PSEC-KEM與ECDSA的橢圓曲線密碼軟體設計與實現

張惟淙
國立高雄師範大學資訊教育所
bug@icemail.nknu.edu.tw

楊中皇
國立高雄師範大學資訊教育所
chyang@nknucc.nknu.edu.tw

摘要

近年來，橢圓曲線密碼系統(Elliptic Curve Cryptosystem，ECC)，已經開始挑戰 RSA。就相同的安全性而言，ECC 所需要的密碼學金鑰長度較 RSA 短，而且有更佳的執行效率。因此在公開金鑰密碼學的實務應用研究領域，已有逐漸取代 RSA 成為主流的趨勢。

本研究旨在開發一個具備 ECC 金鑰交換與數位簽章功能的軟體。主要利用日本 NTT 公司的 PSEC-KEM 原始碼與自行設計的 ECDSA 數位簽章模組開發視窗介面的 ECC 軟體。PSEC-KEM 乃是基於 ECC 的金鑰封裝機制，主要用於金鑰交換。我們依據美國 NIST 的 FIPS 186-2 數位簽章標準建議的橢圓曲線參數，實作 5 種質數體及 10 種二元體等共 15 種 ECDSA 數位簽章。而在整合 PSEC-KEM 功能時，則以其金鑰對(Keypair)產生模組，建立均可用於 PSEC-KEM 金鑰交換及質數體數位簽章的金鑰對，這是由於 PSEC-KEM 目前僅支援質數體。而二元體數位簽章所使用的金鑰對仍以我們自有的程式模組產生。此外本研究以 SHA-2 作為數位簽章使用的雜湊函數演算法，並且將 PSEC-KEM 的 KDF 函數所使用的 SHA-1 改良為 SHA-2，程式的實作則參考整合 GnuPG 的 SHA-2 原始碼。我們的軟體還設計讓使用者可選擇以 Java Card 儲存本身的 ECC 金鑰對及通訊對方的公開金鑰，藉此提升金鑰使用上的便利與安全。

關鍵詞：金鑰交換、數位簽章、PSEC-KEM、ECDSA、Java Card

1. 前言

西元1999年時，日本NTT公司發展出一套基於橢圓曲線密碼學的Public Key加密機制，名為PSEC (Provably Secure Elliptic Curve encryption)，其後的發展架構定位為金鑰封裝機制 (Key Encapsulation Mechanism, KEM)，主要用於金鑰交換，可以保護及遞送對稱式加密演算法(如AES)所使用的秘密金鑰(Secret Key)，因此在西元2001年以後所發展的版本一律以PSEC-KEM作為其正式名稱[10]。PSEC-KEM的發展及應用一向致力於標準化，迄今已為歐盟的NESSIE (New European Schemes for Signatures, Integrity, and Encryption)計畫[6]及日本的CRYPTREC (Cryptography Research & Evaluation Committees)計畫[3]所接受，且在西元2005年時納入IETF的RFC4051標準中[5]。

由於PSEC-KEM官方網頁所提供的原始碼模組僅用於命令模式介面測試及使用，因此本研究以Borland C++Builder 6整合PSEC-KEM原始碼開發一個可用於Windows系統環境的視窗使用介面軟體，以便讓有心研究PSEC-KEM的人士更易操作及明瞭其功用。

除了金鑰交換，ECC的另一項重要的實務應用是數位簽章。本研究所開發的軟體依據美國NIST的FIPS 186-2數位簽章標準[7]所建議的P-192等5種質數體參數及B-163、K-163等10種二元體參數，總共實作了15種不同金鑰參數長度的橢圓曲線數位簽章供使用者選擇運用。PSEC-KEM本身提供產生質數體ECC金鑰對的功能，因此我們將此功能整合自行開發的ECDSA (Elliptic Curve Digital Signature Algorithm) 質數體數位簽章模組，同時應用於ECDSA數位簽章的產生與檢驗，以及PSEC-KEM封裝機制中的加解密功能。至於二元體數位簽章需要使用的金鑰對，則仍以我們自行設計的ECDSA二元體金鑰產生模組來建立。

此外由於目前大多數的ECDSA數位簽章所用以計算訊息摘要的雜湊函數演算法為SHA-1，而且NTT的PSEC-KEM原始碼程式中的KDF密碼導出函數亦是基於SHA-1。但是美國國家標準技術研究院(National Institute of Standards and Technology, NIST)已宣佈將於2010年後不再支持有安全疑慮的SHA-1演算法[9]。因此我們選擇採用較新的SHA-2演算法，並整合GnuPG的SHA-2原始碼[14]作為數位簽章使用的雜湊函數演算法，以及將PSEC-KEM的KDF函數運算所使用的SHA-1改良為SHA-2。

本研究所開發的軟體尚結合智慧卡的使用，我們以IBM的JCOP20 Java Card為例，設計讓使用者可以選擇以智慧卡儲存自己的ECC金鑰對及通訊對方的公開金鑰，藉此提升金鑰使用上的便利與安全。

2. 文獻探討

2.1 橢圓曲線密碼學

西元 1985 年時，美國華盛頓大學的 Neal Koblitz[12]及 IBM 的 Victor Miller[15]各自提出以橢圓曲線演算法設計公開金鑰演算法的密碼技術，此

後更發展出許多關於橢圓曲線密碼系統的國際標準，如 ISO 11770-3、ANSI X9.62、IEEE P1363、FIPS 186-2 等。就相同的安全性而言，ECC 所需要的密碼學金鑰長度較 RSA 短，如表 1 所示[8]。若以 RSA 或 ECC 用於金鑰交換來保護 256 位元的 AES 金鑰時，RSA 的公開金鑰長度應為 15360 位元，相對的 ECC 僅需要使用 512 位元的金鑰。所以無論從增快執行速度或節省空間的角度來看，可見 ECC 是優於 RSA。

表 1 相同安全性時，RSA 與 ECC 金鑰長度比較

安全性 / 演算法	2^{128}	2^{192}	2^{256}
RSA 金鑰長度(bits)	3072	7680	15360
ECC 金鑰長度(bits)	256	384	512
金鑰長度比	12:1	20:1	30:1

2.2 KEM(Key Encapsulation Mechanism)

KEM 是由紐約大學 Victor Shoup 博士所提出的金鑰封裝機制，目前已納入 ISO/IEC-18033-2 標準[16]。KEM 的主要功能是應用於混合加密模式中遞送對稱式加密演算法(如 AES)金鑰，因此 KEM 的實作必須基於非對稱式加密演算法，如 RSA 或 ECC 等。與 KEM 相輔相成的是 DEM (Data Encapsulation Mechanism)資料封裝機制，其功用在於混合加密模式中對實際欲通訊的資料以對稱式加密演算法加密後進行傳輸，由於 DEM 並非本研究的範圍，因此不予贅述。KEM 的實作必須包含三種演算法模組，功能概略說明如下：

(1)*KEM.KeyGen*()金鑰產生演算法：用以產生金鑰對，即成對的 Public Key(*PK*)與 Private Key(*SK*)。

(2)*KEM.Encrypt*(*PK*, *options*)加密演算法：輸入 Public Key(*PK*)及一組參數或亂數，輸出結果為一把金鑰 *K*(對稱式加密演算法用的金鑰)和一組密文 C_0。

(3)*KEM.Decrypt*(*SK*, C_0)解密演算法：輸入 Private Key(*SK*)及密文 C_0，解密後得到 *K*。

2.3 PSEC-KEM

PSEC-KEM 包含三個主要功能模組：金鑰產生模組(KGP-PSEC)、Public Key 加密模組(EP-PSEC)及 Public Key 解密模組(DP-PSEC)[11]。以下簡介各模組運算的輸入參數、輸出訊息及演算步驟：

2.3.1 KGP-PSEC

PSEC-KEM 的金鑰產生模組，主要的功能在於產生 ECC 金鑰對，亦即 Public Key 及 Private Key。

輸入參數：

- *E*：橢圓曲線的參數，包括有限體與橢圓曲線的選擇。
- *KDF*：基於 SHA-1 演算法的密碼導出函數(Key Derivation Function)。依據 PKCS #5[13]文件的定義，KDF 函數的參數為 P (Password)跟 S (Salt)，輸出為 DK (Derived Key)。S 是一個由許多亂數而成的集合，P 是指定的值。KDF 函數的作用在於產生一組 Keys，每次執行都會產生不同的 Key，這些 Keys 可以用來作為混合加密模式中的 Session Key。經由 KDF 函數所產生的 Key 可以有效防止暴力攻擊法或字典攻擊法破解。PSEC-KEM 中 KDF 函數的運算乃是基於 SHA-1 演算法，在本研究中則改良為 SHA-256。KDF 的運作如圖 1。
- *hLen*：一個非負的整數。

輸出訊息分別為 *PK* (Public Key)及 *s* (Private Key)：

- *PK*：包含 *E*、*W*、*KDF* 及 *hLen* 等資訊，其中 *W* 由 $s \bullet P$ 而得，是橢圓曲線 *E* 上的一個點。*P* 則為橢圓曲線 *E* 的基點。
- *s*：為一個非負整數，其值必須小於橢圓曲線 *E* 的級數 *p*。

演算步驟：

(1) 產生一個亂數整數 *s*，*s* 值的範圍介於 0 與 *p* − 1 之間。

(2) 以點乘法計算 $W = s \bullet P$。

(3) 輸出橢圓曲線金鑰對：Public Key 為 *PK* = (*E*, *W*, *KDF*, *hLen*)，Private Key 即為 *s*。

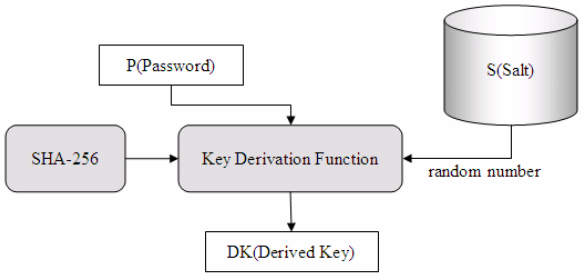

圖 1 KDF 密碼導出函數運作圖

2.3.2 EP-PSEC

輸入參數：

- *PK*：PSEC-KEM 的 Public Key。
- α：為一個亂數，其值必須為非負整數，且小於橢圓曲線 *E* 的級數 *p*。實際應用上，該值將為在演算法過程中以 *KDF* 函數導出混合加密模式用的對稱式演算法之加解密金鑰 *K*。

輸出的加密訊息是下列兩者：

- *Q*：等於對稱式加密演算法金鑰 *K*。
- C_1：不包含金鑰 *K* 的密文訊息，但可依據計算出金鑰 *K*。

演算步驟：

(1) 以點乘法計算 $Q = \alpha \bullet W$。

(2) 以點乘法計算 $C_1 = \alpha \bullet P$。

(3) 輸出 Q 和 C_1，Q 由傳送者保留，只需要傳送 C_1 給接收者。

2.3.3 DP-PSEC

輸入參數：

- PK：PSEC-KEM 的 Public Key。
- C_1：橢圓曲線上的一點，實質上視為已加密的訊息。
- s：PSEC-KEM 的 Private Key

輸出的加密訊息是下列兩者的結合：

- Q'：解密得到的結果應與 Q 相同，即為金鑰 K。

演算步驟：

(1) 以點乘法計算 $Q' = s \bullet C_1$。

(2) 輸出 $Q'(K)$。

2.4 PSEC-KEM 的安全性

PSEC-KEM 安全性是基於計算 Diffie-Hellman 難題(The computational Diffie-Hellman problem)。首先簡要說明橢圓曲線金鑰交換(Elliptic Curve Diffie-Hellman, ECDH)協定(如圖 2 所示)[1]：

(1) 尋找雙方同意之系統橢圓曲線參數與基點 G 作為系統的 Public Key，G 的級數 n 要夠大(例如大於 2^{160})。

(2) 使用者 A 選擇 Private Key u，$1 < u < n$，且計算 $G_u = u \bullet G$，之後 G_u 將傳給欲通信的 B。G_u 為 A 之 Private Key u 對應的 Public Key。

(3) 使用者 B 選擇 Private Key v，$1 < v < n$，且計算 $G_v = v \bullet G$，之後 G_v 將傳給欲通信的 A。G_v 為 B 之 Private Key v 對應的 Public Key。

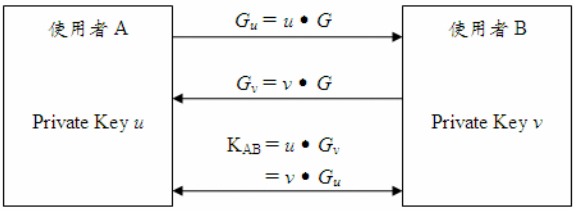

圖 2 橢圓曲線金鑰交換(ECDH)協定

使用者 A 的 Public Key G_u 與 Private Key u 及使用者 B 的 Public Key G_v 與 Private Key v 之安全性便受到離散對數難題的保護。A 與 B 所交換的秘密金鑰為 $K_{AB} = u \bullet G_v = v \bullet G_u$，計算 Diffie-Hellman 難題在此而言的含意即為，已知 G、G_v 及 G_u，但要計算出 $u \bullet G_v$ 或 $v \bullet G_u$ 是非常困難的[4]。

2.5 KEM 與 PKE 之比較

傳統於混合加密模式中，用於加密保護遞送對稱式加密演算法金鑰的方法稱為 PKE (Public Key Encryption)。PKE 加密時需以訊息接收者的 Public Key 對一組明文 M(或金鑰 K)進行加密，輸出為密文 C，密文 C 本身的意義就等同於金鑰 K，因此表示為 $C\{K\}$；KEM 加密則只需要指定訊息接收者的 Public Key，不需要給予明文 M(或金鑰 K)，加密時系統會自動加入一組亂數 R 供運算用，加密後會分別產生金鑰 K 及密文 C，換言之，KEM 加密後所產生的密文 C 在意義上並不等同於金鑰 K，如圖 3 所示[2]。

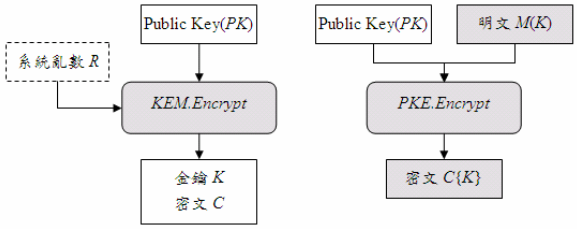

圖 3 KEM 與 PKE 的加密流程比較

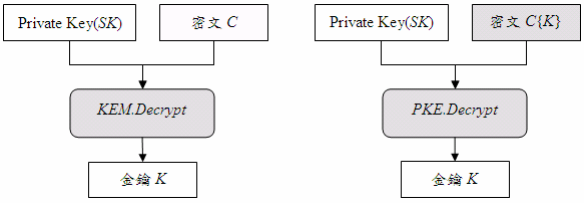

圖 4 KEM 與 PKE 的解密流程比較

加密後傳送時，KEM 機制只需傳送密文 C，接收者便能以其 Private Key 經由 KEM 機制的解密模組解出金鑰 K。在傳送者端所計算出來的 K 值是不必傳送予接收者的。KEM 與 PKE 兩者的解密流程則比較如圖 4。

3. 軟體實作

3.1 軟體設計架構

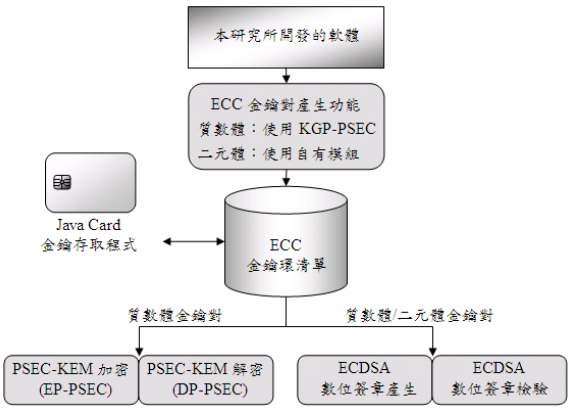

圖 5 本研究所開發的軟體功能架構

圖 5 是本研究所開發的軟體功能架構圖。目前實作的功能有 ECC 金鑰對產生、PSEC-KEM 的 Public Key 加解密、ECDSA 數位簽章產生與檢驗、Java Card 存取 ECC 金鑰等。在 ECC 金鑰參數方面，參考選用 NIST 的 FIPS 186-2[7]標準所建議的 5 種質數體及 10 種二元體，總共 15 種金鑰參數可供使用者選擇。

3.2 軟體使用簡介

圖 6 是本研究所開發的軟體主畫面。軟體開啟時，便會偵測出已存放於電腦中的金鑰並顯示到主畫面的金鑰環清單；假如使用者存取 Java Card，亦會將卡片中的金鑰更新顯示於金鑰環清單。Java Card 可存放的金鑰包括使用者本身的 ECC 金鑰對及通訊對方的公開金鑰。

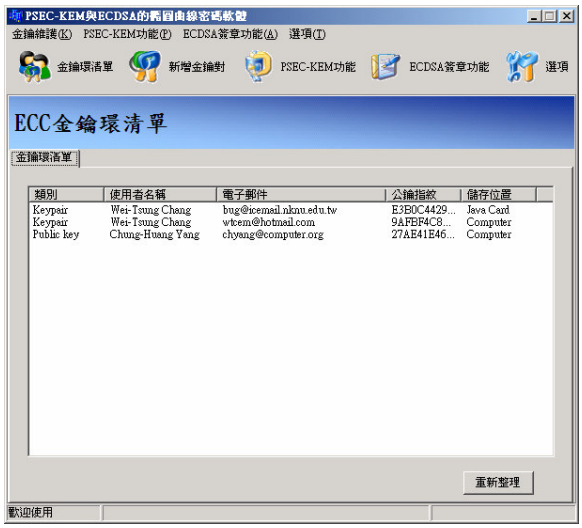

圖 6 軟體主畫面

圖 7 新增 ECC 金鑰對

圖 7 是新增金鑰對的畫面，提供質數體及二元體等 15 種橢圓曲線金鑰參數讓使用者選擇。在新增金鑰之前，使用者必須輸入名稱及電子郵件位址以供金鑰存取識別。另外尚須輸入存取密碼(Passphrase)以保護 Private Key，Private Key 以 AES-128 演算法來加密保護。

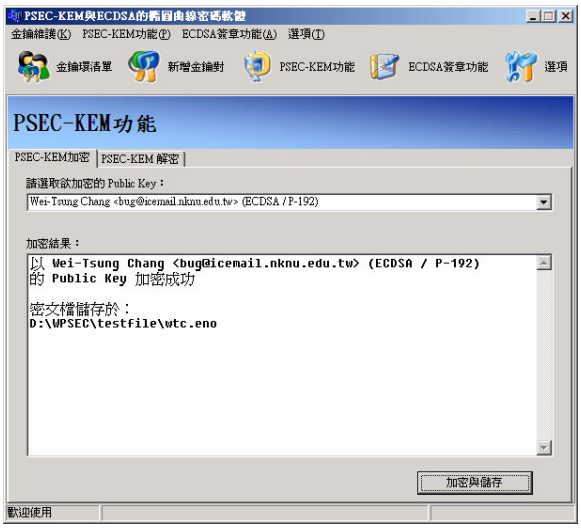

圖 8 PSEC-KEM 加密

圖 8 是 PSEC-KEM 的加密功能。使用者僅需選取 Public key，再執行[加密與儲存]按鈕，便可以 Public Key 將一組系統亂數經加密運算後得出金鑰 K 及密文檔 C 並儲存於電腦中，以供日後金鑰交換所需。若要解密，則是選取金鑰對及已加密過的密文檔便可，如圖 9 所示。

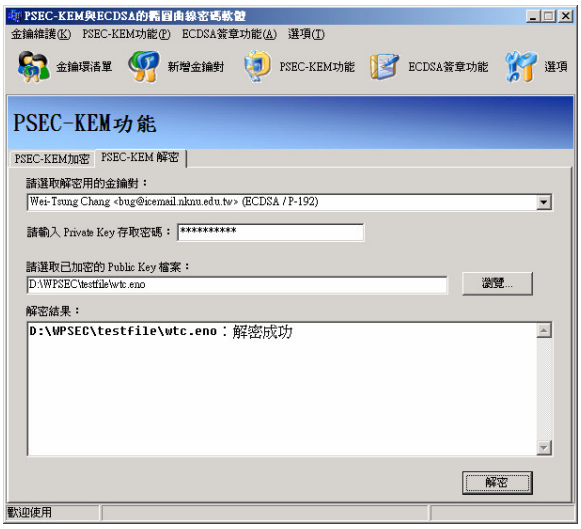

圖 9 PSEC-KEM 解密

圖 10 是 ECDSA 數位簽章產生畫面，使用者瀏覽選取準備要簽署的檔案，並指定要使用的 Private Key 之後，便可執行簽署數位簽章的功能。我們以分離式數位簽章的方式，在原始檔案之外另行儲存一個簽章檔，為與一般簽章檔".sig"區別，副檔名則定為".ecsig"。對於數位簽章的檢驗則選取要檢驗的".ecsig"簽章檔即可，如圖 11 所示。

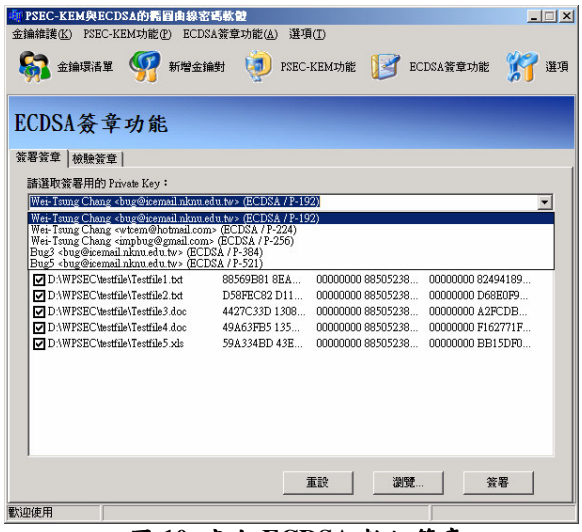

圖 10 產生 ECDSA 數位簽章

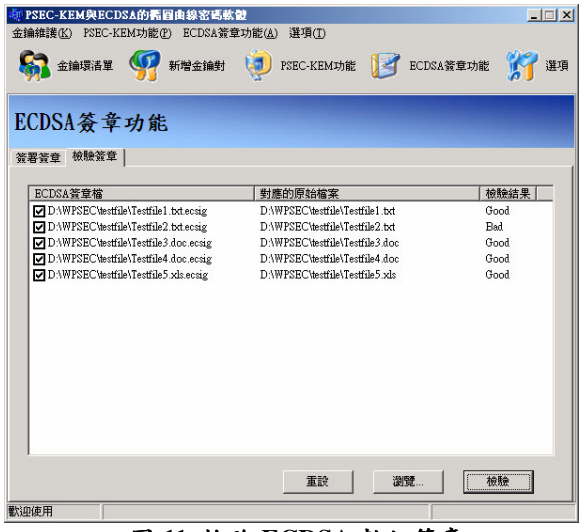

圖 11 檢驗 ECDSA 數位簽章

4. 結論

在本研究中，我們不僅將 PSEC-KEM 的金鑰交換相關功能實作為視窗軟體，亦利用其金鑰對產生功能整合自有的 ECDSA 程式模組開發質數體數位簽章功能，並結合智慧卡的使用。在以 PSEC-KEM 進行金鑰交換時，輔以 ECDSA 數位簽章的應用，進一步可確保 PSEC-KEM 密文檔的完整性及不可否認性。在數位簽章的功能方面，我們依據 FIPS 186-2 標準所建議的橢圓曲線參數，總共實作 15 種不同金鑰參數長度的 ECDSA 數位簽章演算法。

根據對 ECC 相關文獻的探討，我們知道在相同的安全性之下，ECC 所需的金鑰長度遠比 RSA 金鑰來得小，不僅如此，ECC 還有更好的執行效率。因此在可預見的未來，ECC 的技術勢將成為公開金鑰密碼系統的主流。未來的研究，可以考慮將本軟體擴充整合 DEM 機制及相關的實作模組開發成一個具有完整混合加密模式功能的軟體。或者將本軟體的 PSEC-KEM 及 ECDSA 功能模組整合到其他基於 RSA/DSA 演算法的 PKI 網路安全系統，如電子商務、電子郵件或即時通訊服務等，以效率與安全性俱為較佳的橢圓曲線演算法改善並提升安全服務的品質。

致謝

本研究部分之成果承蒙國科會計畫經費補助（NSC 95-3113-P-017-001、NSC 93-2213-E-017-001），特此致謝。

參考文獻

[1] 楊中皇，*網路安全理論與實務*，金禾資訊，2006 年 3 月。

[2] 日立公司，“PSEC-KEM 的安全性評價”（日文），http://www2.nict.go.jp/jt/a124/cryptrec_publicity/rep_ID0017.pdf，2002 年 11 月。

[3] CRYPTREC(Cryptography Research and Evaluation Committees) Project, http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html.

[4] D. Galindo, S. Martin and J.L. Villar, “Evaluating elliptic curve based KEMs in the light of pairings”, http://eprint.iacr.org/2004/084.pdf, 2004.

[5] IETF, RFC4051, “Additional XML Security Uniform Resource Identifiers (URIs)”, http://www.ietf.org/rfc/rfc4051.txt, 2005.

[6] NESSIE(New European Schemes for Signatures, Integrity, and Encryption) Project, https://www.cosic.esat.kuleuven.be/nessie/.

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

[8] NIST, DRAFT Special Publication 800-57, “Recommendation on Key Management”, http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf, 2003.

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

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

[11] NTT corporation, “PSEC-KEM Specification”, http://info.isl.ntt.co.jp/crypt/eng/psec/dl/cryptrec/01espec.pdf, 2001.

[12] N. Koblitz, “Elliptic Curve Cryptosystems”, *Mathematics of Computation*, Vol. 48, No. 177, pp. 203-209, 1987.

[13] RSA Laboratories, “Password-Based Cryptography Standard”, PKCS #5 v2.0, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf, 1999.

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

[15] V. Miller, “Use of Elliptic Curves in Cryptography”, volume 218 of Lecture Notes in Computer Science, pp. 417-426, Berlin, *Springer-Verlag*, 1986.

[16] V. Shoup, “Encryption algorithms – Part2: Asymmetric ciphers,” Final Committee Draft 18033-2, http://shoup.net/iso/std6.pdf, 2004.