RSA Hardware Research Ideas

RSA Public Key Cryptosystem

- Developed in 1978, by Rivest, Shamir & Adleman
- Its security is based on the *integer factoring problem*
- The most popular method :-
  - simple to understand & implement
  - same algorithm for encryption & decryption
  - can also be used for digital signature

RSA Security

* Security depends on the key size.

larger key size → more secure system

RSA Implementations

software
slow speed
Can be less secure

hardware

Modular Exponentiation
repeated squaring

&

Modular Multiplication
- multiply/divide
- add/subtract
- logarithmic speed
- *Montgomery*

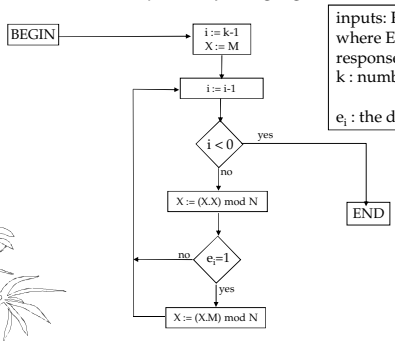
RSA Hardware Approaches 1990's

RSA Designs

Modular Arithmetic Designs

Modular Exponentiation

repeated squaring algorithm



Modular Exponentiation Example

repeated squaring algorithm

- Compute: 3^9 mod 7
- k = 9 = $(1001)_2$; i = 3 ; X = M = 3
  - i = 2 ; X = 3.3 = 2; e_2=0⇒X = 2 (no change since e_i = 0)
  - i = 1 ; X = 2.2 = 4; e_1=0⇒X = 4 (no change since e_i = 0)
  - i = 0 ; X = 4.4 = 2; e_0=1⇒X = 2.3 = 6
- 3^9 mod 7 = X = 6

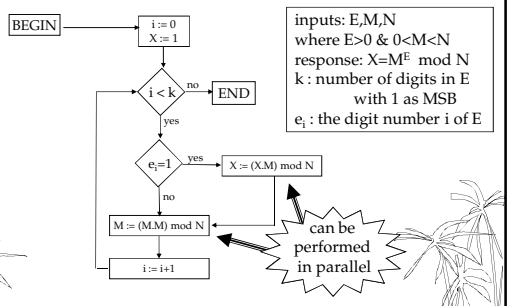
Modular Exponentiation Example

repeated squaring algorithm

- Compute: 7^{27} mod 10 = 3
- k = 27 = $(11011)_2$; i = 5 ; X = M = 7
  - i = 3 ; X = 7.7 = 9; e_3=1⇒X = 7.9 = 3
  - i = 2 ; X = 3.3 = 9; e_2=0⇒X = 9 (no change since e_i = 0)
  - i = 1 ; X = 9.9 = 1; e_1=1⇒X = 1.7 = 7
  - i = 0 ; X = 7.7 = 9; e_0=1⇒X = 9.7 = 3

Modular Exponentiation

improved repeated squaring



Modular Exponentiation Example

improved repeated squaring algorithm

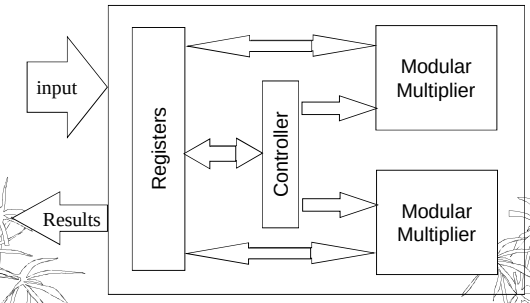
- Compute: 3^9 mod 7
- k = 9 = $(1001)_2$; i = 0→3 ; X = 1; M = 3
  - i = 0 ; e_0=1⇒ X = 1.3 = 3; M = 3.3 = 2
  - i = 1 ; e_1=0⇒X = 3 (no change since e_i = 0) ; M = 2.2=4
  - i = 2 ; e_2=0⇒X = 3 (no change since e_i = 0) ; M = 4.4=2
  - i = 3 ; e_3=1⇒X = 3.2 = 6; M = 2.2=4
- 3^9 mod 7 = X = 6

Modular Exponentiation Example

improved repeated squaring algorithm

- Compute: 7^{10} mod 10 = 9
- k = 10 = $(1010)_2$; i = 0→3 ; X = 1; M = 7
  - i = 0 ; e_0=0⇒X = 1 (no change since e_i = 0) ; M = 7.7 = 9
  - i = 1 ; e_1=1⇒X = 1.9 = 9; M = 9.9 = 1
  - i = 2 ; e_0=0⇒X = 9 (no change since e_i = 0) ; M = 1.1 = 1
  - i = 3 ; e_0=1⇒X = 9.1 = 9; M = 1.1 = 1

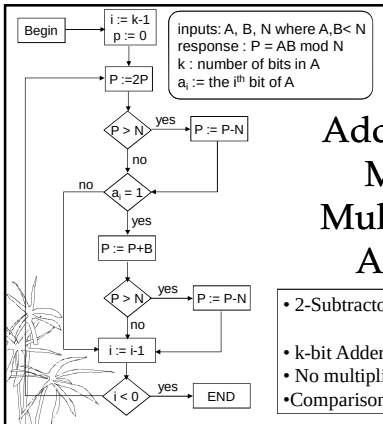
Modular Exponentiation Hardware



Modular Multiplication Implementations

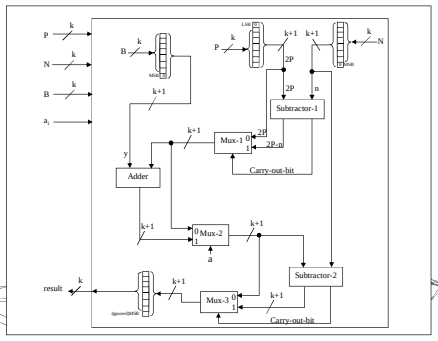
- Add/Subtract Modular Multiplication (Orton)
- Expandable Montgomery Modular Multiplication (Adnan Gutub)
- Merged Montgomery Modular Multiplication (C. K. Koc)

Add/Subtract Modular Multiplication Algorithm



- 2-Subtractors & Adder
  => Max. speed
- k-bit Adder
- No multiplier
- Comparison => sign-bit (subtractor)

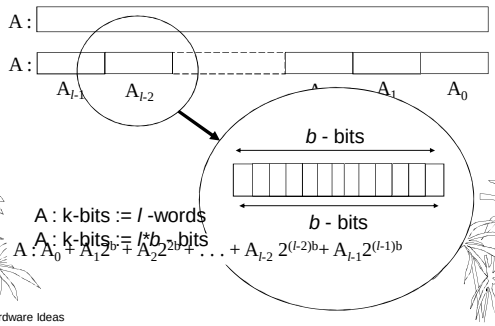
Add/Subtract Modular Multiplication Algorithm Data-path



Expandable RSA Hardware

- Crypto Hardware is faster & more secure compared to software
- However, it is limited by its design parameters and capability.
- If more security is needed & more bits are needed to be computed, a new hardware is to be designed and the old one cannot be utilized.
- Expandable hardware will be flexible to solve this problem by adding extra chips whenever needed.

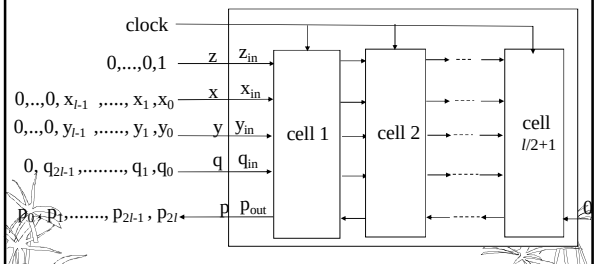
Numbers Representation



A : k-bits := l -words
A : k-bits := l*b -bits

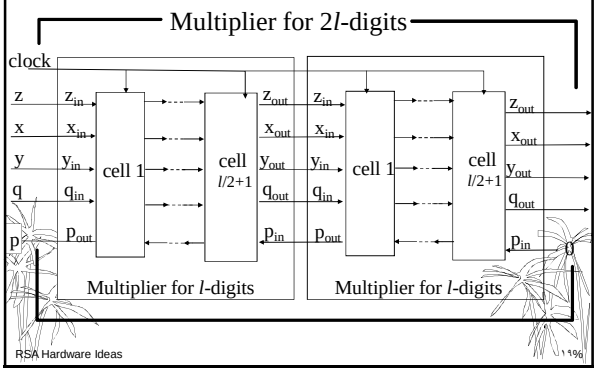
$A : A_0 + A_1 2^b + A_2 2^{2b} + \ldots + A_{l-2}\, 2^{(l-2)b} + A_{l-1} 2^{(l-1)b}$

Building the Systolic Multiplier



- (l/2 + 1) cells required for l-digit multiplication

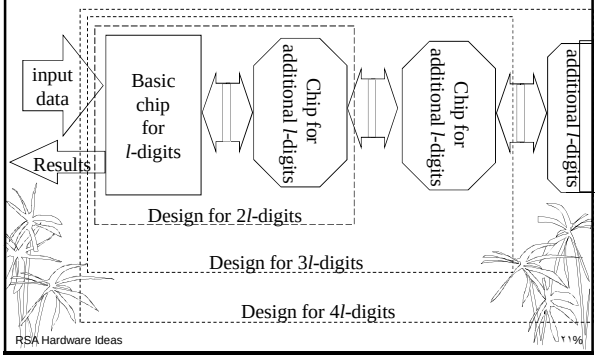
Expandable Systolic Multiplier



For Expandability

- Allow input data to have more digits
- Allow systolic multiplier to be expandable
- Allow registers to be expandable
- Multiplexing

The Expandable RSA Design



Expandable RSA Hardware Ideas Summary

- The expandable hardware has the best speed while its area is the largest.
- The add/subtract model has the best area while its speed is the slowest.
- The cost (Area*Time^2) of the merged design is the best followed by the expandable design, and then the add/subtract model.