Modular exponentiation

To make RSA practical we must compute $x^D \bmod N$ when N and D are huge.

Example: $N = 101$, $a = 2$, $\overset{D}{\cancel{E}} = 43$.

$2^{43} \mod 101$

$43 = 32 + 11 = 32 + 8 + 3 = 32 + 8 + 2 + 1$

$= 2^5 + 2^3 + 2^1 + 2^0$

$a^{43} = a^{(2^0 + 2^1 + 2^3 + 2^5)}$

$= a^{2^0} \cdot a^{2^1} \cdot a^{2^3} \cdot a^{2^5}$

$a = 2$ $\quad 2^{2^5}$

$2^2 = 4$ $\quad (2^2)^2 = 16 \cdot 2 = 32$

$a^{2^n} = (((a^2)^2)^2)\cdots$ n times

$2 \cdot (2^2) \cdot (8)(32)$

$2, 2^2, 2^{\cancel{3}4}, 2^8, 2^{16}, 2^{32}$ $\quad$ 9 mod.

Repeated squaring

Algorithm: Given a, E, and N, the goal is to compute a^E mod N.

- Initialize: Set S=a, P=1, T=E
- Loop: While T>1:
  - if T is odd, set P = (S*P) mod N.
  - Replace S by S*S mod N.
  - Divide T by 2, dropping any remainder
- Finish: Return (S*P) mod N

S	P	T
2	1	43
4	2	21
16	8	10
54	8	5
88	28	2
68	28	1

16^2

$16^2 = 256$
202
54

$\sim \log_2 E$ steps

Return = 86 = 68*28 mod 101.

$2^{43} \equiv 86 \mod 101.$

Proof of repeated squaring

- Initialize: Set S=a, P=1, T=E
  - Loop: While T>1:
    - if T is odd, set P = (S*P) mod N.
    - Replace S by S*S mod N.
    - Divide T by 2, dropping any remainder
  - Finish: Return (S*P) mod N

Suppose M is the output of the algorithm with input $S = \underline{a}$, $P = 1$, and $T = E$.

- The first observation we make is that if we initialize the algorithm with $P = k$ instead of $P = 1$, then the output is kM.

- If $E = 1$, the output is a, as it should be.

Returns $S \cdot P = a^1$ $S = a$ $P = 1$

Return $a = a^1$.

Suppose the algorithm works for all $e < E$, and any a and N. We show it works for E.

If E is even:

- the first step replaces a by a^2, keeps P as 1 and replaces E by $E/2$.
- By induction the rest of the algorithm computes $(a^2)^{E/2}$ mod N or a^E mod N as desired.

S	P	T
a	1	E
$\underline{a^2}$	1	$\underline{E/2}$
	$\vdots$	

computes

$(a^2)^{E/2}$ mod

$\parallel$

a^E mod N.

If E is odd:

- the first step replaces a by a^2, replaces P by a, and replaces E by $(E-1)/2$.

- By induction the rest of the algorithm with an initial P of 1 would computes $(a^2)^{(E-1)/2}$ which is $a^{E-1} \bmod N$. But since P started at a, it computes $aa^{E-1} = a^E \bmod N$ as desired.

S	P	T
a	1	E
a^2	a	$(E-1)/2$

$a\left[a^{2(E-1)/2} \bmod N\right]$

$\parallel$

$a \cdot a^{E-1} \bmod N \equiv a^E \bmod N$

Running time $\sim \log_2 E < \log_2 N$