Reduce-by-Feedback: Timing resistant and DPA-aware Modular Multiplication plus: How to Break RSA by DPA

M. Vielhaber
vielhaber@gmail.com

Hochschule Bremerhaven und/y Universidad Austral de Chile

CHES 2012

Overview

DH .. EC .. RSA $\equiv$ Modular Addition

DH $\equiv$ Modular Exponentiation
RSA $\equiv$ Modular Power Function

$\mapsto$ ”Square-and-Multiply” $\longrightarrow$

Modular Multiplication

$\mapsto$ ”Shift-and-Add” $\longrightarrow$

Modular Addition
$M^+ := (M << 3 + \alpha \cdot B) \mod N$
(here in octal base, 3 bits per cycle)
or
$M^+ := (M >> 3 + \alpha \cdot B) \mod N$
(Montgomery multipliction, $M = [A \cdot B \cdot 2^{-L}] \mod N$)

Schoolbook Multiplication I: Algorithm Shift-and-Add

```
Parameters:
operand length $L$ [$e.g. = 1024$]
shift length per clock cycle
$z$ [$e.g. = 3$], with $Z := 2^z$ [$e.g. = 8$]

IN $A, B < 2^l$ // factors, where
$A = \sum_{k=0}^{L-1} a_k 2^k = \sum_{k=0}^{\lceil L/z \rceil -1} \alpha_k Z^{\lceil L/z \rceil -1-k}$

OUT $M$ // product $M = A \cdot B$

Algorithm:
$M := 0$
FOR $k := 0$ TO $\lceil L/z \rceil - 1$
   $M := (M << z) + \alpha_k \cdot B$
ENDFOR
```

```
172 * 315
---------
      860
     172
    516
---------
    54180
```

Properties of Shift-and-Add

Four trivial, but remarkable properties of Shift-and-Add:

(*i*) $\alpha_k \in \{0, 1, \ldots, Z-1\}$,
thus Z possible multiples of B.

(*ii*) Exactly $\lceil l/z \rceil$ cycles to go in the loop $\rightarrow$ no timing attack.

(*iii*) Cut number of multiples in half (I):
It is sufficient to store the multiples for $\alpha \geq Z/2$, and $\alpha = 0$, by supplying shifted copies for the smaller cases.

(*iv*) Cut number of multiples in half (II):
The "1-off trick":
Replace the odd multiples by the next higher even ones,
subtract $Z \cdot B$ in the next clock cycle:
$((\alpha_k \cdot B) << z) + \alpha_{k+1} \cdot B = (((\alpha_k + 1) \cdot B) << z) + (\alpha_{k+1} - Z) \cdot B.$
Putting $C_{\alpha,k} := 1$, iff α_k is odd, 0 otherwise, we set

$$\overline{\alpha}_k := \alpha_k + C_{\alpha,k} - Z \cdot C_{\alpha,k-1} \text{ and } M := (M << z) + \overline{\alpha}_k \cdot B.$$

Physically Stored Multiples

(*iii*) and (*iv*) combined require multiples
$\pm(Z/2+2), \pm(Z/2+4), \ldots, \pm Z, 0$,
where we first applied (*iv*), then (*iii*).

Only the $Z/4$ multiples $Z/2+2, Z/2+4, \ldots, Z$ have to be stored in hardware, a 75% savings.

Schoolbook Multiplication II

```
172 * 315
----------
       860
      172
     516
----------
     54180
```

```
    |860|
      >>
    |  86|0
  + |172|
  = |258|0
      >>
    |  25|80
  + |516|
  = |541|80
      >>
    |  54|180●
```

Problem I:
Decimal Point is far to the right (green ●), not |

Solution I:
"Live" in residue classes $[x \cdot 2^L]$, $(A \cdot 2^L) \cdot (B \cdot 2^L) \cdot 2^{-L} = A \cdot B \cdot 2^L$, all results include factor 2^L, only adjust in the first and last step

Problem II:
Bits run off to the right ...

Montgomery Multiplication

Montgomery Multiplication

Problem II:
Bits run off to the right ...

Solution II:
Add suitable multiple of modulus $N \rightarrow$ only zeroes run off to the right
Decimal example, let $N = 111$

```
172 * 315
   |860|
+  |000|
=  |860|
      >>
   |  86|0
+  |172|
+  |222|
=  |480|0
```

```
      >>
   |  48|00
+  |516|
+  |666|
=1|230|00
        >>
   |123|000
```

Montgomery: Adjustment of LSBs

Let N end in (e.g.) ..101

When M ends in	Adjust by	which is $N \cdot \ldots$	which is also $N \cdot \ldots$
..000	..000	0	$8 = 1 << 3$
..001	..111	3	-5
..010	..110	6	$(-1) << 1$
..011	..101	1	1
..100	..100	4	$1 << 2$
..101	..011	7	-1
..110	..010	2	$1 << 1$
..111	..001	5	5

25% Physically stored:
..01 Physically stored

75% for free:
..11 Negative,
2s complement for free
..0 Shifts,
for free (only wires, no FF)

Schoolbook Multiplication III

Once again, but in reverse order, shifting to the left ...

```
172 * 315        |516|
---------         <<
    516         5|160|
     172     +   |172|
      860    =  5|332|
---------         <<
    54180      53|320|
             +   |860|
             =54|180|
```

Problem:
Digits run off to the left

Solution:
Reduce-by-Feedback
(LFSR-style)

Reduce-by-Feedback

Reduce-by-Feedback: The Idea

Reduce-by-Feedback: Mix of LFSR and Shift-and-Add ideas

The original idea stems from the analogy with LFSR's

The z bits running off in front for each Shift-and-Add step are fed back into the accumulator:

Partition M into its lower $L+z+1$ bits and the higher part,

$$M_H = \lfloor M/2^{L+z+1} \rfloor, M_L = M \mod 2^{L+z+1},\ M = (M_H|M_L).$$

Also, let $K \equiv 2^{L+2z+1} \mod N, 0 \leq K < N$.
Then

$$(M_H|M_L) << z = M_H \cdot 2^{L+2z+1} + M_L \cdot 2^z \equiv M_H \cdot K + M_L \cdot 2^z \mod N$$

Algorithm Reduce-by-Feedback

```
Shift-and-Add-with-Reduce-by-Feedback

$M := 0, C_{\alpha,-1} := 0, C_{\mu,-1} := 0$
FOR $k := 0$ TO $\lceil l/z \rceil - 1$
    $C_{\alpha,k} := \alpha_k$ AND $1, \overline{\alpha}_k := \alpha_k + C_{\alpha,k} - Z \cdot C_{\alpha,k-1}$
    $\mu_k := \lfloor M/2^{l+z+1} \rfloor$
    $C_{\mu,k} := \mu_k$ AND $1, \overline{\mu}_k := \mu_k + C_{\mu,k} - Z \cdot C_{\mu,k-1}$ // this is $M_H$
    $M := ((M \mod 2^{l+z+1}) << z) + \overline{\alpha}_k \cdot B + \overline{\mu}_k \cdot K$
ENDFOR
```

Reduce-by-Feedback preserves the 4 properties of Shift-and-Add

(*i*) The standard range for the multiples of K is
$\mu_k \in \{-1, 0, 1, \ldots, 2^z\}$.

(*ii*) The `FOR` loop excutes exactly $\lceil l/z \rceil$ times, each run comprising a shift and 2 additions.
NO Timing Attack!

(*iii*) Required multiples of K:
$\mu_k \in \{0\} \cup \{Z/2 + 1, \ldots, Z\}$,
the others by shifting.

(*iv*) NO odd multiples of K by the "1-off trick"
In total we need $\alpha_k, \mu_k \in \{0, \pm(Z/2 + 2), \pm(Z/2 + 4), \ldots, \pm Z\}$,
with 0 and $\pm$ for free in hardware.

Reduce-by-Feedback is thus *completely analogous* to Shift-and-Add.

Historic Timetable

1985	Montgomery, ”Modular multiplication without trial division”
	Reduce-by-Feedback:
1987	V., Diploma thesis (TH Karlsruhe, Prof. Thomas Beth)
1989	V., E.I.S.S. Report 89/14
1989	Beth,Gollmann, ”Algorithm Engineering ...”
1990	Patent DE 3924344 (V., ”Multiplikations-/Reduktionseinricht.”)
	Rediscovery of Reduce-by-Feedback:
1995	Benaloh, Dai ”Fast Modular Reduction (Crypto Rump S.)
	Re-Re-Discovery of Reduce-by-Feedback:
1997	Jeong, Burleson, ”VLSI Array Algorithms ...”
1998	Patent US 5724279 (Josh Benaloh, Wei Dai, ”Computer-implemented method ...”)

Comparison Montgomery Multiplication and Reduce-by-Feedback

Montgomery multiplication (1985):
1st factor: Bits from LSB to MSB — shift **down** and add
residue classes $[x \cdot 2^L]$ mod N instead of standard residue classes $[x]$

Reduce-by-Feedback (1987 etc.)
1st factor: Bits from MSB to LSB — shift and add
standard residue classes $[x]$

Both MM and RbF ...
Immune against timing attacks, since
exactly $L/3 + \varepsilon_{const}$ cycles per mult/square

Susceptible (but fixable) to DPA ... later ...

Reduce-by-Feedback: No Overflow

$$(M_H^+|M_L^+) := (M_L << 3) + \alpha \cdot B + \mu \cdot K$$

with

$$\begin{aligned} 0 \leq & \quad M_L < & 8 \cdot 2^{L+4} \\ 0 \leq & \quad B, K < & 2^L \\ -8 \leq & \quad \alpha, \mu \leq & 8 \end{aligned}$$

Total:

$$0 + (-8) \cdot 2^L + (-8) \cdot 2^L$$
$$< M^+ <$$
$$8 \cdot 2^{L+4} + 8 \cdot 2^L + 8 \cdot 2^L$$
$$\Leftrightarrow$$
$$-1 \cdot 2^{L+4} < M^+ < 9 \cdot 2^{L+4} \Rightarrow -1 \leq M_H^+ \leq 8$$

Including the “1-off trick”, $-8, -6, \ldots, 6, 8$ are the necessary multiples

H/W Issues I: Re-use of MUX Tree and MUX Ctrl Vars

Compare $\alpha \cdot B$ and $\mu \cdot K$:
Same decision logic for $A \to \alpha$ and $M_H \to \mu$
Same 75% physical savings only $6 \cdot B, 8 \cdot B$ and only $6 \cdot K, 8 \cdot K$ phys.
Same MUX tree MUX Inputs
$-8B, -6B, -4B, ..., 6B, 8B$ and $-8K, -6K, -4K, ..., 6K, 8K$

Idea: Use H/W in both clock half cycles
Clk = L: do $A \to \alpha$, Clk = H: do α →MUX→ αB
Clk = H: do $M_H \to \mu$, Clk = L: do μ →MUX→ μK

Same Ctrl glue logic, same MUX tree, same shift wires used twice:
50% savings in both CTRL and BITSLICE (this beats Montgomery!)

Map 1987's 13 bit slices/mm^2 with 1.0μ design rules to current 65 *nm* rules, naïvely shrinking by $\frac{65}{1000}^2$: $13 \cdot \frac{65}{1000}^2 \approx 3000$ bits/mm^2

Full 4096 bit RSA with control unit on about 1.5 *mm*^{2}
FPGA implementation [not yet] under way...

H/W Issues II: Delayed-Carry-Adder

Use Brickell's Delayed-Carry-Adder, a chain of halfadders instead of full adders with the property $c_{i+1} \wedge s_i = 0$.

Standard Boolean function		Using NAND
$d_i := s_i \wedge b_i,$	$t_i := s_i \oplus b_i$	$\overline{d}_i := \overline{s_i \wedge b_i},$
$e_i := t_i \wedge k_i,$	$u_i := t_i \oplus k_i$	$\overline{e}_i := \overline{t_i \wedge k_i},$
$f_i := c_i \vee d_{i-1}$	(which are not both 1,	$f_i := \overline{\overline{c}_i \wedge \overline{d}_{i-1}}$
	due to $c_{i+1} \wedge s_i = 0$)	
$g_{i+1} := u_i \wedge f_i,$	$v_i := u_i \oplus f_i$	$\overline{g}_{i+1} := \overline{u_i \wedge f_i},$
$h_{i+1} := e_i \vee g_i$	(not both 1: $e_i = 1 \Rightarrow u_i = 0$)	$h_{i+1} := \overline{\overline{e}_i \wedge \overline{g}_i}$
$c^+_{i+1} := v_i \wedge h_i,$	$s^+_i := v_i \oplus h_i$	$\overline{c}^+_{i+1} := \overline{v_i \wedge h_i},$

4 halfadders plus two OR's, matches carry-save in GE

But: Result has the Delayed-Carry Property

$$c_{i+1} \wedge s_i = 0$$

which is crucial, when calculating μ_k fast

H/W Issues III: No Overflow with DCA

z leading MSB bits have to be in the range $-1, 0, \ldots, Z$ (assumption)
DCA: $c_{i+1} \wedge s_i = 0$, hence the following patterns are the highest values possible (shown for the case $z = 3, Z = 8$), Table 1

1	$c_{2^{l+z+1}+2,1,0;-1,-2}$		0	0	0	0	1	sum is 8 with carry, OK
	$s_{2^{l+z+1}+2,1,0;-1,-2}$		1	1	1	1	1	avoids case 4
	$M_{H,2^{l+z+1}+3,2,1,0;-1,-2}$	1	0	0	0	0	0	
2	$c_{2^{l+z+1}+2,1,0;-1,-2}$		0	0	0	1	1	sum is 8 with carry, OK
	$s_{2^{l+z+1}+2,1,0;-1,-2}$		1	1	1	1	0	avoids case 5
	$M_{H,2^{l+z+1}+3,2,1,0;-1,-2}$	1	0	0	0	0	1	
3	$c_{2^{l+z+1}+2,1,0;-1,-2}$		0	0	1	1	1	sum is 8, OK
	$s_{2^{l+z+1}+2,1,0;-1,-2}$		1	1	1	0	0	
	$M_{H,2^{l+z+1}+3,2,1,0;-1,-2}$	1	0	0	0	1	1	
4	$c_{2^{l+z+1}+2,1,0;-1,-2}$		0	1	1	1	1	sum is 9, to be avoided
	$s_{2^{l+z+1}+2,1,0;-1,-2}$		1	1	0	0	0	by case 1
	$M_{H,2^{l+z+1}+3,2,1,0;-1,-2}$	1	0	0	1	1	1	
5	$c_{2^{l+z+1}+2,1,0;-1,-2}$		1	1	1	1	1	sum is 11, to be avoided
	$s_{2^{l+z+1}+2,1,0;-1,-2}$		1	0	0	0	0	by case 2
	$M_{H,2^{l+z+1}+3,2,1,0;-1,-2}$	1	0	1	1	1	1	

H/W Issues IV: Fast computation of MUX Ctrl Vars

Per clock, add $\overline{\alpha} \cdot B$ and $\overline{\mu} \cdot K$ to DCA (c, s).
Previous 2 half cycles: Choose $\overline{\alpha} \cdot B$ and $\overline{\mu} \cdot K$ by the same H/W.
time-critical only for $\overline{\mu}$: Depends on the addition just performed in the half cycle $(k+1, H)$.

Cycle	Half C.	Selection	Computation
k	H	$\overline{\alpha}_k \cdot B$	$(M_H \vert M_L)_k := \dots$
k	L	$\overline{\mu}_k \cdot K$	
$k+1$	H	$\overline{\alpha}_{k+1} \cdot B$	$(M_H \vert M_L)_{k+1} := ((M_L)_k << z) + \overline{\alpha}_k \cdot B + \overline{\mu}_k$
$k+1$	L	$\overline{\mu}_{k+1} \cdot K$	

Precompute M_H positions:
1. In (k, H), partial sum $(M_H)_k \cdot Z + \overline{\alpha}_k \cdot B$
2. In (k, L), add $\overline{\mu}_k \cdot K$, for M_H bit positions.
3. Also add 0,1,2,3: Possible final values for $\overline{\mu}_{k+1}$, precompute the MUX control vars (4 sets) for $\overline{\mu} \cdot K$.
4. In $(k+1, H)$, choose by MUX via carries from M_L part.
5. In $(k+1, L)$: Ready to fetch $\overline{\mu}_{k+1} \cdot K$ from one of the 4 sets.

(FPGA with 6:1 LUTs: Addition maybe (even) faster than CTRL)

DCA and Timing Attacks

Final carry from DCA to standard representation:
Either

(*i*) we use carry-look-ahead logic, space-intensive, or

(*ii*) we keep the result in delayed-carry-form, space-intensive, or

(*iii*) we wait until the longest carry chain ($L + z$ bits) will have passed, time-intensive, or

(*iv*) we use interrupt techniques, efficient, but time-variant.

The variation due to carries in case (*iv*) is the only potential information leak for a timing attack. This is though independent of Reduce-by-Feedback (or Montgomery multiplication), but a consequence of using carry-save or delayed-carry techniques.

DPA attack on RSA with MM or Reduce-by-Feedback:

Before first cycle:

$$M = 0, M_H = 0, \mu = 0$$

At first cycle:

$$M^+ := (M << 3) + \alpha \cdot B + \mu \cdot K = 0 + \alpha B + 0$$

IF $\alpha = 0$ (*i.e.* *A* starts with 3 zeroes):
$M^+ := 0 + 0 + 0 = M$, NO change of FF charges

IF $\alpha \neq 0$ (*i.e.*the other 7 cases):
$M^+ := 0 + \alpha B + 0 \neq 0 = M, \approx 50\%$ of FF go $0 \rightarrow 1$
(same effect for Reduce-by-Feedback and Montgomery)

We observe (only) this “point-of-interest”

DPA on RSA II

Run C trials with different m, same (unblinded) exponent d:
Observe $L \cdot 1.5$ mult./squarings per trial
Information content / Entropy per trial:

$$H = -(1/8 \cdot \log_2(1/8) + 7/8 \cdot \log_2(7/8)) = 0.544$$

We have 1.5 observations per bit of d, thus $1.5 \cdot 0.544 = 0.816$ bits, recovering 81% of d's bits, or with $C = 2$, everything!! Or do we????

Crucial, difficult case is "always $\alpha \neq 0$", the "big bin"
This bin has to contain only a single solution, no false positives:

$$2^L \cdot \left(\frac{7}{8}\right)^{1.5L \cdot C} = 1$$

or

$$\left(\frac{7}{8}\right)^{1.5 \cdot C} = \frac{1}{2} \Leftrightarrow C = 3.47$$

So we actually need 4 trials in this worst and quite typical case.

DPA on RSA III

Run 4 decryptions with known *m*'s (DUT)

Simulate for all possible prefixes for *d*,
compare occurrence of $\alpha = 0$ vs. $\alpha \neq 0$ with actual DUT

Throw away non-fitting prefixes, enlarge the survivors
(we usually should have about just one survivor)

And that breaks RSA!

How to fix it

$\alpha_0 = \mu_0 = 0$ is exploitable by DPA

1. (NEW!) Both Reduce-by-Feedback and Montgomery
Start with $M = N$, not $M = 0$
(more H/W, additional MUX input, not just Reset)

2. (NEW!) Montgomery
For $M = ..000$, add $8 \cdot N$, not $0 \cdot N$

3. Reduce-by-Feedback
$M = 0 \mapsto M^+ = 0$ can be avoided, use “1-off” trick with

$$0 = 1 + (-1)$$

Instead of $0 \cdot B$, add B once, subtract $Z \cdot B$ in the next step.
This brings us back to zero every second step.

B has $\approx 50\%$ 1’s: Flips back-and-forth half of the register bits

On the outside: typ. power consumption, no side channel

Example with $z = 3, Z = 8$

Old: regular "1-off" case including a multiple 0.
New: $0 = 1 + (-1)$, also $\Sigma = -1, 1, 2$, and 3 differently
Minimize the information flow (bias) from $\overline{\alpha}, \overline{\mu}$ to C, A, M_H
Irregular "1-off" + Shifts. Still only $Z/4$ values phys. stored, *e.g.* 6;8.

C_α, C_μ	α_k, M_H	Σ	$\overline{\alpha}_k$, $\overline{\mu}_k$(old)	C^+	$\overline{\alpha}_k$, $\overline{\mu}_k$(new)	C^+
0	-1	-1	0	1	1	0
0	000	0	0	0	1	1
0	001	1	2	1	1	0
0	010	2	2	0	3	1
0	011	3	4	1	3	0
0	100	4	4	0	4	0
0	101	5	6	1	6	1
0	110	6	6	0	6	0
0	111	7	8	1	8	1
0	1000	8	8	0	8	0

Bias: Nearer zero

Bias = pr(1) $-$ pr(0)

Bias of C and Σ (internals, partly revealing A and M),
conditional on certain value sets for $\overline{\alpha}, \overline{\mu}$,
namely zero, positive, shifts of 8, and shifts of 6
(potentially observable by DPA):

Assumed probabilities:
C: pr = $1/2$ for $C = 0$ and $C = 1$
α: Pr = $1/8$ each for $\alpha = 0, \ldots, 7$.
μ: Fold 3 equidistributions over the intervals
$[0, 8[$ (from M_H) ,
$[-1/2, 1/2[$ (from $\alpha \cdot B$), and
$[-1/2, 1/2[$ (from $\mu \cdot K$),

giving
Pr = $1/8$ each for $\mu = 1, \ldots, 6$,
Pr = $5/48$ for $\mu = 0$ and 7, and

Bias II

We now have probability zero for $\overline{\alpha} = 0$, which was $1/8$ before.

Sets $\{1, 2, 4, 8\}$ and $\{3, 6\}$ for α, μ give zero bias (all bits of C, Σ).

For α, μ positive, the bias shrinks:

	C	Σ_2	Σ_1	Σ_0
$\overline{\alpha} > 0$ new	-1	0	0	0
$\overline{\alpha} > 0$ old	-1	$1/7$	$1/7$	$1/7$
$\overline{\mu} > 0$ new	$-23/24$	$1/24$	$1/24$	$1/24$
$\overline{\mu} > 0$ old	-1	$-2/21$	$-2/21$	$-2/21$

Table : Bias of C, Σ, conditional on $\overline{\alpha}, \overline{\mu}$

The remaining strong bias -1 is from $\overline{\alpha}, \overline{\mu}$ positive to $C = 0$ (or ... negative to $C = 1$), almost a tautology.

$\overline{\alpha}, \overline{\mu} > 0$: mix of cases $1, 2, 3, 4, 6, 8$, quite more difficult to analyze by DPA than the distinction $\alpha = 0$ vs. $\alpha \neq 0$, now ruled out.

Conclusion

Reduce-by-Feedback has all the advantages of
Montgomery Multiplication (for full-length register addition),

in particular, timing invariance, and 75% savings in physical storage.

Additionally Reduce-by-Feedback enjoys the analogy of Shift-and-Add with Reduce-by-Add, saves up to 50% logic/MUXes by re-use.

— — — — — — — — —

Avoid an empty accumulator, start with N, not zero, or ...

avoid the occurrence of $M^+ := (0 << 3) + 0 + 0 = M$ in the first cycle, otherwise ...

(unblinded) RSA can be broken with 4 (or less) observed decryptions for an implementation of 3 (or less) bits/cycle