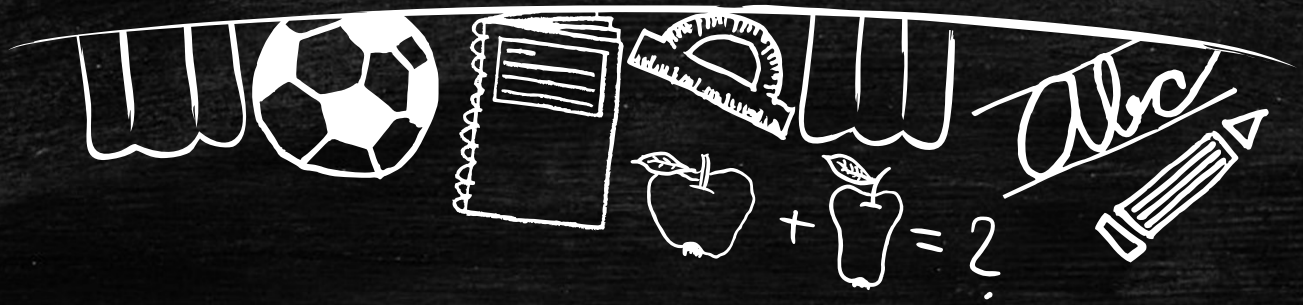
RSA Optimization
based on
Chinese Remainder Theorem



Review RSA: Choosing keys

1. Choose two large prime numbers p, q.
   (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime").

4. Choose d such that $ed-1$ is exactly divisible by z.
   (in other words: ed mod $z = 1$).

5. *Public* key is (n,e). *Private* key is (n,d).

Review RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

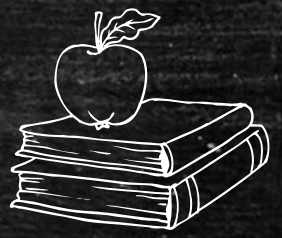
   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Review RSA: Problem

Equation like $m = c^d \bmod n$ takes too much time to implement since d can be a large number.

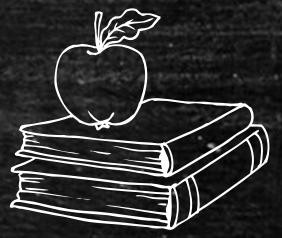
For efficiency, many popular crypto libraries use the following optimization for decryption and signing based on the Chinese remainder theorem.

RSA Optimization



- The following values are precomputed and stored as part of the private key:
- p and q : the primes from the key generation
- $d_P = d \pmod{p - 1}$
- $d_Q = d \pmod{q - 1}$
- $q_{inv} = q^{-1} \pmod{p}$

RSA Optimization



- These values allow the recipient to compute the exponentiation $m = c^d \pmod{pq}$ more efficiently as follows:
- $m_1 = c^{dp} \pmod{p}$
- $m_2 = c^{dQ} \pmod{q}$
- $h = q_{inv} (m_1 - m_2) \pmod{p}$
- (If $m_1 < m_2$ then some libraries compute h as

$$q_{inv} \left(m_1 + \left\lceil \frac{q}{p} \right\rceil p - m_2\right) \pmod{p})$$

- $m = m_2 + hq \pmod{pq}$

RSA Optimization

- https://en.wikipedia.org/wiki/RSA_(cryptosystem)

Security and practical considerations [edit]

Using the Chinese remainder algorithm [edit]

For efficiency many popular crypto libraries (like OpenSSL, Java and .NET) use the following optimization for decryption and signing based on th theorem. The following values are precomputed and stored as part of the private key:

- p and q: the primes from the key generation,
- $d_P = d \pmod{p-1}$,
- $d_Q = d \pmod{q-1}$ and
- $q_{\text{inv}} = q^{-1} \pmod{p}$.

These values allow the recipient to compute the exponentiation $m = c^d \pmod{pq}$ more efficiently as follows:

- $m_1 = c^{d_P} \pmod{p}$
- $m_2 = c^{d_Q} \pmod{q}$
- $h = q_{\text{inv}}(m_1 - m_2) \pmod{p}$ (if $m_1 < m_2$ then some libraries compute h as $q_{\text{inv}} \left[\left(m_1 + \left\lceil \frac{q}{p} \right\rceil p \right) - m_2 \right] \pmod{p}$)
- $m = m_2 + hq$

This is more efficient than computing exponentiation by squaring even though two modular exponentiations have to be computed. The reason is that exponentiations both use a smaller exponent and a smaller modulus.

Integer factorization and RSA problem [edit]

RSA Optimization

国际局存档的最新著录事项 永久链接

公布号: WO/1998/052319 国际申请号: PCT/US1998/009593
公布日: 19.11.1998 国际申请日: 12.05.1998
已提出第二章要求: 02.12.1998
IPC: *G06F 7/72* (2006.01), *H04L 9/30* (2006.01)

申请人: **YEDA RESEARCH AND DEVELOPMENT CO. LTD.** [IL/IL]; Weizmann Institute of Science, P.O. Box 95, 76100 Rehovot (IL).
FLEIT, Lois [US/US]; (US) *(UG only)*

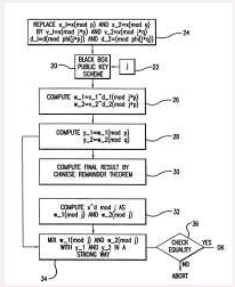
发明人: **SHAMIR, Adi**; (IL)

代理人: **FLEIT, Martin**; Evenson, McKeown, Edwards & Lenahan, PLLC, Suite 700, 1200 G Street, Washington, D.C. 20005 (US)

优先权数据: 08/854,464 12.05.1997 US

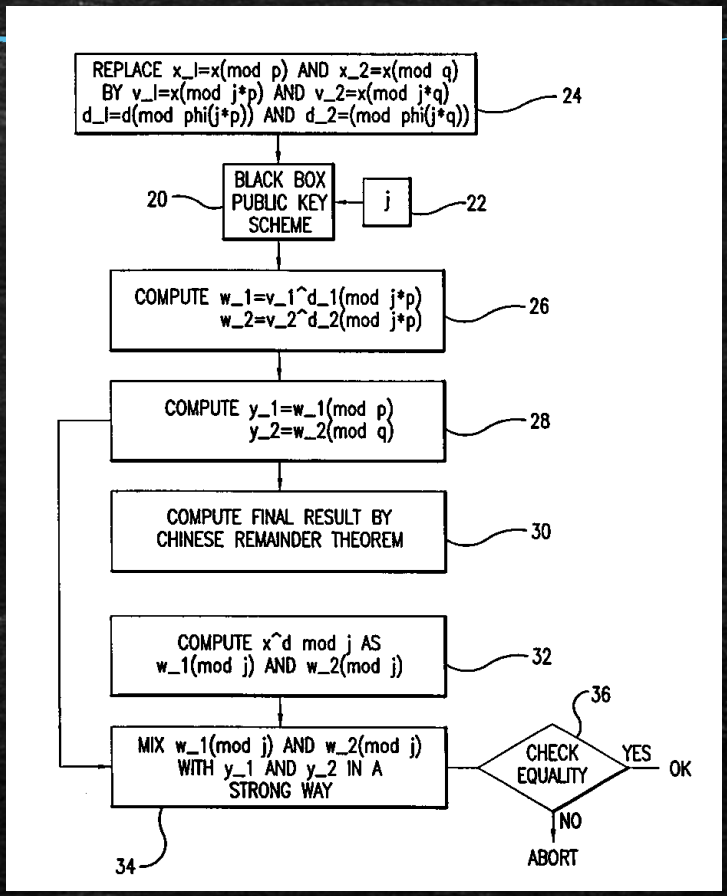
标题 **(EN)** IMPROVED METHOD AND APPARATUS FOR PROTECTING PUBLIC KEY SCHEMES FROM TIMING AND FAULT ATTACKS
(FR) PROCEDE ET DISPOSITIF AMELIORES PERMETTANT DE PROTEGER LES LOGIQUES DE CLES PUBLIQUES CONTRE LES ATTAQUES BASEES SUR LA SEQUENCE DES OPERATIONS ET LES FAUTES

摘要: **(EN)** Improved methods and apparatus for protecting public key schemes based on modular exponentiation (including RSA and Diffie-Hellman) from indirect cryptanalytic techniques such as timing and fault attacks. Known methods for making the implementation of number-theoretic schemes resistant to such attacks typically double their running time, whereas the novel methods and apparatus described in this patent add only negligible overhead. This improvement is particularly significant in smart card and software-based implementations, in which the modular exponentiation operation is quite slow, and doubling its time may be an unacceptable solution.
(FR) Procédé et dispositif améliorés permettant de protéger les logiques de clés publiques, sur la base de l'exponentiation modulaire (RSA et Diffie-Hellman notamment), contre les techniques crypto-analytiques indirectes, telles que les attaques basées sur la séquence des opérations et les fautes. Typiquement, les procédés connus permettant d'implémenter des logiques théoriques de nombres qui résistent à ces attaques doublent le temps d'exécution, tandis que les nouveaux procédés et dispositifs décrits dans ce brevet n'ajoutent qu'une surcharge système négligeable. Cette amélioration est particulièrement significative pour les cartes à puce et les implémentations reposant sur un logiciel, dans lesquelles l'opération d'exponentation modulaire est assez lente et où le doublement de leur durée risque d'être une solution inacceptable.



指定国家: AL, AM, AU, AZ, BA, BB, BG, BR, BY, CA, CN, CU, CZ, EE, FI, GE, GH, GM, GW, HU, ID, IL, IS, JP, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LV, MD, MG, MK, MN, MX, NO, NZ, PL, RO, RU, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UZ, VN, YU, ZW.
African Regional Intellectual Property Organization (GH, GM, KE, LS, MW, SD, SZ, UG, ZW)
European Patent Office (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE)
African Intellectual Property Organization (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

公布语言: English (EN)
申请语言: English (EN)



RSA Optimization

Ronald **R**ivest

Adi **S**hamir

Leonard **A**dleman

RSA Optimization Explanation

- $\Phi(p) = p - 1$ $\qquad c^{p-1} = 1 \pmod{p}$
- $d_p = d \pmod{p - 1}$ $\qquad d_p + k_1 (p - 1) = d$
- $m_1 = c^{dp} = c^d = c^{dp + k1(p - 1)} \pmod{p}$

- $\Phi(q) = q - 1$ $\qquad c^{q-1} = 1 \pmod{q}$
- $d_Q = d \pmod{q - 1}$ $\qquad d_Q + k_2 (q - 1) = d$
- $m_2 = c^{dQ} = c^d = c^{dQ + k2 (q - 1)} \pmod{q}$

RSA Optimization Explanation

$$\begin{cases} c^d = m_1 \pmod p \\ c^d = m_2 \pmod q \end{cases} \quad \text{similar to} \quad \begin{cases} x = a_1 \pmod{p_1} \\ x = a_2 \pmod{p_2} \end{cases}$$

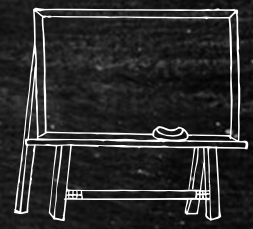
p and q are two prime number, so p and q are coprime

$c^d = (m^e)^d = m = x \pmod{pq}$

Review Chinese Remainder Theorem

- Suppose $m_1, \ldots, m_k$ are positive integers that are pairwise coprime. Then, for any given sequence of integers $a_1, \ldots, a_k$, there exists one integer x module m $= m_1 m_2 \ldots\ldots m_k$ solving the following system of simultaneous congruences.
- $x = a_1 \pmod{m_1}$ …… $x = a_k \pmod{m_k}$
- Provided that $M_i = m_1 \ldots\ldots m_{i-1} m_{i+1} \ldots\ldots m_k$, and M_i' is the inverse element of M_i module m_i.
- $x = M_1 M_1' a_1 + M_2 M_2' a_2 + \ldots\ldots + M_k M_k' a_k \pmod{m}$

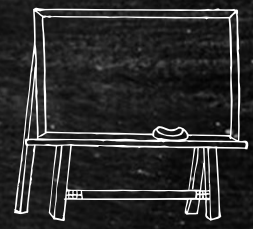
RSA Optimization Explanation



- $M_1 = q$ $\quad\quad$ $M_2 = p$
- $M_1' = q^{-1} \pmod{p}$ $\quad$ $M_2' = p^{-1} \pmod{q}$

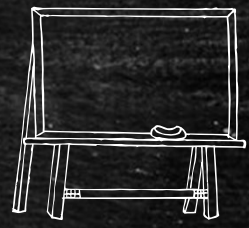
- $m_1 = c^{dP} \pmod{p}$
- $m_2 = c^{dQ} \pmod{q}$
- $h = q_{inv} (m_1 - m_2) \pmod{p}$
- $m = m_2 + hq \pmod{pq}$

RSA Optimization Explanation



$$
\begin{aligned}
m &= m_2 + hq && \pmod{pq} \\
&= m_2 + q_{inv}\,(m_1 - m_2) \pmod{p}\; q && \pmod{pq} \\
&= m_2 + q_{inv}\,(m_1 - m_2)\, q && \pmod{pq} \\
&= q_{inv}\, q\, m_1 + (1 - q_{inv}\, q)\, m_2 && \pmod{pq}
\end{aligned}
$$

RSA Optimization Explanation



$q = M_1$

$q_{inv} = q^{-1} \pmod{p} = M_1'$

$q_{inv} = q^{-1} \pmod{p}$

$q_{inv}\, q = 1 \pmod{p}$

$q_{inv}\, q = 1 + tp$

$1 - q_{inv}\, q = 1 - 1 - tp = -tp$

q_{inv} is an integer

$-tp = 1 \pmod{q}$

$-t = p^{-1} \pmod{q}$

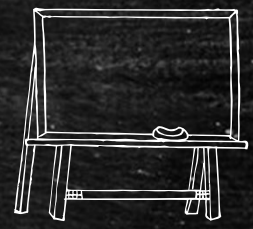
$M_2 = p$

$M_2' = -t$

$M_1\, M_1' = q_{inv}\, q$

$M_2\, M_2' = -tp = 1 - q_{inv}\, q$

RSA Optimization Explanation



According to Chinese Remainder Theorem

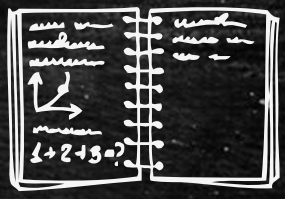
$$x = M_1 M_1' a_1 + M_2 M_2' a_2 \pmod{pq}$$

$$= q_{inv}\, q\, m_1 + (1 - q_{inv}\, q)\, m_2 \pmod{pq}$$

$$= m_2 + hq \pmod{pq}$$

$$= m$$

RSA Optimization Explanation



If $m_1 < m_2$ then some libraries compute h as

$$q_{inv} (m_1 + \left\lceil \frac{q}{p} \right\rceil p - m_2) \pmod p$$

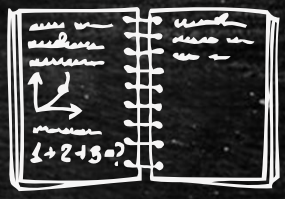
$$q_{inv} (m_1 + \left\lceil \frac{q}{p} \right\rceil p - m_2) \pmod p$$

$$= q_{inv} (m_1 - m_2) + q_{inv} \left\lceil \frac{q}{p} \right\rceil p \pmod p$$

$q_{inv}, \left\lceil \frac{q}{p} \right\rceil$ are all integers

$$= q_{inv} (m_1 - m_2) \pmod p$$

RSA Optimization Explanation



$$q_{inv} (m_1 + \left\lceil \frac{q}{p} \right\rceil p - m_2) >= q_{inv} (m_1 + q/p * p - m_2)$$

$$m_1 + q/p * p - m_2 = m_1 + q - m_2$$

$$m_2 = c^d \pmod q \dashrightarrow m_2 <= q$$

$$m_1 + q - m_2 > 0 \qquad q_{inv} > 0$$

$$q_{inv} (m_1 + \left\lceil \frac{q}{p} \right\rceil p - m_2) > 0$$

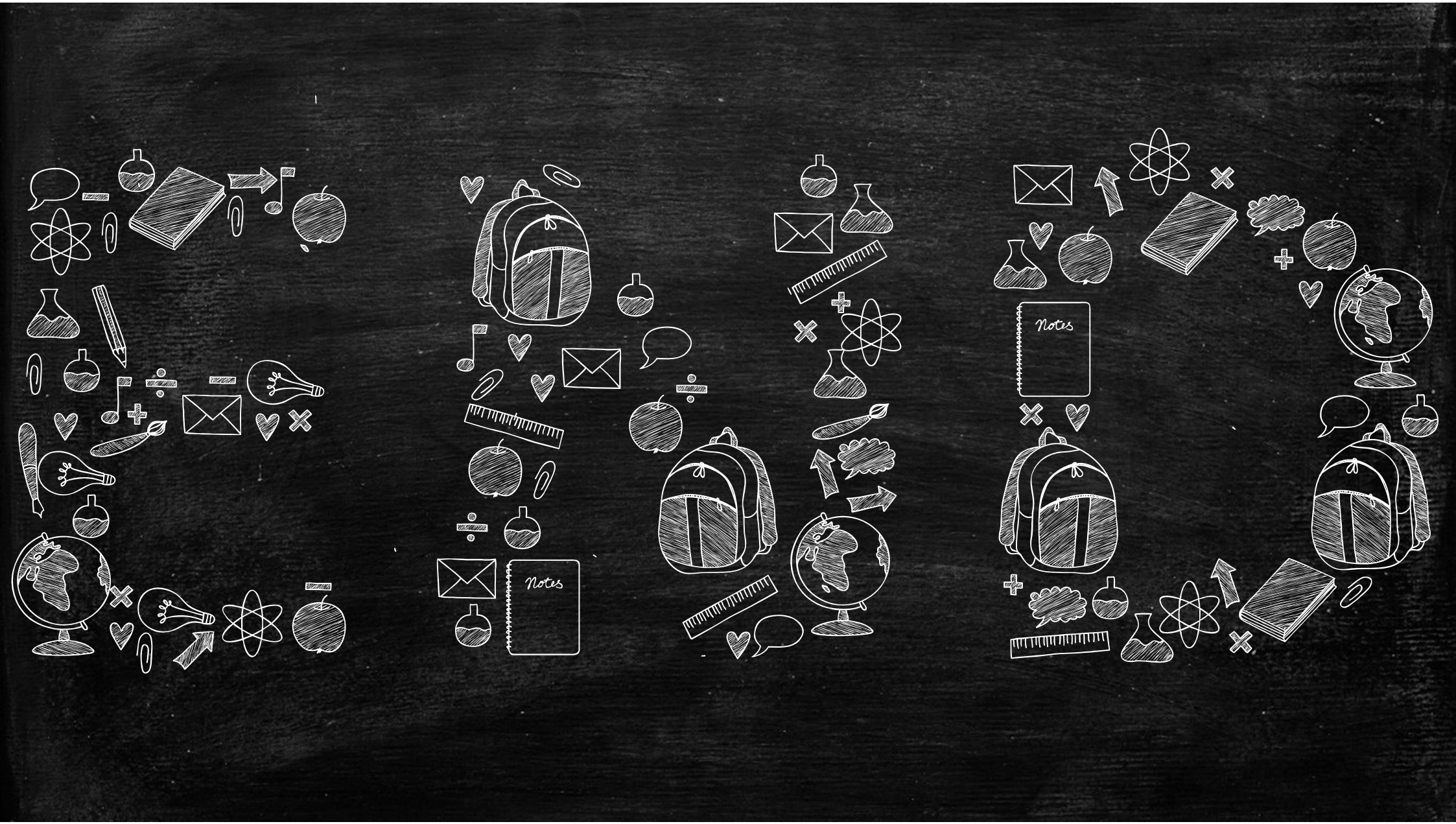
Conclusion

This is a very efficient way to implement, even more efficient than computing exponentiation by squaring even though two modular exponentiations have to be computed.

The reason is that these two modular exponentiations both use a smaller exponent and a smaller modulus.

CRT Applications

- Sequence numbering
- Fast Fourier transform
- Encryption
- Range ambiguity resolution
- Hermite interpolation
- Dedekind's theorem


Notes
Notes