
1	32	Attacks in "real-time" by individuals.	Only acceptable for authentication tag size.
2	64	Very short-term protection against small organizations	Should not be used for confidentiality in new systems.
3	72	Short-term protection against medium organizations, medium-term protection against small organizations	
4	80	Very short-term protection against agencies, long-term protection against small organizations.	Smallest general-purpose level, < 5 years protection.
5	112	Medium-term protection.	Approximately 20 years.
6	128	Long-term protection.	Good, generic application-independent recommendation, ≈ 30 years.
7	256	"Foreseeable future"	Good protection against quantum computers.

For Mobile ECC at **security strength of 96 bits**, its protection period is between the protection periods of 5 years (80 bits) and 20 years (112 bits). **Moore's Law** and **key strengthening** are applied and it has about **20 years of protection period**.

Moore's Law is the empirical observation that the transistor density of integrated circuits, with respect to minimum component cost, doubles every 24 months. It is attributed to Gordon E. Moore, a co-founder of Intel (URL: http://en.wikipedia.org/wiki/Moore_Law).

Key strengthening is explained as below using the password length equation.

Password length equation:

$$S = n * L * R / P$$

S = Key space / password space
n = Number of computers
L = Maximum lifetime of a password
R = Number of guesses per unit of time per unit of computer
P = Probability that a password can be guessed in its lifetime

Let:
S = Number of keys in bits
$n = 10^9 = 29.9$ bits
L = 5, 10, 20, 30, 50, 100, 300 years = 27.2, 28.2, 29.2, 29.8, 30.6, 31.6, 33.1 bits
$R = 1.5 \times 10^7 = 23.8$ bits (best PC performance by the end of year 2005)
R = 1 second per slow computer (using key stretching)
$P = 10^{-6} = -19.9$ bits

Considering variety of computers and Moore's Law:
Computer performance varies from 1 to 10 times.
0.05second per fast computer to 1 second per slow computer => $\log_2 20 = 4.3$ bits.

Moore's Law (ML):
The number of transistors per chip doubles every 2 years => [L (in year) / 2] bits.

$$S = (n * L * R / P) * 2^{4.3} * 2^{L/2}$$

Table 1 Taking $R = 1.5 \times 10^7 = 23.8$ bits (without key strengthening).

L (year)	n (bit)	L (bit)	R (bit)	P (bit)	0.05 – 1s	ML (bit)	S (bit)	ECC-2S
5	29.9	27.2	23.8	-19.9	0	2.5	103.3	206.6
10	29.9	28.2	23.8	-19.9	0	5	106.8	213.6
20	29.9	29.2	23.8	-19.9	0	10	112.8	225.6
30	29.9	29.8	23.8	-19.9	0	15	118.4	236.8
50	29.9	30.6	23.8	-19.9	0	25	129.2	258.4
100	29.9	31.6	23.8	-19.9	0	50	155.2	310.4
300	29.9	33.1	23.8	-19.9	0	150	256.7	513.4

The number of loops in the key stretching or key strengthening has to be increased by 1 bit for every 2 years. Then the security of a cryptosystem can be maintained at constant security level.

Table 2 Fixing at R = 1 for slow computer (with key strengthening).

L (year)	n (bit)	L (bit)	R (bit)	P (bit)	0.05 – 1s	ML (bit)	S (bit)	ECC-2S
5	29.9	27.2	0	-19.9	4.3	2.5	83.8	167.6
10	29.9	28.2	0	-19.9	4.3	5	87.3	174.6
20	29.9	29.2	0	-19.9	4.3	10	93.3	186.6
30	29.9	29.8	0	-19.9	4.3	15	98.9	197.8
50	29.9	30.6	0	-19.9	4.3	25	109.7	219.4
100	29.9	31.6	0	-19.9	4.3	50	135.7	271.4
300	29.9	33.1	0	-19.9	4.3	150	237.2	474.4

4) Mobile ECC also has a digital signature system. It uses **ECDSA** (**Elliptic Curve Digital Signature Algorithm**) for both signature generation and signature verification. Information about ECDSA can be obtained from ANSI (American National Standards Institute, URL: http://webstore.ansi.org), USA, for the document of ANSI X9.62:2005 "Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)" dated 16 November 2005. Another source is NIST (National Institute of Standards and Technology, URL: http://www.nist.gov), USA, for the document of FIPS 186-2 with Change Notice 1 dated October 5, 2001 announced on 15 February 2000 (URL: http://csrc.nist.gov/cryptval/dss.htm).

5) For **public key generation**, enter the private key into the textbox of "Private Key Field (192-bit)". Private key can be obtained either from the key hash of 2-dimensional key (2D key) input method and system, multilingual key input method and system, and/or 2-factor multimedia key input method and system.

If 2-dimensional key input method and system is used, enter the private key into the textbox of "Secret Key Field" until the effective key size in the textbox of "Key Entropy (bit)" is at least 192 bits. Click the button of "Hash Key" to hash the private key using SHA-512. Then click the button of "Copy Hash" to copy the 512-bit hash.

For the multilingual key input method and system, please read the user manual for various key styles. Once the private key is input to be at least 192 bits, all the following steps will be the same as 2-dimensional key input method and system.

Yet if 2-factor multimedia key input method and system are used, select a software token from a multimedia data type such as text, image, sound, or video. Create an at least 128-bit password the secret key using conventional method, 2D key method, or multilingual key. Use this password to encrypt the token to become a token ciphertext and store in a USB flash drive for future usage. Whenever the software token is needed, decrypt the token cuphertext using the password, and then click the button of "Hash Key" to generate a private key for Mobile ECC.

For **private key optimization of 192-bit Mobile ECC software**, when this software is used together with 2D key input method, there is a memory optimization to minimize the memorizable size of private key.

Firstly, a 128-bit 2D secret is created using 2D key to act as the token password in the 2-factor multimedia key method. Secondly, an at least 192-bit multimedia data file can be used as software token to be encrypted and stored in a USB flash drive as token ciphertext.

The token password (i.e. 2D key secret) and token plaintext, after the optional processes of multihash key, can generate a secret to act as a 192-bit private key secret in the 192-bit Mobile ECC. Whenever there is a need, get the USB flash drive storing the token ciphertext, decrypt it using the token password (i.e. 2D key secret), and select the hash iteration of multihash key to create a slave key.

Coming back to the Mobile ECC, click the button of "Paste Pte" to paste the private key. The 512-bit hash will be automatically truncated from the beginning to get a 192-bit private key. Press the button of "Create Public Key". Enter the file name of the public key file. Select either the file extension of ".txt" or ".pub". Click the button of "Save" and a public key file will be generated. The file directory and file name of the public key file is printed beside the label of "Destination File".

Optionally, please fill in the details of the public key holder into the public key file to enable future feature of public key certificate. **Public key certificate** is a signed file having public key and identity/identities. The signature(s) can be generated by any trust entities such as **certification authority (CA)** or any **introducer(s)** in the communication network. Do not change anything in the first 6 lines. For line 7 and above, fields can be added or deleted.

Lastly, **click** the **button** of "**Clear Pte**" to clear the input private key in the clipboard of the computer memory.

6) For **hybrid encryption**, check whether the public key file has already been included. If there is a file directory together with a file name beside the label of "Public Key File", public key file is included. If not, click the button of "Include Pub" to include the public key file.
Then check whether the source file for hybrid encryption has already been included. If there is a file directory together with a file name beside the label of "Source File", source file is included. If not, click the button of "Source File" to include the source file.

Click the button of "Encrypt". Enter the file name of the ciphertext file. Select either the file extension of ".txt" or ".mec". Click the button of "Save" and a ciphertext file will be generated. The file directory and file name of the ciphertext file is printed beside the label of "Destination File".

7) For **hybrid decryption**, check whether the ciphertext file has already been included. If there is a file directory together with a file name beside the label of "Source File", ciphertext file is included. If not, click the button of "Source File" to include the ciphertext file.

Then enter the private key into textbox of "Private Key Field (192-bit)" as in step 4.

Click the button of "Decrypt". The file name of the plaintext file is printed beside the label of "Destination File". The plaintext file is in the working file directory of Mobile ECC.

Lastly, **click** the **button** of "**Clear Pte**" to clear the input private key in the clipboard of the computer memory.

8) For **signature generation**, check whether the message file has already been included. If there is a file directory together with a file name beside the label of "Source File", message file is included. If not, click the button of "Source File" to include the message file.

Then enter the private key into textbox of "Private Key Field (192-bit)" as in step 4.

Click the button of "Sign". Enter the file name of the signature file. Select either the file extension of ".txt" or ".sig". Click the button of "Save" and a signature file will be generated. The file directory and file name of the signature file is printed beside the label of "Destination File".

Optionally, please fill in the file name of the signed file into the signature file. Do not change anything in the first 6 lines. For line 7 and above, fields can be added or deleted.

Lastly, **click** the **button** of "**Clear Pte**" to clear the input private key in the clipboard of the computer memory.

9) For **signature verification**, check whether the public key file has already been included. If there is a file directory together with a file name beside the label of "Public Key File", public key file is included. If not, click the button of "Include Pub" to include the public key file.

Second, check whether the message file has already been included. If there is a file directory together with a file name beside the label of "Source File", message file is included. If not, click the button of "Source File" to include the message file.

Third, check whether the signature file has already been included. If there is a file directory together with a file name beside the label of "Signature File", signature file is included. If not, click the button of "Signature" to include the signature file.

To verify the message file and signature file using the public key file, click the button of "Verify". If the textbox of "Message" displays "**Valid**", the message file and signature file are **verified**. Or else if the textbox of "Message" displays "**Invalid**", the message file and signature file shall be **rejected**.

10) Files with the extensions of ".pub", ".mec" and ".sig" can be viewed using WordPad.

11) There are two indirect functions for this software application derived from its direct functions: **Key exchange** and **challenge-and-response authentication protocol without shared secret**.

12) For **key exchange**, let Alice and Bob to be the two parties intending to set up a secure communication channel using one-key encryption or symmetric key encryption.

Alice has public key (pub_A) and private key (pte_A). Bob has public key (pub_B) and private key (pte_B).

Alice jots down a secret key with timestamp into a text file, encrypts it with pub_B, and sends the ciphertext to Bob via a communication channel using two-key encryption or asymmetric key encryption. Then, Bob decrypts the Alice's ciphertext using pte_B, gets the secret key with timestamp, encrypts it using pub_A, and sends the ciphertext to Alice. Lastly, Alice decrypts the Bob's ciphertext using pte_A and verifies the secret key with timestamp. If the verification is accepted, the key exchange between Alice and Bob is successful and they can start communicating via the established communication channel using one-key encryption. If it is rejected, it fails.

13) For **challenge-and-response authentication protocol without shared secret**, let Alice and Bob to be the two parties intending to authenticate between themselves in a secure communication channel using two-key encryption. It is similar to the key exchange except the secret key is replaced with a random message.

Alice has public key (pub_A) and private key (pte_A). Bob has public key (pub_B) and private key (pte_B).

For Alice to authenticate Bob, Alice jots down a random message with timestamp into a text file, encrypts it with pub_B, and sends the ciphertext as **challenge** to Bob. Then, Bob decrypts the Alice's ciphertext using pte_B, gets the random message with timestamp, encrypts it using pub_A, and sends the ciphertext as **response** to Alice. Lastly, Alice decrypts the Bob's ciphertext using pte_A and verifies the random message with timestamp. If the verification is accepted, the identity of Bob is authenticated. Or else if the verification is rejected, the identity of Bob is refused.

For Bob to authenticate Alice, Bob does the similar tasks. Bob jots down a random message with timestamp into a text file, encrypts it with pub_A, and sends the ciphertext as **challenge** to Alice. Then, Alice decrypts the Bob's ciphertext using pte_A, gets the random message with timestamp, encrypts it using pub_B, and sends the ciphertext as **response** to Bob. Lastly, Bob decrypts the Alice's ciphertext using pte_B and verifies the random message with timestamp. If the verification is accepted, the identity of Alice is authenticated. Or else if the verification is rejected, the identity of Alice is refused.

At the end, Alice and Bob have mutually authenticated one another.

Part B : Urgency & Importance of Matters

B.1 Strongest Communication Modes Between Two or More Humans :

[1] After checking if there is no active mobile phone and there is no surveillance like sound recorder, video camera, camcorder, etc., by using the electronic detector(s), then the face-to-face communication between two or more humans is the strongest communication mode.

[2] To totally avoid surveillance, especially sound, build a closed computer communications network in a meeting room, where each participant can use a computer linked inside the meeting room only and cut off from the external computer network like LAN and Internet.

[3] The smallest set-up of item [2] is two computers linked using a direct cable in a closed room that can be used by two humans for bi-computer communications.

B.2 Recommended Computer Communications Network :

[1] Matters (Urgent, Important, Small Message) : Face-to-Face, Phone using semantic cryptography, Email using MePKC.

[2] Matters (Urgent, Important, Big Message) : Face-to-Face, Email using MePKC followed by a Phone Reminder.

[3] Matters (Urgent, Not Important, Small Message) : Instant Messenger, Email, Face-to-Face, Phone (Mobile, Fixed Line).

[4] Matters (Urgent, Not Important, Big Message) : Face-to-Face, Email.

[5] Matters (Not Urgent, Important, Small Message) : Face-to-Face, Email using MePKC followed by a Phone Reminder.

[6] Matters (Not Urgent, Important, Big Message) : Face-to-Face, Email using MePKC followed by a Phone Reminder.

[7] Matters (Not Urgent, Not Important, Small Message) : Instant Messenger, Email.

[8] Matters (Not Urgent, Not Important, Big Message) : Email.

Part C : Using Public Key Certificate & Meeting Face-to-Face

C.0 Assumptions :

Alice = First party acting as a sender
Bob = Second party acting as a receiver

C.1 From Public Key File to Public Key Certificate :

[1] Alice and Bob meet face-to-face.

[2] Alice and Bob exchange their public key file and sign the public key file of opposite party to generate public key certificate.

[3] Alice and Bob mutually act as introducer. Alice is the introducer of Bob, and Bob is the introducer of Alice.

[4] Alice and Bob mutually exchange the signature of the public key file.

[5] Alice and Bob both obtain their public key certificate consisting of owner's public key file and its signature signed by the introducer.

[6] Public Key Certificate = Public Key File + Its Signature File Signed by an Introducer

[7] The security of this method is the strongest for the exchange of public key certificates to start building a web of trust full with introducers.

Part D : Using Public Key Certificate with the Existence of a Direct Introducer

D.1 From Public Key File to Public Key Certificate :

[1] When a public key file is signed by a trusted person known by two parties wishing to have a remote communication, the public key file will become a public key certificate refereed by the trusted person, who acts as an introducer.

[2] Public Key Certificate = Public Key File + Its Signature File Signed by an Introducer

[3] When both the sender and receiver have exchanged their respective public key certificate via email, mutually verify the introducer's digital signature of the public key file.

[4] The secure computer communications between the sender and receiver via an insecure channel can now be established.

[5] The security of this method is subject to the trust level of the direct introducer.

Part E : Using Public Key File without the Existence of a Direct Introducer

E.0 Assumptions :

Alice = First party acting as a sender
Bob = Second party acting as a receiver

E.1 First Method – Uploading the Alice's Public Key File :

[1] Alice can upload her public key file to her website that is open to the public.

[2] Bob can do the same like Alice.

[3] If Bob does not have personal website, then he can try Method E.2 steps 1 to 10.

[4] The security of this method is subject to the website attacks.

E.2 Second Method – Securely Sending Alice's Public Key File to Bob Using Email & Mobile Phone :

[1] Alice uses the software of MobileECC (Beta version) and creates a public key file called F1 by keeping a private key P1 as secret.

[2] The secret P1 can be copied into a text file or Microsoft Word document, and then stored it into a USB flash drive as a token.

[3] Alice fills in her details into the F1.

[4] Select all and copy all the text in F1, and paste it into a Microsoft Word document called F2.

[5] After that, Alice encrypts the F2 using the security function at the MS Word tool bar [Tools > Options > Security] by using a symmetric key/password K1.

[6] Then, Alice sends F1 and encrypted F2 to Bob using email.

[7] Alice sends the key/password K1 to Bob using her mobile phone to Bob's mobile phone.

[8] Bob decrypts file F2 using key K1 to get a message to be compared with file F1.

[9] If the decrypted F2 is different from the F1, Bob rejects Alice's public key file.

[10] If the decrypted F2 is the same as the F1, Bob accepts Alice's public key file.

[11] Bob can do the same like Alice to send his public key file to Alice.

[12] The security of this method is subject to the interception attacks of email and mobile phone by hackers and network administrators.

E.3 Third Method – Securely Sending Alice's Public Key File to Bob Using Email, (Floppy Disk/CD/DVD/USB Flash Drive) & Postal Mail :

[1] Alice uses the software of MobileECC (Beta version) and creates a public key file called F1 by keeping a private key P1 as secret.

[2] The secret P1 can be copied into a text file or Microsoft Word document, and then stored it into a USB flash drive as a token.

[3] Alice fills in her details into the F1.

[4] Select all and copy all the text in F1, and paste it into a Microsoft Word document called F2.

[5] After that, Alice encrypts the F2 using the security function at the MS Word tool bar [Tools > Options > Security] by using a symmetric key/password K1.

[6] Then, Alice sends F1 and encrypted F2 to Bob using email.

[7] Alice copies the key/password K1 into storage medium like floppy disk, CD, DVD, USB flash drive, etc., and sends the storage medium with key/password K1 to Bob using postal mail.

[8] Bob decrypts file F2 using key K1 to get a message to be compared with file F1.

[9] If the decrypted F2 is different from the F1, Bob rejects Alice's public key file.

[10] If the decrypted F2 is the same as the F1, Bob accepts Alice's public key file.

[11] Bob can do the same like Alice to send his public key file to Alice.

[12] The security of this method is subject to the interception attacks of email and postal mail by hackers and network administrators.

E.4 Fourth Method – Securely Sending Alice's Public Key File to Bob Using Several Serial Introducers:

[1] Alice establishes a serial path with several nodes representing the introducers from herself to Bob.

[2] Serial Path: Alice – Introducer_1 – Introducer_2 – … – Introducer_N – Bob

[3] Alice uses the software of MobileECC (Beta version) and creates a public key file called F1 by keeping a private key P1 as secret.

[4] The secret P1 can be copied into a text file or Microsoft Word document, and then stored it into a USB flash drive as a token.

[5] Alice fills in her details into the F1.

[6] Select all and copy all the text in F1, and paste it into a Microsoft Word document called F2.

[7] After that, Alice encrypts the F2 using the security function at the MS Word tool bar [Tools > Options > Security] by using a symmetric key/password K1.

[8] Then, Alice sends F1 and encrypted F2 to Introducer_1 using email.

[9] Alice sends the key/password K1 to Introducer_1 using her mobile phone to Introducer_1's mobile phone.

[10] Introducer_1 decrypts file F2 using key K1 to get a message to be compared with file F1.

[11] If the decrypted F2 is different from the F1, Introducer_1 rejects Alice's public key file.

[12] If the decrypted F2 is the same as the F1, Introducer_1 accepts Alice's public key file.

[13] Introducer_1 signs Alice's public key file to generate a digital signature S1.

[14] Introducer_1 sends the generated digital signature S1 to Alice, which when combined with Alice's public key file, they become a public key certificate.

[15] Introducer_1 can do the same like Alice to connect to Introducer_2.

[16] Steps [6-14] are repeated from Introducer_1 to Introducer_N so as to finally link to Bob.

[17] In opposite direction, Bob can do the same like Alice to send his public key file to Alice.

[18] The security of this method is subject to the trust level of the introducers, interception attacks of email and mobile phone by hackers and network administrators.

E.5 Fifth Method – Securely Sending Alice's Public Key File to Me the Software Author who Acts as an Introducers :

[1] Alice uses the software of MobileECC (Beta version) and creates a public key file called F1 by keeping a private key P1 as secret.

[2] The secret P1 can be copied into a text file or Microsoft Word document, and then stored it into a USB flash drive as a token.

[3] Alice fills in her details into the F1.

[4] Select all and copy all the text in F1, and paste it into a Microsoft Word document called F2.

[5] After that, Alice encrypts the F2 using the security function at the MS Word tool bar [Tools > Options > Security] by using a symmetric key/password K1.

[6] Then, Alice sends F1 and encrypted F2 to me the software author using email.

[7] Alice sends the key/password K1 to me using her mobile phone to my mobile phone number [+6013-6134998 @ Malaysia].

[8] Steps [4-7] can be replaced by steps [9-11].

[9] After that, Alice encrypts the F1 using the encryption function of MobileECC with my attached public key file [xpreeli_public_key.pub]. This public key file [xpreeli_public_key.pub] is safer to be downloaded from my website at http://www.geocities.com/xpree/mobileECC.htm.

[10] Then, Alice sends encrypted public key file called F2 to me using email.

[11] The email communications used in steps [6, 10] can be replaced by the storage media of floppy disk, CD, DVD, USB flash drive, etc., and communicated using postal mail.

[12] Bob decrypts file F2 using key K1 to get a message to be compared with file F1.

[13] If the decrypted F2 is different from the F1, Bob rejects Alice's public key file.

[14] If the decrypted F2 is the same as the F1, Bob accepts Alice's public key file.

[15] Bob can do the same like Alice to send his public key file to Alice.

[16] The security of this method is subject to my trust level, website attack, interception attacks of email and postal mail by hackers and network administrators.

Part F : Symmetric Key Sharing Scheme Using MePKC/MoPKC

F.1 A Symmetric Key Encrypting a Plaintext is Shared Using MePKC/MoPKC between Two Parties:

[1] To increase the security level of the MobileECC, the ciphertext of MobileECC has to be short and random.

[2] This is possible if the ciphertext of MobileECC is just a short message of symmetric key.

[3] Create a symmetric key as K1 can store it into a text file as F1.

[4] Create your message M1 in the file format of Microsoft Word document, and encrypt it as file F2 by using the same symmetric key K1.

[5] For other data format, compression software with encryption function like WinZip can be used. During the data compression, encrypt the data M1 as file F2 by using the same symmetric key K1.

[6] Encrypt file F1 using the receiver's public key and MobileECC to create a ciphertext C1.

[7] Send the files F2 and C1 to the receiver.

[8] The receiver recovers symmetric key K1 from file C1 by using receiver's private key.

[9] Recovered K1 is used by the receiver to decrypt file F2 to get back message M1.

[10] The receiver can use the same symmetric key K1 to communicate with the sender.

[11] The security of this method by using a short encrypted symmetric key as the ciphertext of MobileECC is stronger than the computer communications of MobileECC operating on a big message directly.

Part G : OTP (One-Time Password) Scheme Using MePKC/MoPKC

G.1 Application for Key Sharing Scheme between Two Parties :

[1] The symmetric key in Part F can be increased to get a list of OTP shared by two parties.

[2] The OTP can be any string of random ASCII characters.

[3] The OTP list is exchanged between the Alice and Bob using the MePKC/MoPKC.

[4] For each computer communication of electronic file, a symmetric key in the OTP list is used and stricken through, marked or removed.

[5] In the next computer communication of electronic file, a new symmetric key, which is the next, in the OTP is used.

[6] The electronic file in this section can be the file attachment of email.

[7] The security of this method is subject to the obscurity of OTP list, security of MePKC/MoPKC and symmetric key encryption.

[8] The number of OTP lists will be too big when it involves too many bipartite communications. Hence, this method is encouraged to be used among important users.

G.2 Application for Instant Messaging Scheme using Instant Messenger between Two Parties :

[1] The symmetric key in Part F can be increased to get a list of pairs (ID, OTP) shared by two parties.

[2] The ID can be ordered numbers or any string of random ASCII characters.

[3] The OTP can be any string of random ASCII characters.

[4] The (ID, OTP) pair list is exchanged between the Alice and Bob using the MePKC/MoPKC.

[5] The instant messengers are like Yahoo! Messenger, MSN/Windows Messenger, Skype, AOL Messenger, Google Talk, Miranda, etc.

[6] For each computer communication of instant messaging using the instant messenger between Alice and Bob, an ID is sent from Alice to Bob.

[7] If the ID has been used before, Bob asks Alice to send another ID for a maximum of three ID.

[8] When Bob gets an ID that has not been used before, he looks up the (ID, OTP) pair list to get the corresponding OTP and sends the OTP to Alice using the instant messenger.

[9] Then, Bob strikes through or removes the corresponding (ID, OTP) pair.

[10] When Alice receives the OTP from Bob, if the (ID, OTP) pair is not matched, then she asks Bob to send the OTP again for a maximum of three OTP.

[11] When Alice has matched the OTP received from Bob, she strikes through, marked or removed the corresponding (ID, OTP) pair.

[12] Now, Alice and Bob can start to send instant messages.

[13] The security of this method is subject to the man-in-the-middle attack, obscurity of (ID, OTP) list, security of MePKC/MoPKC and symmetric key encryption.

[14] Due to the threat of man-in-the-middle attack, the instant messaging using (ID, OTP) list communicated with MePKC/MoPKC is not encouraged for important matters. It is only suggested for the applications of casual chatting, discussion, and meeting.

[15] For urgent matters, please use the telephone, be it mobile or fixed line.

[16] The number of (ID, OTP) lists will be too big when it involves too many bipartite communications. Hence, this method is encouraged to be used among important users.

*** The End ***