網路安全的理論與實務

楊中皇 著

第五章 數位簽章

http://crypto.nknu.edu.tw/textbook/

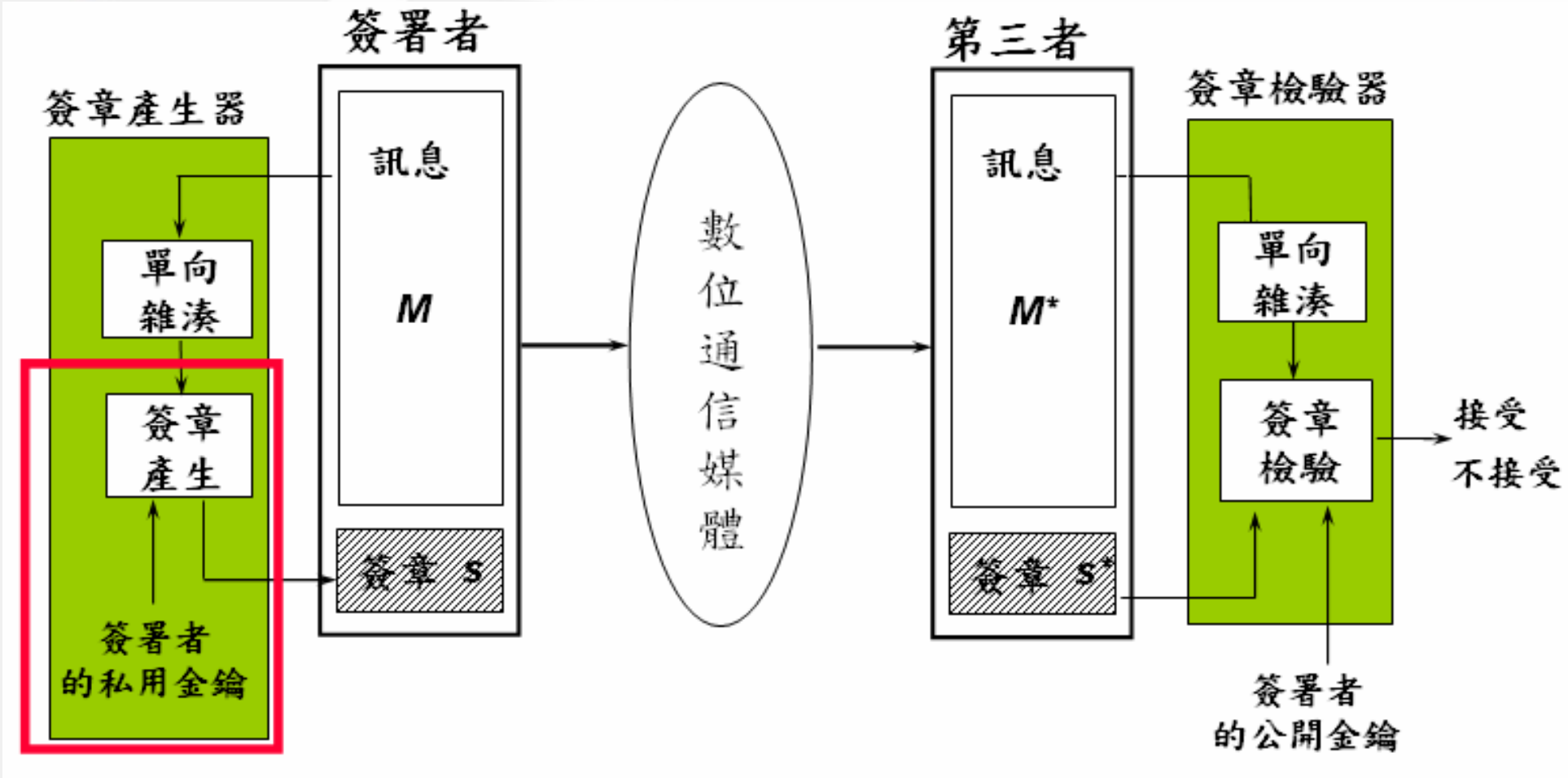
第五章 數位簽章

- 數位簽章概論
- RSA數位簽章演算法
- DSA數位簽章演算法
- ESIGN數位簽章演算

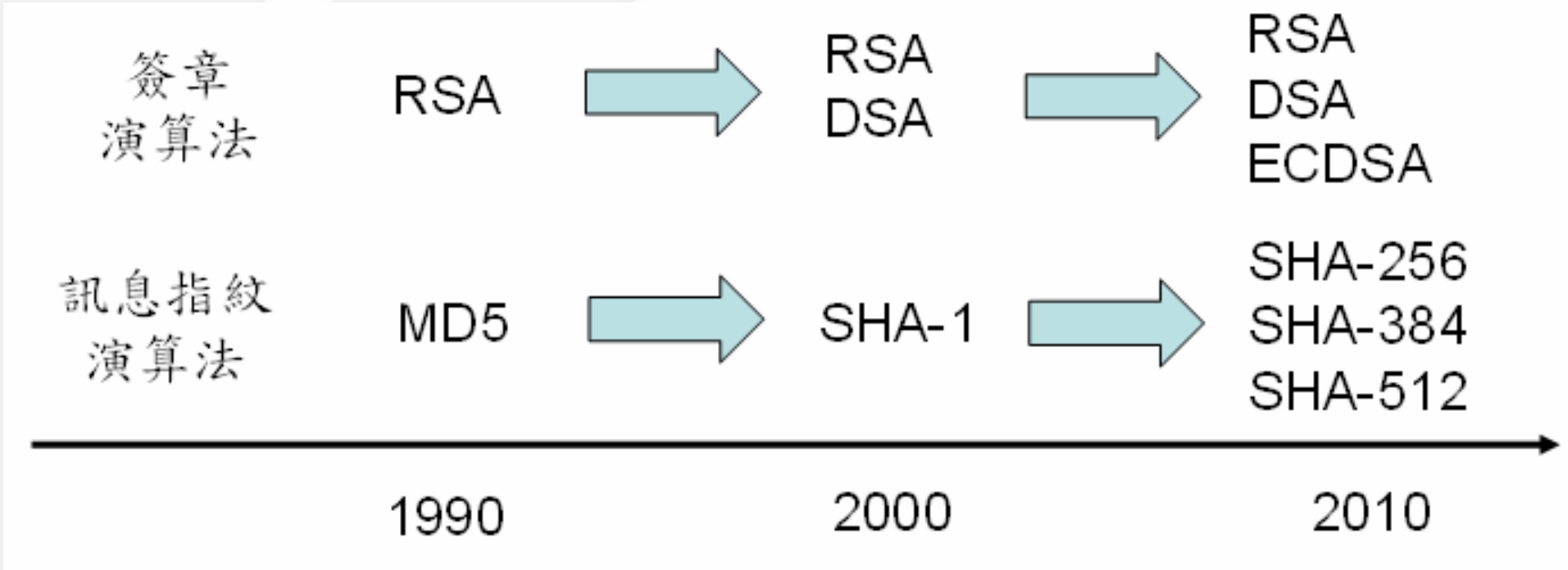
數位簽章演算法

- 數位簽章是目前最重要的數位資料防偽技術
- 電子簽章(electronic signature)可取代傳統的簽名或蓋章，但須確保資料在網路傳輸過程中未被竄改，且能鑑別傳輸者之身分，並防止事後否認傳輸之事實。
- 目前國內網路電子簽章的使用皆是植基於數位簽章(digital signature)的，而數位簽章已廣泛用於電子公文、電子契約、電子支票、軟體防偽、網路報稅等
- 數位簽章採用公開金鑰密碼系統，它將一串數字(例如300位數字) 依附於要保護的資料
- 相同簽署者不同文件時，產生的數位簽章將不同;相同文件不同簽署者時，對應的數位簽章也會不同
- 數位簽章的運作是每個機構或使用者各自擁有成對的公開金鑰與私密金鑰，而簽署者根據自己的私密金鑰製造資料的數位簽章
- 數位簽章檢驗容易，且可由任意第三者根據簽署者的公開金鑰辨認數位章的真偽。
- 數位簽章的產生與檢驗用到兩類演算法：單向雜湊函數演算法與簽章演算法

數位簽章的產生與檢驗


簽署者
第三者
簽章產生器
簽章檢驗器
訊息
M
單向雜湊
簽章產生
簽章 S
簽署者的私用金鑰
數位通信媒體
訊息
M*
簽章 S*
單向雜湊
簽章檢驗
接受
不接受
簽署者的公開金鑰

數位簽章的發展歷程


簽章
演算法
RSA
RSA
DSA
RSA
DSA
ECDSA
訊息指紋
演算法
MD5
SHA-1
SHA-256
SHA-384
SHA-512
1990
2000
2010

RSA數位簽章演算法

- 假設$h(M)$代表訊息M的訊息指紋值，那麼RSA數位簽章的產生與檢驗可簡單描述如下

(1) 產生RSA簽章: 用私密金鑰d與公開金鑰n以下列公式計算簽章s

$$s = h(M)^d \bmod n$$

(2) 檢驗RSA簽章: 將收到的訊息M重新計算訊息指紋值$h(M)$。若且唯若下列公式成立則接受簽章，否則拒絕簽章

$$h(M) = s^e \bmod n$$

OpenSSL進行RSA數位簽章的產生

- 假設私密金鑰儲存於檔案”rsa_privatekey.pem”，公開金鑰儲存於檔案”rsa_publickey.pem”。假設欲簽署的訊息放於檔案”doc.txt”，而我們希望使用單向雜湊函數SHA-256計算訊息摘要，再用RSA私密金鑰檔”rsa_privatekey.pem”計算數位簽章，產生的SHA256/RSA數位簽章放在”doc.sig” 檔，那麼可用下列指令:

```
openssl dgst -sha256 -sign rsa_privatekey.pem -out doc.sig doc.txt
```

OpenSSL進行RSA數位簽章的檢驗

- 任意的第三者可利用單向雜湊函數與簽署者的公開金鑰進行簽章檢驗判斷訊息與簽章是否吻合。假設訊息檔”doc.txt”與數位簽章檔”doc.sig”且簽署者公開金鑰檔”rsa_publickey.pem”，我們可利用下列指令來進行SHA256/RSA數位簽章的檢驗

  openssl dgst -sha256 -verify rsa_publickey.pem -signature doc.sig doc.txt

OpenSSL執行RSA數位簽章產生與檢驗畫面

- 我們將訊息檔”doc.txt”的內容”This is a test”更改為”That is a test”，再儲存於檔案”doc2.txt”。這會使得數位簽章檔”doc.sig”與訊息檔”doc2.txt”檢驗不吻合

```
C:\openssl\OUT32>sign_rsa

C:\openssl\OUT32>openssl dgst -sha256 -sign rsa_privatekey.pem -out doc.sig doc.txt
Enter pass phrase for rsa_privatekey.pem:

C:\openssl\OUT32>openssl dgst -sha256 -verify rsa_publickey.pem -signature doc.sig doc.txt

Verified OK

C:\openssl\OUT32>openssl dgst -sha256 -verify rsa_publickey.pem -signature doc.sig doc2.txt

Verification Failure

C:\openssl\OUT32>type doc.txt
This is a text.

C:\openssl\OUT32>type doc2.txt
That is a text.
```

DSA (Digital Signature Algorithm)演算法

- 1994年美國正式公佈數位簽章標準(Digital Signature Standard，DSS)
- 無法用來加解密或進行金鑰交換
- 相同金鑰長度進行數位簽章產生時，DSA比RSA快速

DSA金鑰產生

1. 系統首先決定一個固定為160位元的質數q及512位元到1024位元的質數p，但p-1必須是q的倍數。質數p長度需為64位元的倍數，意即長度為512/576/640/…或1024位元
2. 系統再任意找個介於2與p-1的正整數t，$2 \le t \le p\text{-}1$，滿足

$$t^{\frac{p-1}{q}} \bmod p \neq 1$$

生成數(generator) g為 $g = t^{\frac{p-1}{q}} \bmod p$

3. 每個簽署者挑選介於1與q-1的亂數x，$1 \le x \le q\text{-}1$，當做私密金鑰。再用g與x計算對應的公開金鑰y

$$y = g^x \bmod p$$

4. 上述步驟1與步驟2的正整數p, q, 與g為系統公開參數，可由多個簽署者共用當做公開金鑰。每個簽署者的個別私密金鑰為x，公開金鑰為y。

DSA簽章 (*r*, *s*)產生

1. 產生介於1與q-1的亂數k，$1 \le k \le q$-1。
2. 計算

$$r = (g^k \bmod p) \bmod q。$$

3. 如果 $r = 0$, 則回到步驟1。
4. 計算

$$s = k^{-1} \{h(M) + x\, r\} \bmod q$$

5. 如果 $s = 0$, 則回到步驟1。

DSA簽章檢驗

收到訊息M與數位簽章(r, s)，DSA檢驗的步驟：

- 檢驗r, s的範圍是否符合$1 \le r \le q-1$且$1 \le s \le q-1$，否則拒絕簽章。
- 計算

$$w = s^{-1} \bmod q \text{ 及 } h(M)$$

- 計算

$$u_1 = h(M)\, w \bmod q$$

及

$$u_2 = (r\, w) \bmod q$$

- 計算

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

若且唯若 $v = r$ 接受數位簽章，否則拒絕簽章

DSA簽章檢驗$v = r$的理由

- 因為 $s \equiv k^{-1} \{h(M) + x\, r\} \bmod q$

  所以$k\, s \equiv \{h(M) + x\, r\} \bmod q$

- $\{h(M) + x\, r\} w \bmod q \equiv k\, s\, w \bmod q \equiv k\, s\, s^{-1} \bmod q \equiv k$

- 由於公開金鑰$y = g^x \bmod p$ ，我們有

$$v = (\quad g^{u_1} g^{x u_2} \qquad \bmod p) \bmod q$$

$$v = (\quad g^{h(M)w} g^{x r w} \quad \bmod p) \bmod q = (\quad g^{(h(M)+x r)w} \quad \bmod p) \bmod q$$

$$v = (\quad g^{k} \qquad \bmod p) \bmod q = r$$

- 所以如果簽章$(r,\ s)$與訊息M都沒被修改，那麼計算出來的v 與r兩數應該相等。

OpenSSL進行DSA金鑰對的建立

- 首先我們需進行DSA金鑰對的建立。假設指定1024位元公開金鑰(*p*)並產生系統參數檔 ”dsa_prame.pem”：

  openssl dsaparam -out dsa_param.pem 1024

- 從系統參數檔”dsa_prame.pem”來產生私密金鑰檔案”dsa_privatekey.pem”，然後使用AES-128演算法加密保護：

  openssl gendsa -out dsa_privatekey.pem -aes128 dsa_param.pem

- DSA公開金鑰檔需從私密金鑰檔轉換而來，我們可利用下列指令產生對應的公開金鑰”dsa_publickey.pem”

  openssl dsa -in dsa_privatekey.pem -pubout -out dsa_publickey.pem

OpenSSL執行DSA金鑰產生的畫面


```
C:\openssl\OUT32>openssl dsaparam -out dsa_param.pem 1024
Loading 'screen' into random state - done
Generating DSA parameters, 1024 bit long prime
This could take some time
.............+...+..............+..+............+.+.....+..............+.....+....
....+......+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++*
.........+..+.......+....+.................................+.....+....+...+.+..+...+....
.......+.............+....+.......+...+..+.....................+........+.....+.....
......+.........................+.+...+..+.............................+..+............
..+.+.....+................+......+...+................+.............+.+.............
.+..+...+.......+...+........+...+.........................+.+.........................+...+.
............+............+......+............+...+.+..........+..............+..+.
..+...+.................................+........+.....+.......+......+..+....+....+.+..
.........................+...+...+.....+.....................................+.....+........
...+.......+..............+.....+.....+....+.................+..............+.........
.............+........+.+...+.+............+........+...+......+.....................+
.+.........+...+..........+.....................+....+...................+...........+...
+..+.........+...............+.............+.+....+.......+..........+...............+..
..........+..+.+...+....+................+........+.......+...........+...........++++++++
++++++++++++++++++++++++++++++++++++++++*

C:\openssl\OUT32>openssl gendsa -out dsa_privatekey.pem -aes128 dsa_param.pem
Loading 'screen' into random state - done
Generating DSA key, 1024 bits
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

C:\openssl\OUT32>openssl dsa -in dsa_privatekey.pem -pubout -out dsa_publickey.pem
read DSA key
Enter PEM pass phrase:
writing DSA key
```

OpenSSL進行DSA數位簽章的產生與檢驗

- 假設欲簽署的訊息放於檔案”doc.txt”，而我們希望使用單向雜湊函數SHA-1計算訊息摘要，再用DSA私密金鑰檔”dsa_privatekey.pem”計算數位簽章，產生的SHA1/DSA數位簽章放在”doc.sig” 檔:

  openssl dgst -dss1 -sign dsa_privatekey.pem -out doc.sig doc.txt

- 有了訊息檔及簽章檔，任意的第三者可利用單向雜湊函數與簽署者的DSA公開金鑰判斷訊息與簽章是否吻合。假設訊息檔”doc.txt”與數位簽章檔”doc.sig”且簽署者公開金鑰檔”dsa_publickey.pem”，我們可進行SHA256/DSA數位簽章的檢驗:

  openssl dgst -dss1 -verify dsa_publickey.pem -signature doc.sig doc.txt

OpenSSL執行DSA數位簽章產生與檢驗畫面

- 我們將訊息檔”doc.txt”的內容”This is a test”更改為”That is a test”，再儲存於檔案”doc2.txt”。這會使得數位簽章檔”doc.sig”與訊息檔”doc2.txt”檢驗不吻合

```
C:\openssl\OUT32>openssl dgst -dss1 -sign dsa_privatekey.pem -out doc.sig doc.txt
Enter pass phrase for dsa_privatekey.pem:

C:\openssl\OUT32>openssl dgst -dss1 -verify dsa_publickey.pem -signature doc.sig doc.txt
Verified OK

C:\openssl\OUT32>openssl dgst -dss1 -verify dsa_publickey.pem -signature doc.sig doc2.txt
Verification Failure

C:\openssl\OUT32>type doc.txt
This is a text.

C:\openssl\OUT32>type doc2.txt
That is a text.
```

OpenSSL測試DSA數位簽章效率

- openssl speed dsa512 dsa1024

```
C:\openssl\OUT32>openssl speed dsa512 dsa1024
To get the most accurate results, try to run this
program when this computer is idle.
First we calculate the approximate speed ...
Doing 10485 512 bit sign dsa's: 10485 512 bit DSA signs in 33.75s
Doing 5242 512 bit verify dsa's: 5242 512 bit DSA verify in 20.96s
Doing 2621 1024 bit sign dsa's: 2621 1024 bit DSA signs in 27.17s
Doing 1310 1024 bit verify dsa's: 1310 1024 bit DSA verify in 16.91s
OpenSSL 0.9.8 05 Jul 2005
built on: Mon Sep 19 08:36:38 2005
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,cisc,4,long) aes(partial) idea(i
nt) blowfish(idx)
compiler: bcc32 -DWIN32_LEAN_AND_MEAN -q -w-aus -w-par -w-inl  -c -tWC -tWM -DOP
ENSSL_SYSNAME_WIN32 -DL_ENDIAN -DDSO_WIN32 -D_stricmp=stricmp -O2 -ff -fp -DOPEN
SSL_NO_RC5 -DOPENSSL_NO_MDC2 -DOPENSSL_NO_KRB5
available timing options: TIMEB HZ=1000
timing function used: ftime
                  sign    verify    sign/s verify/s
dsa  512 bits 0.003219s 0.003998s    310.7    250.1
dsa 1024 bits 0.010366s 0.012905s     96.5     77.5

C:\openssl\OUT32>
```

ESIGN (Efficient digital SIGNature)演算法

- ESIGN (Efficient digital SIGNature，電子印鑑)
- 日本電話公司(NTT)岡本龍明(Tatsuaki Okamoto)院士1990年發明的一種快速數位簽章演算法
- 已被制訂於國際標準ISO/IEC14888-3、IEEE P1363a、RFC 4051等
- 與DSA相同無法用來加解密或進行金鑰交換，簽章屬於隨機數位簽章(randomized digital signature)，亦即同一訊息有多個有效的數位簽章
- ESIGN與DSA都能進行事先計算(pre-computation)，也就是說數位簽章產生的計算裡最耗時間的模指數(modular exponentiation)運算與訊息*M*無關，僅與亂數*k*及公開金鑰有關

ESIGN金鑰產生

- ESIGN金鑰產生方式爲簽署者首先自行挑選兩個大質數，p和q
- 簽署者公開金鑰$n = p^2q$，私密金鑰與RSA相同爲p與q
- 系統有個安全參數e值至少爲8，建議至少設爲1024
- ESIGN其實有好幾個變形的演算法，有的提供可証明安全(provably secure)的設計。我們這裡介紹TSH-ESIGN (Trisection Size Hash ESIGN)就是基於ESIGN的一個更具安全性的簽章演算法

TSH-ESIGN訊息*M*的簽章*s*產生

1 產生介於1與pq-1的亂數k，$1 \le k \le pq-1$。

如果$k \bmod p = 0$，則回到步驟1。

2 計算 $$w = \left\lceil \frac{h(M) - k^e \bmod n}{pq} \right\rceil$$

其中符號$\lceil x \rceil$代表天花板函數(ceiling function)，意即大於或等於x的最小整數。$h(M)$在此代表訊息M的單向雜湊函數值但長度為$|p|-1$位元，$|p|$為私密金鑰p的位元長度；長度為$3|p|$位元(先擺個零，串接$|p|-1$位元的$h(M)$，再串接$2|p|$個零)。

3 計算 $$y = w\left(e\,k^{e-1}\right)^{-1} \bmod p$$

4 計算簽章s $$s = k + y\,pq$$

TSH-ESIGN的簽章檢驗

- TSH-ESIGN訊息*M*與簽章*s*的檢驗需使用公開金鑰*n*與*e*算出$s^e \bmod n$，再判斷$s^e \bmod n$的|*p*|個最有效的位元(most significant bits)是否與(*0*|| *h(M)*)相同
- 下列公式成立就接受簽章

$$\text{最有效位元}\left(s^e \bmod N\right) = 0 \,||\, h(M)$$

1992年三種數位簽章演算法執行效率比較

- 於**H8/300 IC**卡晶片

數位簽章演算法	512-bit DSA	512-bit RSA	576-bit ESIGN
事先計算	28 秒	不適用	不需要
簽章產生時間	0.05 秒	25 秒	0.45 秒
簽章檢驗時間	56 秒	5 秒	0.27 秒

參考資料

- 經濟部商業司，電子簽章法，http://www.moea.gov.tw/~meco/doc/ndoc/s5_p05.htm
- 日本NTT密碼學演算法，http://info.isl.ntt.co.jp/crypt/
- ISO/IEC 14888-3, Information technology - Security techniques - Digital Signatures with Appendix-Part 3: Certificate-Based Mechanisms, Dec. 1998.
- A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press Series on Discrete Mathematics and Its Applications, 1996.
- NIST, Digital Signature Standard (DSS), FIPS 186-2, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, (2000)
- Tatsuaki Okamoto, "A Fast Signature Scheme Based on Congruential Polynomial Operations", *IEEE Trans. Info. Theory*, Vol. 36, pp. 47-53, January 1990.
- Tatsuaki Okamoto and Eiichiro fujisaki, “On Comparison of Practical Digital Signature Schemes,” *NTT Review*, Vol. 5, No. 1, pp. 75-81, January 1993.