SCIS 2006 The 2006 Symposium on Cryptography and Information Security
Hiroshima, Japan, Jan. 17-20, 2006
The Institute of Electronics, Information and Communication Engineers

A User-Friendly Implementation of PSEC-KEM and ECDSA on the Windows Environment

張惟淙*1 Wei-Tsung Chang
楊中皇*2 Chung-Huang Yang

Abstract— PSEC-KEM key encapsulation mechanism is based on the elliptic curve cryptography, in which is developed by the NTT company in Japan can utilize for key agreement schemes. Due to the fact that implementation modules of PSEC-KEM provided by the official source of NTT only support command mode operation, in this research, we developed a user-friendly implementation of PSEC-KEM on the Windows environment, and use the keypair generation function of PSEC-KEM for our ECDSA implementation. In generating and verifying ECDSA digital signature, we use SHA-2 hash algorithm to compute the message digest of input message. Our software is created with Borland C++Builder 6 and it allows the user to choose optionally using Java Card as the medium for key storage. The user may regard the Java Card as a keyring, it may store both user's public key and private key, and other people's public keys, this may improve the security of key use and management.

Keywords: Elliptic Curve Cryptography, PSEC-KEM, Digital Signature, ECDSA, Java Card

1 Introduction

In 1999, NTT developed the public key encryption called PSEC (Provably Secure Elliptic Curve encryption) based on the elliptic curve discrete logarithm problem. Subsequently, the framework called the key encapsulation mechanism KEM was proposed as a delivery method for secret keys used in symmetric key encryption such as AES. The corresponding form called PSEC-KEM was developed in 2001. The development and application of PSEC-KEM is always devoted to standardization, and was selected by the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project of European Union and the CRYPTREC (Cryptography Research and Evaluation Committees) project of Japan in 2003. Afterward, in 2005, PSEC-KEM was certified as the IETF standard cipher for RFC4051 [1].

In this research, we developed a user-friendly implementation of PSEC-KEM on the Windows environment integrating the source code of PSEC-KEM, and it is created with Borland C++Builder 6 [2]. The software may facilitate the person who wants to understand PSEC-KEM easily.

One of the applications of elliptic curve cryptography (ECC) is digital signature. PSEC-KEM provided the ECC public key and private key generation function, so we use the keypair generation function for our ECDSA (Elliptic Curve Digital Signature Algorithm) implementation to generate the ECC keypair for generation and verification of ECDSA digital signature. The software allows the user to choose ECC parameters which are suggested by the ECDSA standards such as FIPS PUB 186-2 [3] or ANSI X9.62 [4] for the keypair generation function.

Furthermore, most of ECDSA digital signature implementation uses SHA-1 hash algorithm to compute the message digest of input message, but NIST(National Institute of Standards and Technology of the U.S. Department of Commerce) announced that SHA-1 will be phased out in 2010 [5]. Therefore, in generating and verifying ECDSA digital signature, we use the newer SHA-2 hash algorithm (which contains SHA-256, SHA-384 and SHA-512 sub-algorithms) [6] and integrate SHA-2 source code of GnuPG [7] into the software.

The software also integrates smart card technology. Smart cards are very useful in the areas of personal

* Institute of Information and Computer Education, National Kaohsiung Normal University, 116, Ho Ping First Road, Kaohsiung 802, TAIWAN.
[1] Email: wtcem@hotmail.com.
[2] Web: http://crypto.nknu.edu.tw/, Email: chyang@computer.org.

security, and they can be used to add authentication and secure access to information systems that require a high level of security and stored information is portable [8]. The software allows the user to choose optionally using Java Card as the medium for key storage. The user may regard the Java Card as a keyring, it may store and retrieve both user's keypair and other people's public keys, and this may improve the security of key use and management.

2 Implementation of PSEC-KEM on the Windows Environment

There are three mainly functions in PSEC-KEM: ECC keypair generation, public key encapsulation (encryption) and decapsulation (decryption). Figure 1 displays the main form of the software.

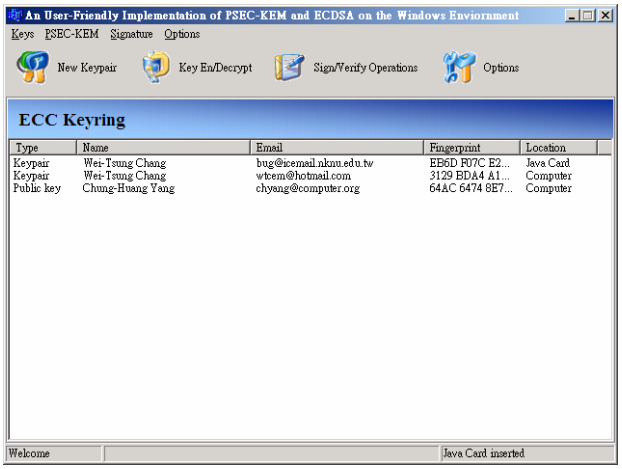

Figure 1: Main form

When the software starts, it could detect ECC keys (including user's keypairs and other people's public keys) which were stored in computer and display them in the ECC keyring list of the main form. If user uses the Java Card, the software could retrieve the keys from the card and put them on the ECC keyring form and refresh the form. ECC keys could be imported/exported from computer to the Java Card according to user's requirements, and contrariwise. When user add, edit, delete, import or export keys, the software would refresh the ECC keyring list.

Figure 2 shows the new keypair form which contains the following required fields: a. Name - This is the first and last name of the user. b. Email - The e-mail address of the user. c. Passphrase - The passphrase which protects the user's private key of the new keypair, and we use the AES (AES-128) symmetric encryption algorithm to protect the private key. Moreover, we provided 15 choice of curve (P-XXX means the elliptic curve on prime field, and B-XXX or K-XXX means the elliptic curve on binary field) suggested by the FIPS PUB 186-2 document. The software allows the user to choose optionally using Java Card as the medium for key storage (the card must be inserted and verified successfully), or save keys in computer.

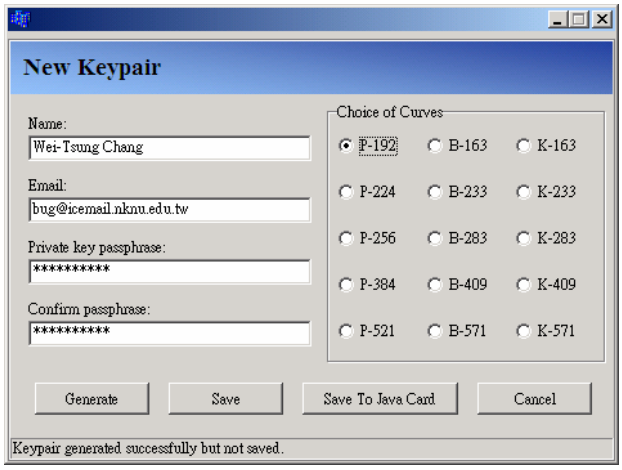

Figure 2: Add a new keypair

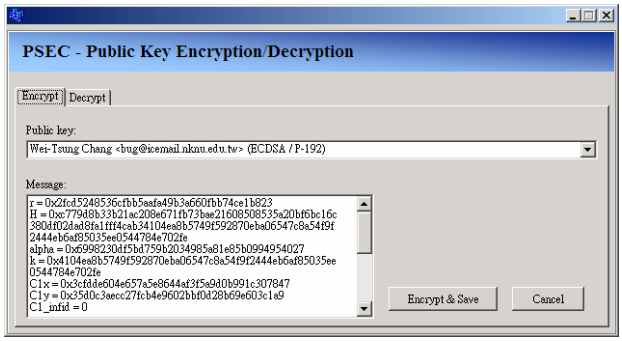

Figure 3: PSEC-KEM Public key encryption

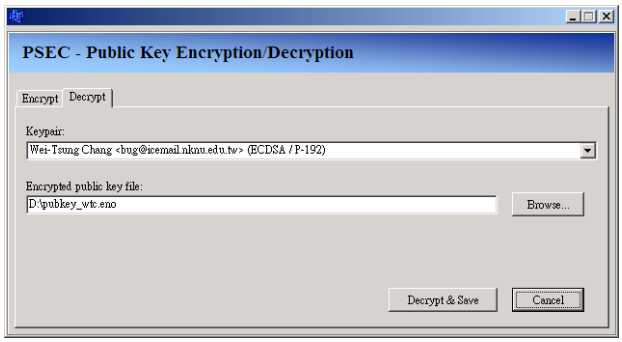

Figure 4: PSEC-KEM Public key decryption

Figure 3 shows the public key encryption function of PSEC-KEM on the Windows environment. The user just choose a public key which displayed by Public key combo box (identified by user information which including name and email), and then press the Encrypt & Save button to encrypt the public key chosen by the user and save the output as a file. The

related messages of the public key encryption process would show in the Message memo box. And if the user would like to decrypt the encrypted public key, he only needs to open the encrypted public key file and choosing the keypair including the original public key as shown in Figure 4.

3 ECDSA Implementation

In this research, we use the keypair generation function of PSEC-KEM for ECDSA implementation to develop ECDSA signature generation/verification functions. The keypairs used in ECDSA signature generation/verification are displayed in the ECC keyring list of the main form as shown in Figure 1.

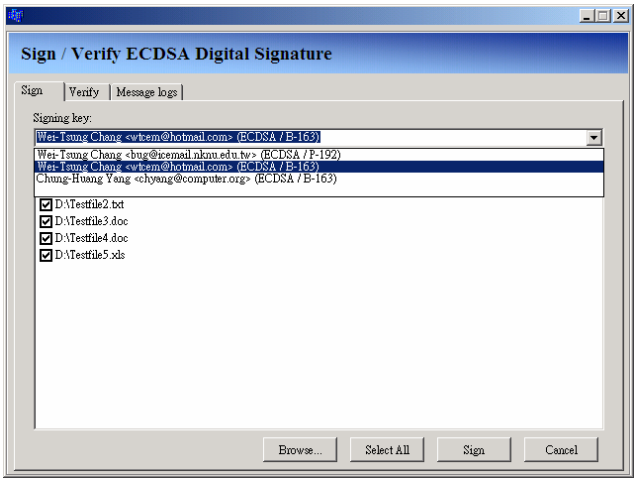

Figure 5: ECDSA signature generation

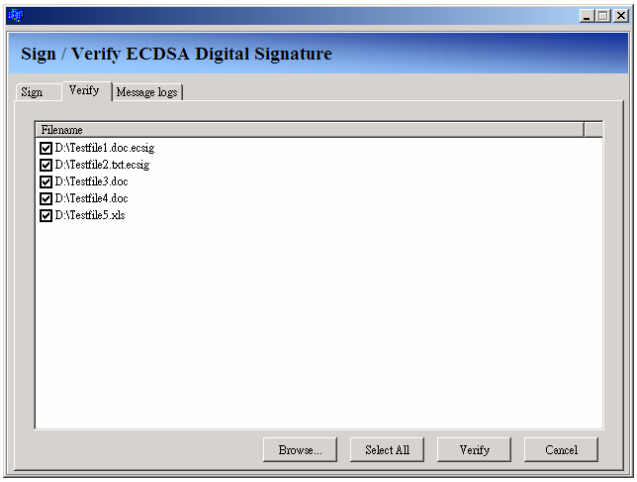

Figure 6: ECDSA signature verification

Figure 5 shows the ECDSA signature generation form. The user may browse and select files, and choose a private key which displayed by Signing key combo box to sign ECDSA signatures. The software saves signature file as .ecsig (detached ECDSA signature file which is commonly used). Moreover, if the user would like to verify ECDSA signatures, only need to choose .ecsig files as shown in Figure 6.

4 Conclusion

Although PSEC-KEM mainly utilized for key agreement schemes, it has very good flexibility on the ECC keypair generation function. Therefore, we not only created a user-friendly implementation of PSEC-KEM on the Windows environment, but also use the keypair generation function of PSEC-KEM for our ECDSA implementation integrating Java Card, and this may be considered as an expansion application of PSEC-KEM.

References

[1] NTT corporation, PSEC-KEM, http://info.isl.ntt.co.jp/crypt/eng/psec/index.html.

[2] Borland software corporation, C++Builder, http://www.borland.com/us/products/cbuilder/index.html.

[3] NIST, FIPS 186-2, "Digital Signature Standard", http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, 2001.

[4] ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA) , 1998.

[5] NIST, NIST Brief Comments on Recent Cryptanalytic Attacks on SHA-1, http://csrc.nist.gov/hash_standards_comments.pdf, 2005.

[6] NIST, FIPS PUB 180-2, "Secure Hash Standard", http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, 2002.

[7] The GnuPG project, http://www.gnupg.org/.

[8] C.E. Ortiz, "An Introduction to Java Card Technology", http://developers.sun.com/techtopics/mobility/javacard/articles/javacard1/, 2003.