Modular exponentiation

Hyoungshick Kim

Department of Software

College of Software

Sungkyunkwan University

Computation of $C = M^e$

- Must compute $C = M^e \bmod n$ and $P = C^d \bmod n$ for very large n, e, d, P, and C
  - Example: $3{,}000{,}000^{12007}$
- Problem: M^e is a <u>really big</u> number!
  - Too large for fast computation or even storage on most systems
  - M^e requires exponential time

Big integer implementation

- A big integer will be represented using an array of digits in base B. For example, the following struct can be used:

```
typedef struct {
int sign;
int size;
int *tab; } bignum;
```

  where sign is the sign bit, and size is the size of the dynamic array tab.

- Big number libraries (https://www.openssl.org/docs/crypto/bn.html, https://gmplib.org/, http://www.shoup.net/ntl/)

How can we reduce multiplications?

- We can simply compute $3{,}000{,}000^{12007}$ by multiplying 3,000,000 12007 times.
- However, it requires exponential time.
- Fast modular exponentiation is required.

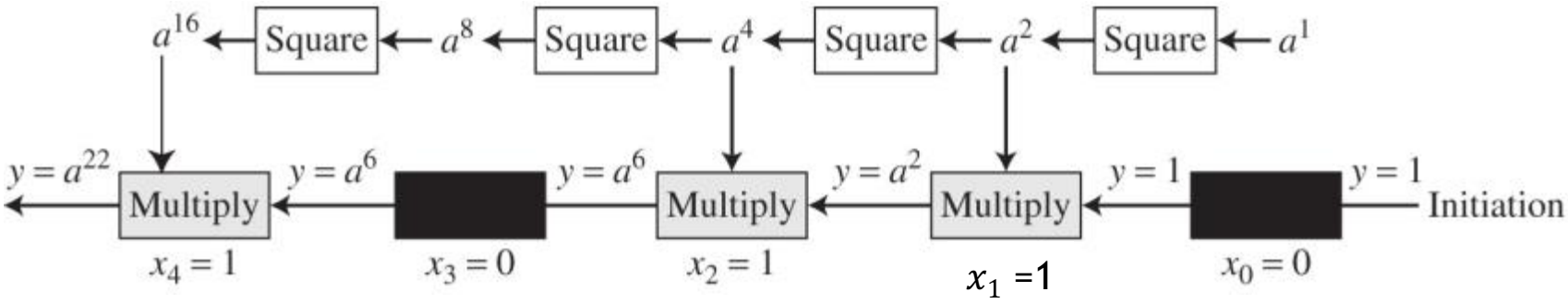
Do you have any idea about this?

How to calculate exponentiation

- Modular exponentiation example
  - $5^{20} \bmod 35 = 95367431640625 \bmod 35 = 25 \bmod 35$
- A better way: **repeated squaring**
  - $20 = 10100$ base 2
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - Note that $2 = 1 \cdot 2$, $5 = 2 \cdot 2 + 1$, $10 = 2 \cdot 5$, $20 = 2 \cdot 10$
  - $5^1 = 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- No huge numbers and it's efficient!

Square and multiply method

- Break exponent ***e*** into product of powers of 2
  - Example: $a^{22} = a^{16} \times a^4 \times a^2$
  - Can be represented as "bits": 22 → 10110
- Use **squaring** to compute powers of 2 quickly
  - $a \rightarrow a^2 \rightarrow a^4 \rightarrow a^8 \rightarrow a^{16}$
- Multiply by running total if corresponding bit = 1

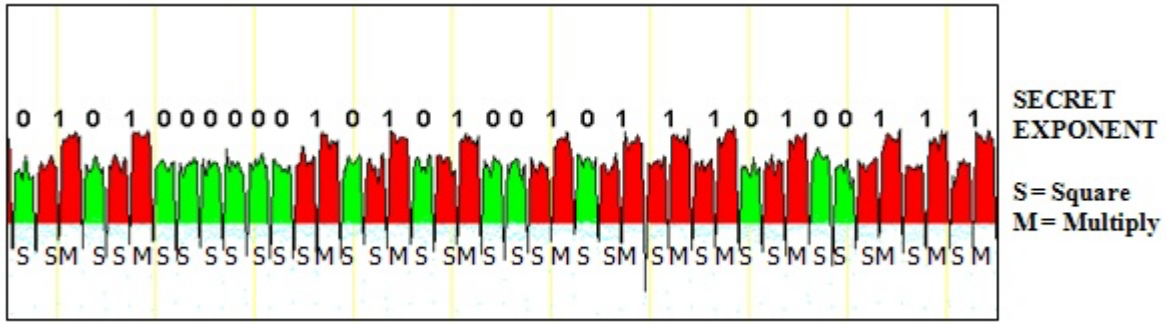


Pseudo code to compute M^e mod n

```
result = 1
for (i = 0 to number of bits in e - 1)
{
   if (i^th bit == 1)
      result = (result * M) mod n
   M = M^2 mod n
}
```

Simple power analysis

A power trace of a portion of an RSA exponentiation operation might recover the secret exponent *d*. Why?



(See https://www.youtube.com/watch?v=cPDDNVKo43w)

See the code!

How to prevent this?

```
result = 1
for (i = 0 to number of bits in e - 1) {
  if (i^th bit == 1)
     result = (result * M) mod n
  M = M^2 mod n
}
```

Questions?

Sungkyunkwan
Campus
Sciences
campus
established
university
University
School
Suwon
Korean
Seoul
Natural
edit
Graduate
English
constructed
Social
Global
Building
Institute
Professors
Confucian
Joseon
located
Engineering
Humanities
lecturers
schools
Science
Administration
Academic
ASLE-Korea
Hall
also
two
four
Advanced
Education