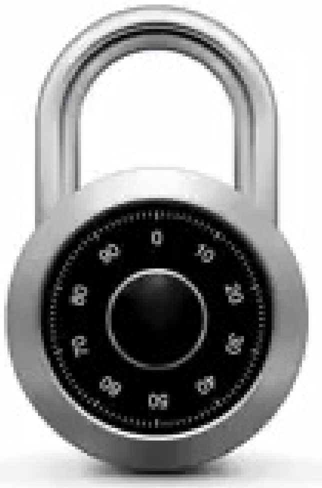
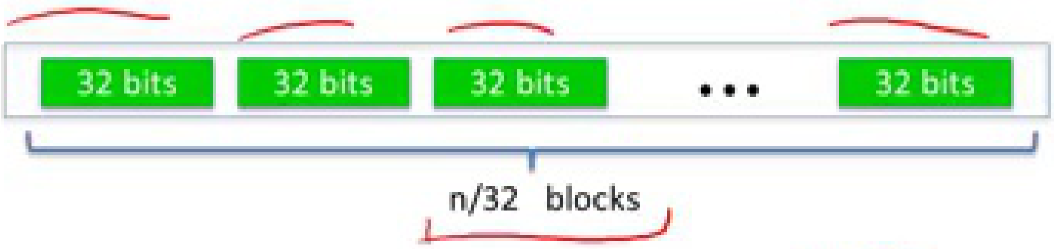


Intro. Number Theory

Arithmetic algorithms

Representing bignums

Representing an n-bit integer (e.g. n=2048) on a 64-bit machine



Note: some processors have 128-bit registers (or more)
and support multiplication on them

Arithmetic

Given: two n-bit integers

- **Addition and subtraction**: linear time $O(n)$
- **Multiplication**: naively $O(n^2)$. Karatsuba (1960): $O(n^{1.585})$ $\log_2 3$

  Basic idea: $(2^b x_2 + x_1) \times (2^b y_2 + y_1)$ with 3 mults.

  ⇒ Best (asymptotic) algorithm: about $\tilde{O}(n \cdot \log n)$.
- **Division with remainder**: $O(n^2)$.

Exponentiation

Finite cyclic group G (for example $G = \mathbb{Z}_p^*$)

$G = \{1, g, g^2, g^3, \ldots, g^{q}\}$

Goal: given g in G and x compute g^x

$g \cdot g \cdot g = g^3$

Example: suppose $x = 53 = (110101)_2 = 32+16+4+1$

Then: $g^{53} = g^{32+16+4+1} = g^{32} \cdot g^{16} \cdot g^{4} \cdot g^{1}$

$g^1 \longrightarrow g^2 \longrightarrow g^4 \longrightarrow g^8 \longrightarrow g^{16} \longrightarrow g^{32}$ $\qquad g^{53}$

9 mult.
comp
g^{53}.

Dan Boneh

The repeated squaring alg.

Input: g in G and x>0 ; **Output**: g^x

write $x = (x_n \, x_{n-1} \dots x_2 \, x_1 \, x_0)_2$

$y \leftarrow g$, $z \leftarrow 1$

$\log_2 x$

for i = 0 to n do:

if (x[i] == 1): $z \leftarrow z \cdot y$

$y \leftarrow y^2$

output z

example: g^{53}

y	z
g^2	g
g^4	g
g^8	g^5
g^{16}	g^5
g^{32}	g^{21}
g^{64}	g^{53}

Running times

Given n-bit int. N:

- **Addition and subtraction in Z_N**: linear time $T_+ = O(n)$
- **Modular multiplication in Z_N**: naively $T_\times = O(n^2)$
- **Modular exponentiation in Z_N (g^x)**:

$$O\Big(\underbrace{(\log x)}\cdot\underbrace{T_\times}\Big) \leq O\Big(\underbrace{(\log x)\cdot n^2}\Big) \underset{x<N}{\leq} O(n^3)$$

x<N

Dan Boneh

End of Segment