The MIST Exp^n Algorithm

{ *To compute: ResultM = M^E* }
StartM ← M ;
ResultM ← 1 ;
While E > 0 **do**
Begin
  Choose a random "divisor" D ;
  R ← E **mod** D ;
  If R ≠ 0 **then**
    ResultM ← StartM^R × ResultM ;
  StartM ← StartM^D ;
  E ← E **div** D ;
  { Invariant: $M^{E.Init} = StartM^E \times ResultM$ }
End

Addition Sub-Chains

- For each pair (D,R) we need an addition chain which calculates $StartM^D$ and $StartM^R$ efficiently.

1+1 = 2	for $D = 2$, any R
1+1 = 2 ; 1+2 = 3	for $D = 3$, any R
1+1 = 2 ; 1+2 = 3 ; 2+3 = 5	for $D = 5$, any $R \neq 4$
1+1 = 2 ; 2+2 = 4 ; 1+4 = 5	for $D = 5$, $R = 4$

- These are minimal, i.e. fewest possible mult^{ns}.

Addition Sub-Chains

- We need instructions which include the update of *ResultM*: *ijk* means multiply contents at addresses *i* and *j* and write result to address *k*.
- Use 1 for location of *StartM*, 2 for *TempM*, 3 for *ResultM*:

(111)	for $(D,R) = (2,0)$
(112, 133)	for $(D,R) = (2,1)$
(112, 121)	for $(D,R) = (3,0)$
(112, 133, 121)	for $(D,R) = (3,1)$
(112, 233, 121)	for $(D,R) = (3,2)$
(112, 121, 121)	for $(D,R) = (5,0)$
(112, 133, 121, 121)	for $(D,R) = (5,1)$
(112, 233, 121, 121)	for $(D,R) = (5,2)$
(112, 121, 133, 121)	for $(D,R) = (5,3)$
(112, 222, 233, 121)	for $(D,R) = (5,4)$

Probability of each (D,R)

- This gives the probabilities:

$p_D = \sum_i p_i p_{D|i}$ for each divisor D

$p_{D,R} = \sum_{i \equiv R \bmod 30} p_i p_{D|i}$ for each pair (D,R)

For the divisor selection process above:

$p_2 = 0.629$
$p_3 = 0.228$
$p_5 = 0.142$

Av^{age} Add^n Chain Properties

- The probabilities of addition sub-chain lengths are:

length 1 is	$p_{2,0}$	= 0.354
length 2 is	$p_{3,0} + p_{2,1}$	= 0.458
length 3 is	$p_{5,0} + p_{3,1} + p_{3,2}$	= 0.139
length 4 is	$p_{5,1} + p_{5,2} + p_{5,3} + p_{5,4}$	= 0.049

- So average divisor sub-chain has length 1.883 mult^s
- Av decrease in E is $2^{p_2}3^{p_3}5^{p_5} = 2.500$ per subchain
- So $0.757 \log_2 E$ subchains & $1.425 \log_2 E$ mult^s
- This is ***faster*** than the binary exp^n algorithm and marginally slower than 4-ary exp^n

Choice of Divisor

Initial choice:

D ← 0 ;
If Random(8) < 7 **then**
  If (E **mod** 2) = 0 **then** D ← 2 **else**
  If (E **mod** 5) = 0 **then** D ← 5 **else**
  If (E **mod** 3) = 0 **then** D ← 3 ;
If D = 0 **then**
Begin
  p ← Random(8) ;
  If p < 6 **then** D ← 2 **else**
  If p < 7 **then** D ← 3 **else**
  D ← 5
End

Av^{ge}: $1.4247 \times \log_2 E$ mult^{ns}

Choice of Divisor

A semi-deterministic choice:

D ← 0 ;
{ ***Delete this line:*** **If** Random(8) < 7 **then** }
If (E **mod** 2) = 0 **then** D ← 2 **else**
If (E **mod** 5) = 0 **then** D ← 5 **else**
If (E **mod** 3) = 0 **then** D ← 3 ;
If D = 0 **then**
Begin
p ← Random(8) ;
If p < 6 **then** D ← 2 **else**
If p < 7 **then** D ← 3 **else**
D ← 5
End

Av^{ge}: $1.4197\times\log_2 E$ mult^{ns}

S&M Chains

- Assume an attacker can distinguish **Squares** and **Multiplies** from a ***single*** exponentiation (e.g. from Hamming weights of arguments deduced from power variation on bus.)
- A **division chain** is the list of pairs ***(D,R)*** used in an exp^n scheme. It determines the *addition chain* to be used, and hence the sequence of *squares* and *multiplies* which occur:

(2,0)	*S*	(2,1), (3,0)	*SM*
(3,1), (3,2), (5,0)	*SMM*	(5,1), (5,2), (5,3)	*SMMM*
(5,4)	*SSMM*		

- Divisor sub-chain boundaries are deduced from occurrences of *S* except for ambiguity between (5,4) and (2,0)(3,*x*) or (2,0)(5,0).

S&M Chains

- There is/are:
  - 1 way to interpret ***S***
  - 2 ways to interpret ***SM***
  - 3 ways to interpret ***SMM*** with no preceding *S*
  - 4 ways to interpret ***SMM*** with preceding *S*
  - 4 ways to interpret ***SMMM***
- Using the known probabilities for each occurring:
  THEOREM: The search space for exponents with the same S&M sequence as *E* has size approx $E^{3/5}$.
- For 4-ary exp^n, it is ***much*** easier to average traces, easier to be certain of the S&M sequence, and the search space is only $E^{7/18}$ – which is smaller.

Operand Re-Use

- **THEOREM: With MIST, the search space for exponents with the same operand sharing sequence as *E* has size approx $E^{1/3}$.**
  - this assumes op^d sharing is determined with total accuracy from one exponentiation;
  - it also assumes unconstrained choice of divisors at each step;
  - in comparison, the search space for *m*-ary exp^n has size E^0.
- It isn't clear if recovery from errors is possible.
- Selecting exact divisors will vastly decrease the search.

Deterministic Choices

- The deterministic constraints cut the search space for *E*.
- By how much? Consecutive divisor choices are not independent, so theory simplified this way is inadequate.
- When the divisor is chosen semi-deterministically (as above) and these constraints are taken into account:
  THEOREM: The search space for exponents with the same S&M sequence as *E* has size approx $E^{1/4}$.
- It is still computationally infeasible to recover *E*.

Deterministic Choices

- Knowledge of op^d sharing cuts the search space further.
- By how much? Simulations were used to find out.
- When the divisor is chosen deterministically and these constraints are taken into account:
  THEOREM: The search space for exponents with the same op^d sharing pattern as *E* has size approx $E^{0.115}$.
- It may now be computationally feasible to recover *E*:
  768-bit exponents give search space of size 2^{88},
  1024-bit known RSA modulus with CRT has size only 2^{59}.