# MONTGOMERY MODULAR MULTIPLICATION

EE577A

Fall 2019

# Montgomery Form

- Assume R is a convenient constant (i.e. $2^7$), the Montgomery form "$A$" of a number "a $mod\ N$ " is defined as:
$$A = aR\ (mod\ N)$$
- Here we define an operation as $S = AR^{-1}\ (mod\ N) = reduce(A)$ .
- $R^{-1}$ is an unsigned number that has $RR^{-1} = 1\ (mod\ N)$.
- *We usually choose the* $R^{-1} \in [0, N)$.

- Axiom: If $a = p\ mod\ m$, $b = q\ mod\ m$ , then $ab\ mod\ m = pq\ mod\ m$.

# Montgomery Form
# Why Montgomery Reduction Useful?

- Here we define an operation as $S=TR^{-1} \pmod N = reduce(T)$.
- $Z = zR \pmod N = x*y*R \pmod N$

$$= xR*yR*R^{-1} \pmod N$$

$$= XYR^{-1} \pmod N$$

$$= reduce(X*Y)$$

# Montgomery Form
# Why Montgomery Reduction Useful?

- Here we define an operation as $S = AR^{-1}\ mod\ N = reduce(A)$ .
- To convert *A=aR (mod N)* to *a (mod N)*
- $a \pmod N = (a*1) \pmod N$

$$= (a*R*R^{-1}) \pmod N$$

$$= ((a*R \pmod N))* R^{-1}) \pmod N$$

$$= (A* R^{-1}) \pmod N$$

$$= reduce(A*1)$$

# Montgomery Form
# Why Montgomery Reduction Useful?

- Here we define an operation as $S=TR^{-1} \pmod N = reduce(T)$.
- To convert $a \pmod N$ to $A=aR \pmod N$,
- $a*R \pmod N = (a*1*R) \pmod N$

  $= (a*R*R^{-1}*R) \pmod N$

  $= ((a \pmod N)*R*R \pmod N)* R^{-1}) \pmod N$

  $= (a*K* R^{-1}) \pmod N$

  $=reduce(a*K)$
- $K=R*R \pmod N$

# Montgomery Reduction Algorithm

- We want to realize:

- Given R, N and T=x*y
- Calculate S=$TR^{-1}$ *(mod N)=reduce(T).*

# Montgomery Reduction Algorithm

- Assumptions:

- x and y are 7-bit unsigned numbers.
- R and N are coprime pairs, i.e. 10 and 7.
- N is a 7-bit unsigned number.
- Here we assume $R= 2^7$
- $T=x*y$ is in the range $[0,R*N-1]$.

# Montgomery Reduction Algorithm

- Algorithm:

(1) Calculate T=x*y

(2) Calculate $m = (T\ (mod\ R)N^{-1})(mod\ R)$

(3) Calculate $t = (T + mN)/R$

(4) S= $\begin{cases} t - N & if\ t \geq N \\ t & if\ t < n \end{cases}$

# Bezout’s Identity:

Bezout’s Identity:

When R and N are coprime, we can find $R^{-1}$ and $N^{-1}$ so that $RR^{-1} - NN^{-1} = 1$.

And we have $N^{-1} \in [0, R-1], R^{-1} \in [0, N-1]$

# Extended Euclidean Algorithm

Find the value of $R^{-1}$ and $N^{-1}$: Extended Euclidean Algorithm

For two integers a and b, we can find two integers x and y that ax+by=gcd(a,b)=d.

If a and b are coprime, then d=1,=> ax+by=1.

Here we assume a=R, b=N,

$R \cdot R^{-1} - N \cdot N^{-1} = 1$

We can find that $R^{-1} = x$, $N^{-1} = -y$

# Extended Euclidean Algorithm

$$r_0 = R, s_0 = 1, t_0 = 0$$
$$r_1 = N, s_1 = 0, t_1 = 1$$
$$q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$$
$$r_{i+1} = r_{i-1} - q_i * r_i, s_{i+1} = s_{i-1} - q_i * s_i, t_{i+1} = t_{i-1} - q_i * t_i$$

The procedure stops when $r_{k+1} = 0$, and it gives $1 = r_k = Rs_k + Nt_k$.

Here we suppose to have $s_k > 0\ and\ t_k < 0$, then $R^{-1} = s_k$, $N^{-1} = -\,t_k$

If the output is reversed, say $s_k < 0\ and\ t_k > 0$, take $R^{-1} = s_k + N$, $N^{-1} = -(\,t_k - R)$

# Extended Euclidean Algorithm Example: R=10, N=3

$$r_0 = R, s_0 = 1, t_0 = 0$$
$$r_1 = N, s_1 = 0, t_1 = 1$$
$$q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$$
$$r_{i+1} = r_{i-1} - q_i * r_i, s_{i+1} = s_{i-1} - q_i * s_i, t_{i+1} = t_{i-1} - q_i * t_i$$

| $i$ | $q_{i-1}$ | $r_i$ | $s_i$ | $t_i$ |
|---|---|---|---|---|
| 0 | | 10 | 1 | 0 |
| 1 | | 3 | 0 | 1 |
| 2 | 10\3=3 | 10-3*3=1 | 1-3*0=1 | 0-3*1=-3 |
| 3 | 3\1=3 | 3-3*1=0 | 0-3*1=-3 | 1-3*(-3)=10 |

$$R^{-1} = 1, N^{-1} = 3$$

# Extended Euclidean Algorithm Example: R=10, N=7

$$r_0 = R, s_0 = 1, t_0 = 0$$
$$r_1 = N, s_1 = 0, t_1 = 1$$
$$r_{i+1} = r_{i-1} - q_i * r_i, s_{i+1} = s_{i-1} - q_i * s_i, t_{i+1} = t_{i-1} - q_i * t_i$$
$$q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$$

| $i$ | $q_{i-1}$ | $r_i$ | $s_i$ | $t_i$ |
|---|---|---|---|---|
| 0 | | 10 | 1 | 0 |
| 1 | | 7 | 0 | 1 |
| 2 | 10\7=1 | 10-1*7=3 | 1-1*0=1 | 0-1*1=-1 |
| 3 | 7\3=2 | 7-2*3=1 | 0-2*1=-2 | 1-2*(-1)=3 |
| 4 | 3\1=3 | 3-3*1=0 | 1-3*(-2)=7 | (-1)-3*3=-10 |

$$R^{-1} = -2 + 7 = 5, N^{-1} = -(3 - 10) = 7$$

# Montgomery Reduction Algorithm

- Assumptions:

- x and y are 7-bit unsigned numbers.
- R and N are coprime pairs, i.e. 10 and 7.
- N is a 7-bit unsigned number.
- Here we assume R= $2^7$
- T=x*y is in the range [0,RN-1].
- $RR^{-1} - NN^{-1} = 1$

# Montgomery Reduction Algorithm

- Algorithm:

(1) Calculate T=x*y

(2) Calculate $m = (T\ (mod\ R)N^{-1})\ (mod\ R)$

(3) Calculate $t = (T + mN)/R$

(4) S= $\begin{cases} t - N & if\ t \geq N \\ t & if\ t < n \end{cases}$

# Montgomery Reduction Algorithm Proof

We want calculate $TR^{-1}\ (mod\ N)$

$$TR^{-1} = \frac{T\color{red}{RR^{-1}}}{R}$$

$(RR^{-1} - NN^{-1} = 1)$

$$= \frac{T(1+NN^{-1})}{R} = \frac{T+TNN^{-1}}{R}$$

# Montgomery Reduction Algorithm Proof

We want calculate $TR^{-1}\ (mod\ N)$

$$TR^{-1} = \frac{TRR^{-1}}{R}$$

$(RR^{-1} - NN^{-1} = 1)$

$$= \frac{T(1+NN^{-1})}{R} = \frac{T+TNN^{-1}}{R}$$

$$TR^{-1}\ (mod\ N) = \frac{T + TNN^{-1}}{R}\ (mod\ N) = \frac{T + (TN^{-1})N}{R}\ (mod\ N)$$

# Montgomery Reduction Algorithm Proof

$$TR^{-1}\ (mod\ N) = \frac{T + (TN^{-1})N}{R}\ (mod\ N)$$

$$TN^{-1} = L * R + [TN^{-1}\ (mod\ R)], L\ is\ an\ integer$$

# Montgomery Reduction Algorithm Proof

$$TR^{-1}\ (mod\ N) = \frac{T + (TN^{-1})N}{R}\ (mod\ N)$$

$$TN^{-1} = L * R + [TN^{-1}\ (mod\ R)], L\ is\ an\ integer$$

$$TR^{-1}\ (mod\ N) = \frac{T + (L * R + [TN^{-1}\ (mod\ R)])N}{R}\ (mod\ N)$$

$$= \frac{T + L * R * N + [TN^{-1}\ (mod\ R)] * N}{R}\ (mod\ N)$$

$$= \frac{T + [TN^{-1}\ (mod\ R)] * N}{R} + LN\ (mod\ N)$$

$$= \frac{T + [TN^{-1}\ (mod\ R)] * N}{R}\ (mod\ N)$$

# Montgomery Reduction Algorithm Proof

$$TR^{-1}\ (mod\ N) = \frac{T + (TN^{-1})N}{R}\ (mod\ N)$$

$$TN^{-1} = L * R + [TN^{-1}\ (mod\ R)], L\ is\ an\ integer$$

$$TR^{-1}\ (mod\ N) = \frac{T + (L * R + [TN^{-1}\ (mod\ R)])N}{R}\ (mod\ N)$$

$$TR^{-1}\ (mod\ N) = \frac{T + L * R * N + [TN^{-1}\ (mod\ R)] * N}{R}\ (mod\ N)$$

$$TR^{-1}\ (mod\ N) = \frac{T + [TN^{-1}\ (mod\ R)] * N}{R} + LN\ (mod\ N)$$

$$TR^{-1}\ (mod\ N) = \frac{T + [TN^{-1}\ (mod\ R)] * N}{R}\ (mod\ N)$$

$$TR^{-1}\ (mod\ N) = \frac{T + \left[\left[\left(T(mod\ R)\right)N^{-1}\right](mod\ R)\right] * N}{R}\ (mod\ N)$$

# Montgomery Reduction Algorithm Proof

We want calculate $TR^{-1}\ (mod\ N)$

$$TR^{-1} = \frac{TRR^{-1}}{R}$$

$(RR^{-1} - NN^{-1} = 1)$

$$= \frac{T(1+NN^{-1})}{R} = \frac{T+TNN^{-1}}{R}$$

$$TR^{-1}\ (mod\ N) = \frac{T + TNN^{-1}}{R}\ (mod\ N) = \frac{T + (TN^{-1})N}{R}\ (mod\ N)$$

$$m = TN^{-1}(mod\ R) = \big(T(mod\ R)\big)N^{-1}(mod\ R) < R$$

$$TR^{-1}\ (mod\ N) = \frac{T + mN}{R}\ (mod\ N)$$

# Montgomery Reduction Algorithm Proof

We want calculate $TR^{-1}\ (mod\ N)$

$$TR^{-1} = \frac{TRR^{-1}}{R}$$

$(RR^{-1} - NN^{-1} = 1)$

$$= \frac{T(1+NN^{-1})}{R} = \frac{T+TNN^{-1}}{R}$$

$$TR^{-1}\ (mod\ N) = \frac{T+TNN^{-1}}{R}\ (mod\ N) = \frac{T+(TN^{-1})N}{R}\ (mod\ N)$$

$$m = TN^{-1}(mod\ R) = \big(T(mod\ R)\big)N^{-1}(mod\ R) < R$$

$$TR^{-1}\ (mod\ N) = \frac{T+mN}{R}\ (mod\ N)$$

T $\in [RN-1]$

$$T + mN \in [0, RN - 1 + (R-1) * N < 2RN]$$

$$t = \frac{T+mN}{R} \in [0,2N-1]$$

# Montgomery Reduction Algorithm Implementation

- Algorithm:

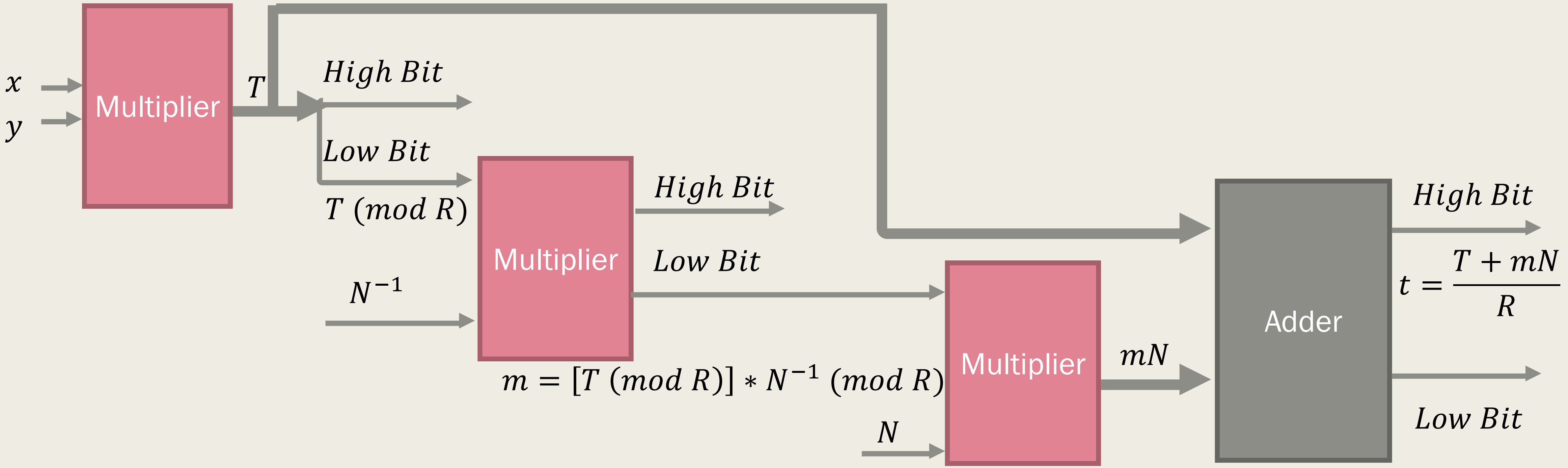
(1) Calculate T=x*y

(2) Calculate $m = (T\ (mod\ R)N^{-1})\ (mod\ R)$

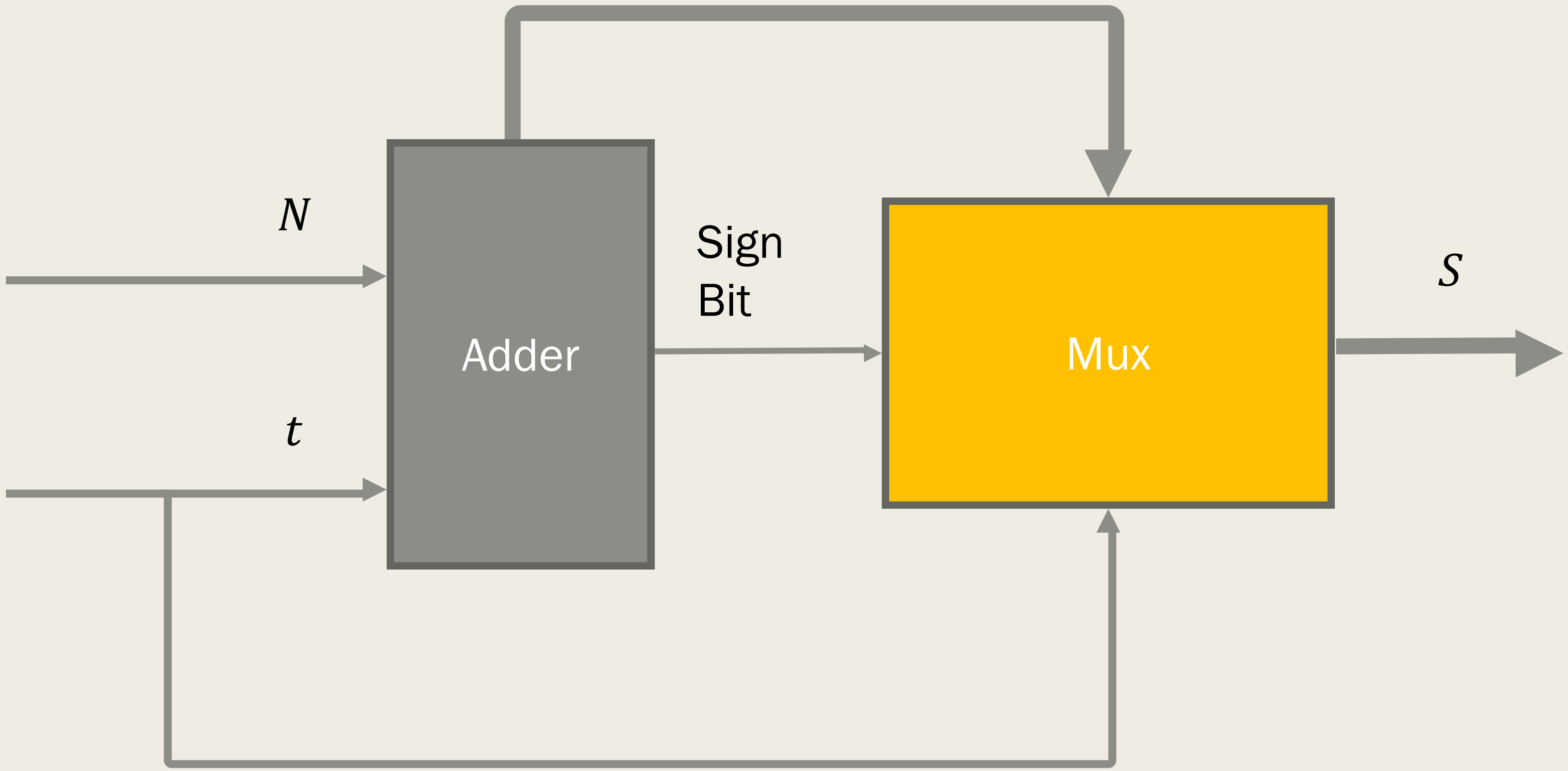
(3) Calculate $t = (T + mN)/R$

(4) S= $\begin{cases} t - N & if\ t \geq N \\ t & if\ t < n \end{cases}$

# Montgomery Reduction Algorithm Implementation



# Montgomery Reduction Algorithm Schematic Diagram



# Montgomery Reduction Algorithm Example

Calculate S=*TR*$^{-1}$ *(mod N)=reduce(T).*

R=128, N=5, x=8, y=57
With Extended Euclidean Algorithm,
*N*=5=00000101, $N^{-1}$=51=00110011, $R^{-1}$ =2=00000010, x=8=00001000, y=57=00111001
T=x*y=456=0000000111001000
T (mod R)=72=01001000=72,
T(mod R) $*$ $N^{-1}$=3672=0000111001011000
[T(mod R) $*$ $N^{-1}$] (mod R)=88=01011000=m
T+mN=896=0000001110000000
(T+mN)/R=7=00000111=t
S=t-N=7-5=2=00000010

# Layout
# Block Diagram