Efficient and Side-channel Resistant RSA Implementation For 8-bit AVR Microcontrollers

Zhe Liu[1], Johann Großschädl[2], Ilya Kizhvatov[2]

[1]School of Computer Science and Technology, Shandong University
[2]Laboratory of Algorithmics, Cryptology and Security, University of Luxembourg

Nov-29-2010

Our contributions

- High-speed RSA implementation for 8-bit microcontroller
  - New hybrid method for multi-precision multiplication.
  - Our RSA implementation is faster than previous implementation.
- Side-channel Attacks and Countermeasures
  - Simple Power Analysis Attack (SPA), Differential Power Analysis Attack (DPA).
  - SPA, DPA Low-cost countermeasures.

Outline

Outline

Outline

Motivation

- Current Internet is secured by RSA-based PKI, CAs and security protocols.
- "Internet of things": Client (sensor nodes, RFID tags) authentication must be performed via private-key operations.
- Efficient RSA implementation for embedded devices plays a vital role in the expansion of well established security protocols to Internet of Things.

RSA

- Given a message M and the receiver's public-key (N, e), compute the cipher text:

$$C = M^e \bmod N$$

- Given cipher text C, to recover M, compute:

$$M = C^d \bmod N$$

8-bit Microcontroller

- Widely used in real world, small, cheap and convenient.
- 80 percent market share in smart cards.
- AVR ATmega is high-performance, low-power 8-bit microcontroller provided by Atmel corporation.

- 128 kB in-system programmable flash memory.
- 4 kB internal SRAM.
- Maximum frequency 16 MHz.
- 32×8-bit general purpose working registers.

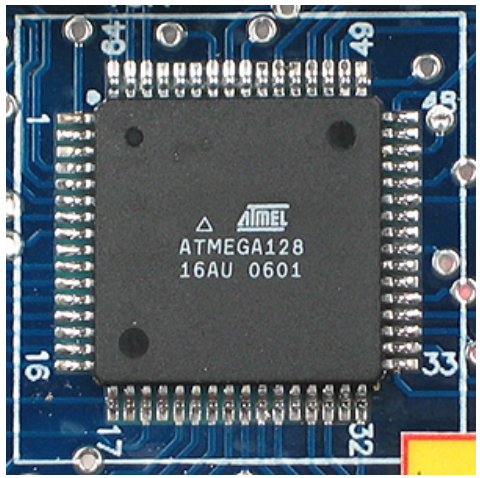


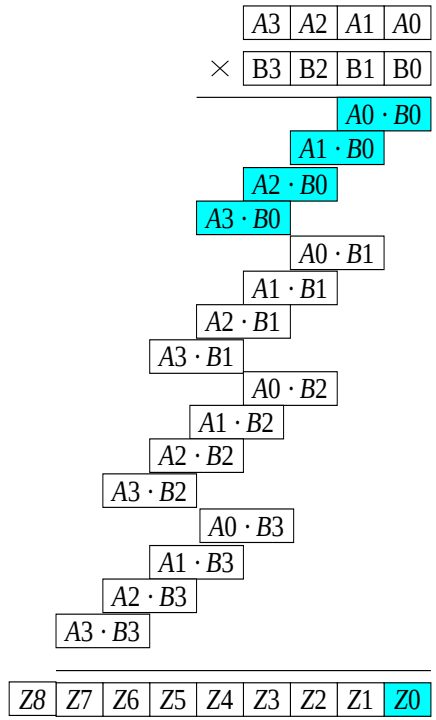
Figure: ATmega-128

Outline

School-book method

Operand Scanning method, Row-wise method, Pencil-and-Paper method.

- The outer loop moves through the words of an operand.
- In each outer loop, a word of multiplier B is multiplied by all words of multiplicand A.
- Each inner loop: $(u, v) \leftarrow a_j \times b_i + r_{i+j} + u$.
- Example: 4-words.

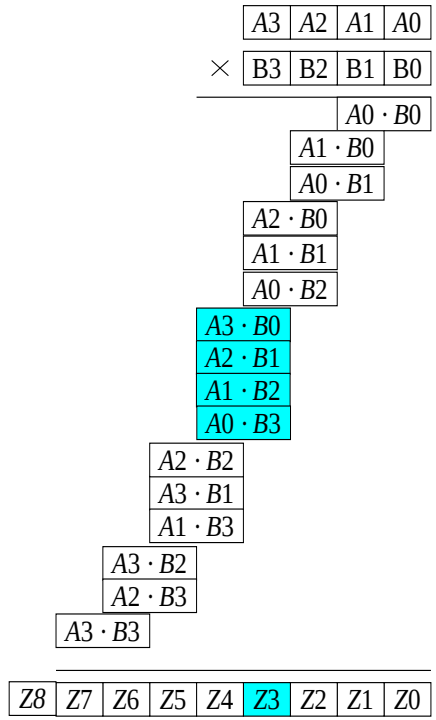


School-book Multiplication

Comba's method

Product Scanning method, Column-wise method. [Comba, IBM90]

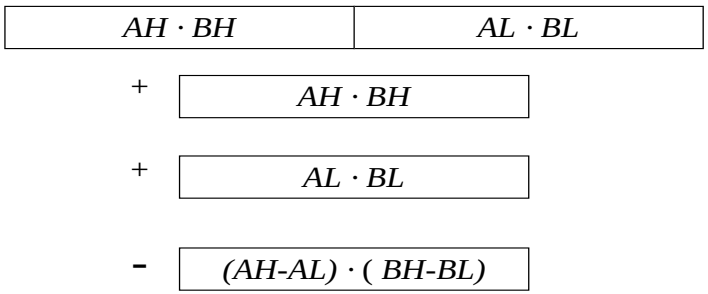
- Two outer loops move through the product R.
- Each inner loop: $(t, u, v) \leftarrow (t, u, v) + a_j \times b_{i-j}$.
- Better performance than the school-book method in assembly language (less add, less store).
- Example: 4-words.



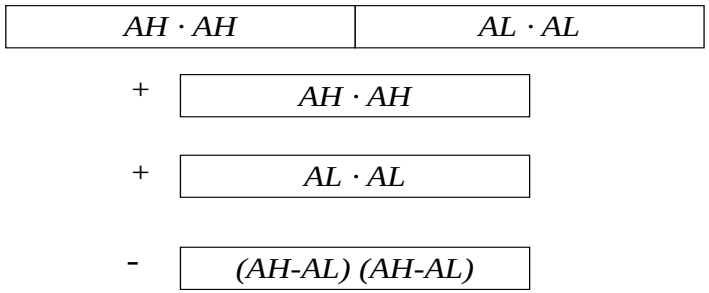
Comba Multiplication

Karatsuba's method

- Reduces a multiplication of two s-word operands to three multiplications of size $\frac{s}{2}$. ($\frac{3s^2}{4}$ single-precision multiplications). [Karatsuba and Ofman, 1962]
- The half-size multiplications can be performed by any multiplication method.
- $A = A_H \times 2^{w/2} + A_L$, A_H and A_L are the higher and lower half-part of A.
- Be careful with the 'subtraction'.



Multiplication



Squaring

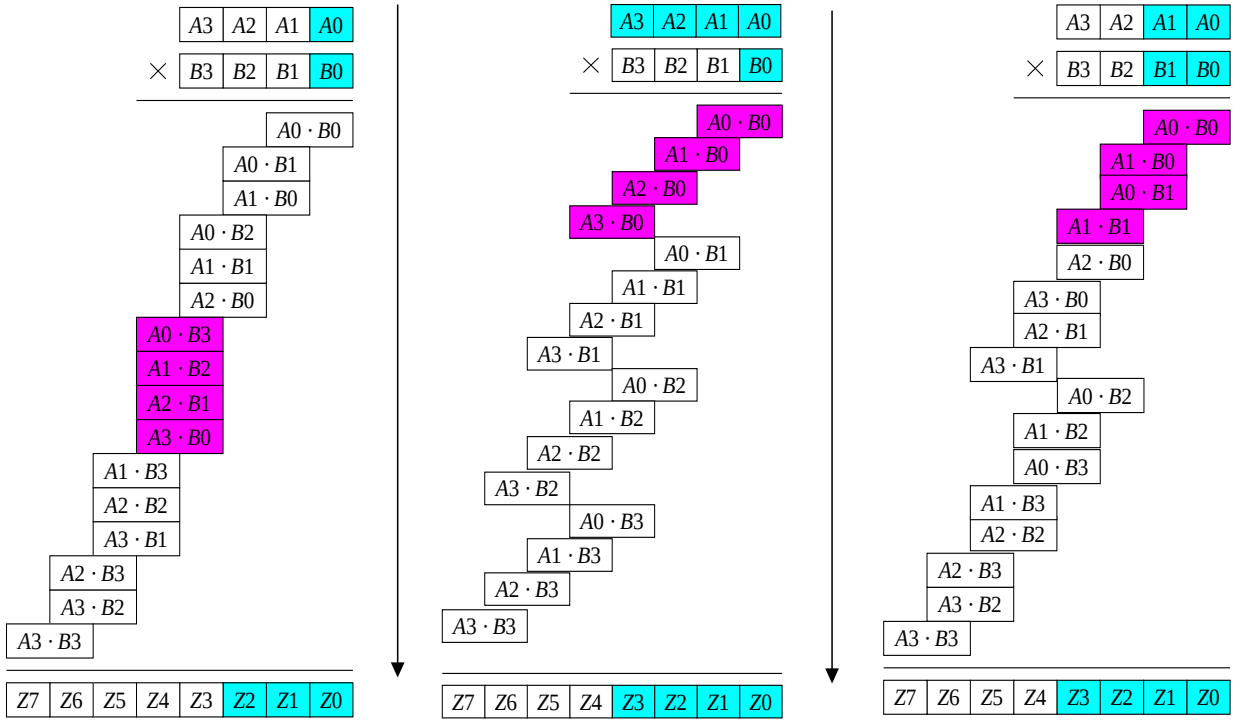
Outline

Hybrid Multiplication

- Hybrid method optimizes both memory accesses and register usage. [Gura, CHES'04]
  - Comba's method needs less accumulator registers, but reloads operands; School-book requires less load operations.
  - Hybrid multiplication computes columns that consist of rows of partial products.
  - Register usage and memory accesses depend on d, the width of the column.
  - $d = 1$: Comba method; $d = n$: school-book method.

Hybrid Multiplication(d = 2)

- Example: 4-words, d = 2.



Comba multiplication School-book multiplication Hybrid multiplication

Montgomery Modular Multiplication

- An efficient method for performing modular multiplication with an odd modulus. [Montgomery, 1985]
  - Transform mod N to mod $R = 2^w$, replaces the trial division with simple shift operations.
- Montgomery multiplication and Montgomery reduction.
  - The Separated Operand Scanning (SOS) method. Reduction after multiplication. [Koc, 1996]
  - The Finely Integrated Product Scnning (FIPS) method and Coarsely Integrated Operand Scanning (CIOS). Reduction and multiplication interleaved. [Koc, 1996]
  - The Karatsuba-Comba-Montgomery reduction (KCM). [Großschädl, CHES'05]

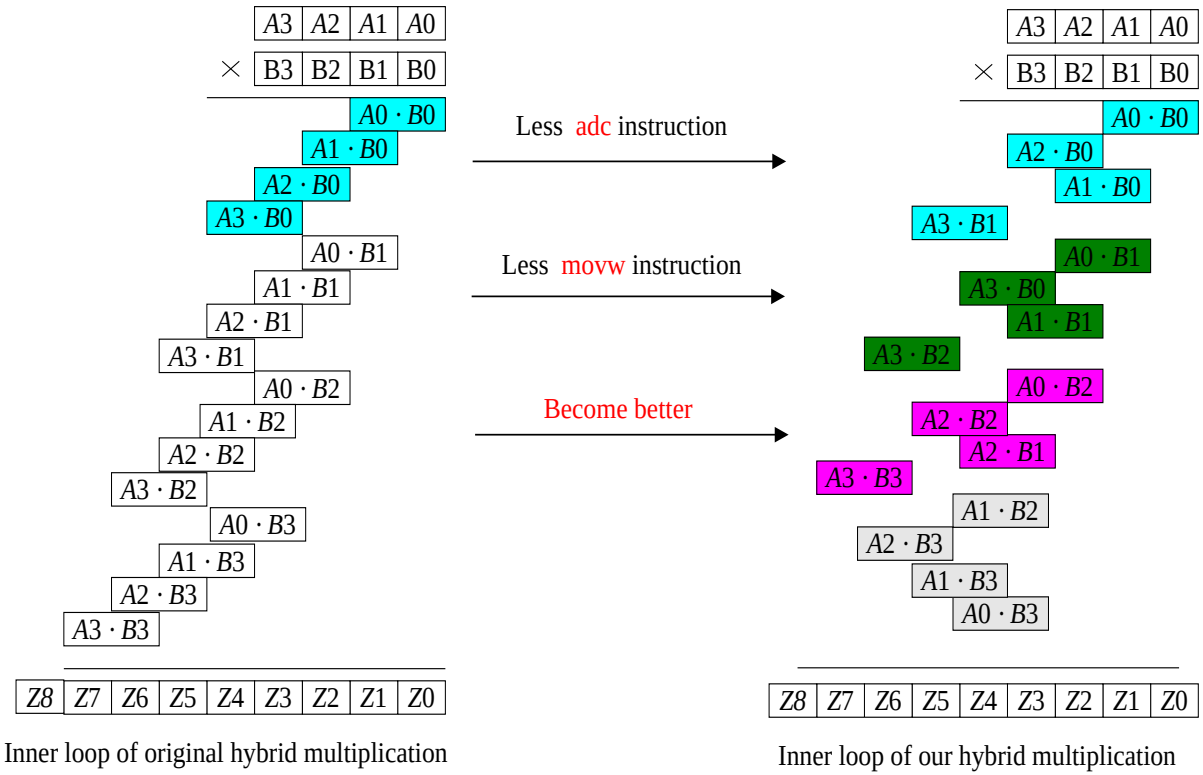
Modular Exponentiation

- The m-ary method is based on m-ary expansion of the exponent e, aims to reduce the number of multiplications.
- Algorithm works as follows (m=16), e.g. $M^{10011101}$ mod N:
  - Generate look-up table.
  - Do four squarings.
  - Look up table, and do one multiplication.
- Compared to using the square and multiply method, the m-ary method with m=16 reduces the execution time by 15.2%.

Outline

Outline

Our Hybrid Multiplication (d = 4)



Inner loop of original hybrid multiplication

Inner loop of our hybrid multiplication

Comparison Result and Analysis

Instruction	*add*	*mul*	*ld*	*st*	*mov*	other	Total
CPI	1	2	2	2	1		
Classic Comba	1200	400	800	40	81	44	3805
Gura *et al*	1360	400	167	40	355	197	3106
Uhsadel *et al*	986	400	238	40	355	184	2881
Scott *et al*	1263	400	200	40	70	38	2651
Our work	1194	400	200	40	212	179	2865

Table: Comparison of instruction counts for 160-bit multiplication on ATmega-128

- Our method is better than the most efficient published version, but slightly slower than that of Scott, who unrolled the loop completely.

Outline

Hybrid Montgomery Multiplication

<table>
<tr><td>Karatsuba squaring
Karatsuba multiplication

Comba squaring
Comba multiplication

Our Improved method

Mul</td><td>Comba squaring
Comba multiplication

Our Improved method

Mul</td><td rowspan="2">Montgomery Squaring

Montgomery Multiplication</td></tr>
<tr><td>Montgomery Reduction</td><td>Montgomery Reduction</td></tr>
<tr><td>HKCM</td><td>HSOS
HFIPS</td><td>CIOS</td></tr>
</table>

Figure: Hybrid Montgomery Multiplication

Analysis and Comparison

Table: Execution time of HSOS, HFIPS, and HKCM (in clock cycles).

Algorithm	256 bit	512 bit	768 bit	1024 bit
HSOS ($d = 4$)	18655	65649	141585	246462
HFIPS ($d = 4$)	19727	70592	153007	266975
HKCM ($d = 4$)	20994	66142	135901	232450

- HSOS is more efficient than HFIPS mainly due to the massive register usage of the hybrid method.
- The HKCM method is faster than HSOS (and also HFIPS) for operands exceeding 512 bits in length.
- Optimizations for Montgomery squaring can be easily integrated into both HSOS and HKCM.

Performance Evaluation of 1024-bit RSA

- CRT + m-ary + HSOS method.

Implementation	Cycle count	Notes
Deng et al	$60.0 \cdot 10^6$	C, public-key operation
Watro et al	$58.0 \cdot 10^6$	C, public-key operation
Gura et al	$87.92 \cdot 10^6$	C and assembly, CRT
Wander et al	$88.0 \cdot 10^6$	C and assembly, CRT
Wang et al	$172.0 \cdot 10^6$	C and assembly, CRT
Our work	$75.68 \cdot 10^6$	C and assembly,CRT

Table: Performance of 1024-bit RSA implementations

- Our implementation sets a new speed record for RSA on an 8-bit microcontroller.

Outline

Power Analysis

- Based on the information gained from the physical implementation of a cryptosystem.

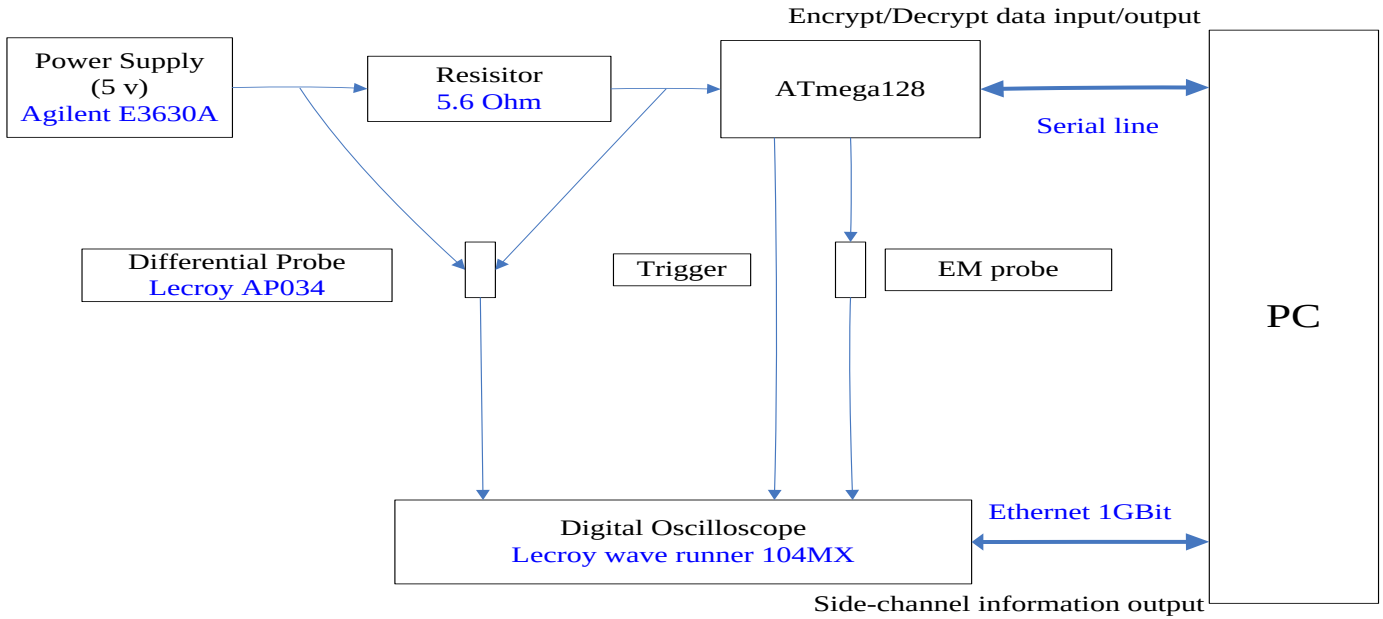
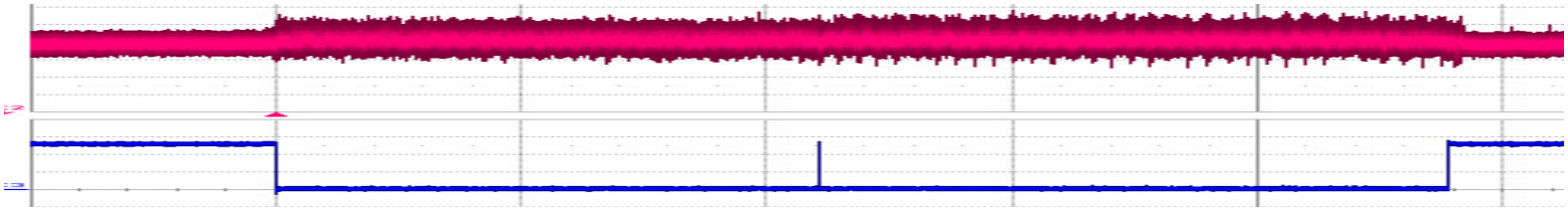


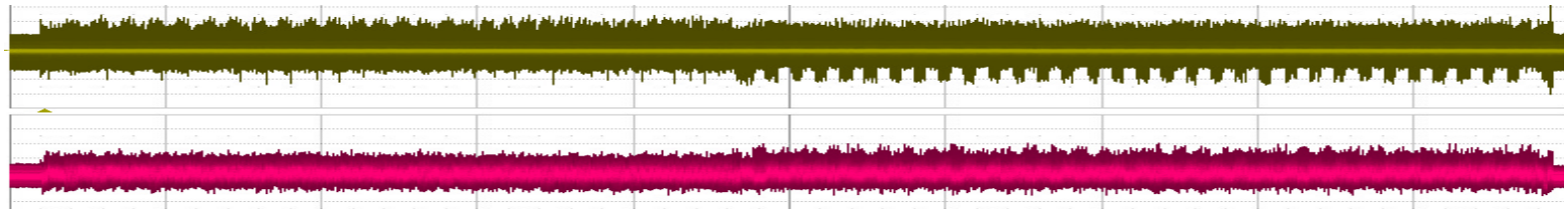
Figure: Block diagram of our measurement setup for power analysis attack

Outline

Simple Power Analysis Attack



- Difference in execution time between squaring and multiplication is 0.6 ms. (Each division is 2 ms)



- Electromagnetic leakage is more informative in the case of SPA.

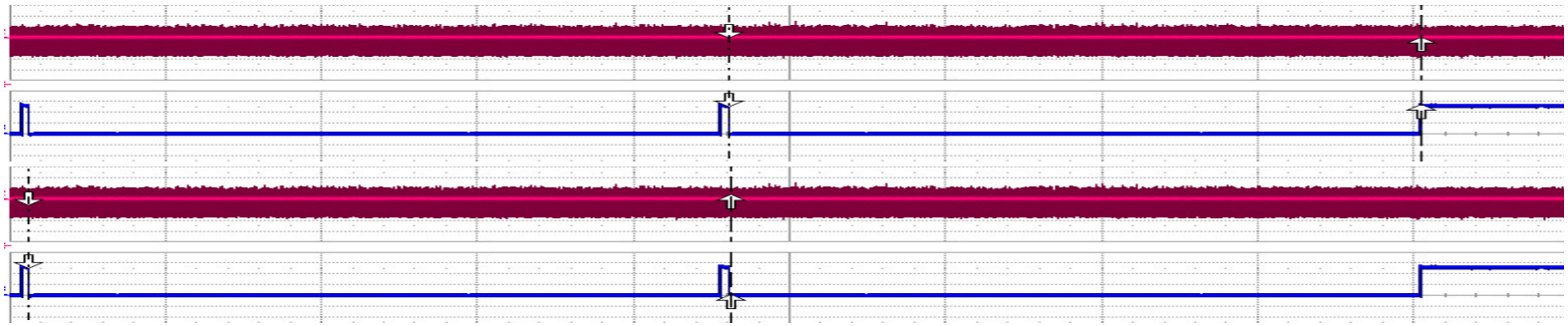
Distinguish Squaring from Multiplication

- Modular exponentiation power trace (Each division is 10 ms):

- Conclusion:
  - We can distinguish the squaring from multiplication. From above figure, we can recover that the exponent is 000000011110110.

Distinguish Final Subtraction

- Conditional subtraction in Montgomery algorithm is vulnerable to side-channel attack (Theoretical analysis). [Colin D. Walter, CT-RSA, 2001]



- In our implementation the time difference due to final subtraction is around 0.07 ms.

Low-cost Countermeasures for SPA

- The vulnerabilities of the implemented algorithm are essentially because of conditional statements.
  - We can avoid using a conditional statement in the exponentiation, as we do in our variant of the m-ary method.
  - We can add an *else* statement whenever there is a conditional statement in the modular multiplication.

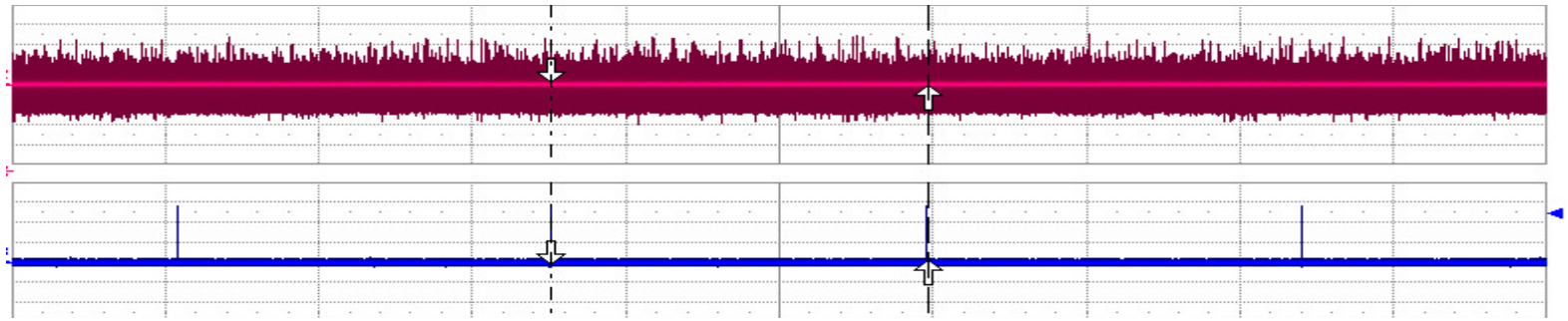


Figure: SPA protected version of the m-ary method

Outline

Differential Power Analysis Attack

- SPA attack needs to know some details about the implemented algorithm and it is difficult to directly observe the key (often obscured with noise and modulated by the device's clock signal).
- DPA is more powerful, uses statistical methods to deal with a large number of power traces, needs less details.
- We will apply Multiple Exponent Single Data (MESD) attack to our implementation. [Thomas S. Messerges, CHES'99]
  - MESD breaks the unprotected version of square and multiply method.
  - MESD breaks the m-ary method with SPA countermeasures.

MESD Attack

- Our experimental MESD attack includes the following steps:
  1. Choose one message a as constant input of each exponentiation.
  2. Collect 10 power traces of exponentiations of a by the secret exponent e_s.
  3. Compress and average the traces.
  $$S = \frac{1}{10} \sum_{i=1}^{10} S_i$$
  4. Guess the the exponent $e_g = FF..FF$ and collect 10 power traces exponentiated by the exponent $e_g = FF..FF$.
  5. Compress and average the traces.
  $$K = \frac{1}{10} \sum_{i=1}^{10} K_i$$
  6. Subtract the two mean traces S and K, get the DPA bias trace.
  $$D_j = S_j - K_j$$
  7. Guess another exponent and repeat step 4 to step 6.

Attack Square and Multiply method (1)

Guessed key (FF)	S	M	S	M	S	M	S	M	S	M	S	M	S	M	S	M
Difference	0	0	0	0	0	0	0	1								
Secret-key	S	M	S	M	S	M	S	S								

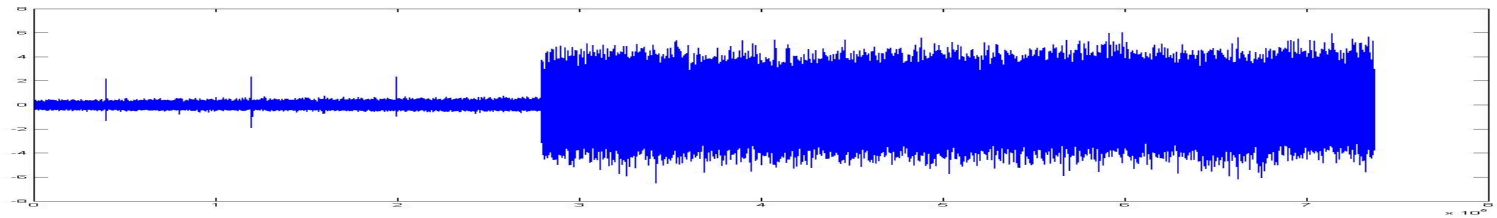


Figure: MESD Result of the first guess $e_g = FF...$

Attack Square and Multiply method (2)

Guessed key(7B)	S	M	S	M	S	M	S	S	M	S	M	S	M	S	M	S
Difference	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Secret-key	S	M	S	M	S	M	S	S	M	S	M	S	M	S	S	

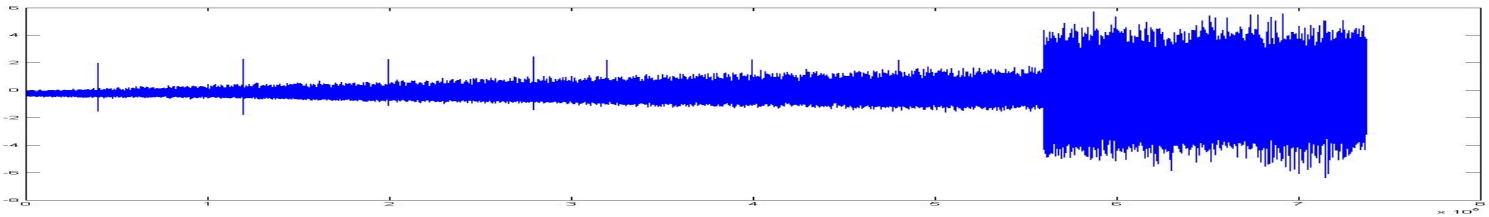


Figure: MESD Result of the second guess $e_g = 7B$...

Attack Square and Multiply method (3)

Guessed key(7BB)	S	M	S	M	S	M	S	S	M	S	M	S	M	S	S	M
Difference	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Secret-key	S	M	S	M	S	M	S	S	M	S	M	S	M	S	S	S

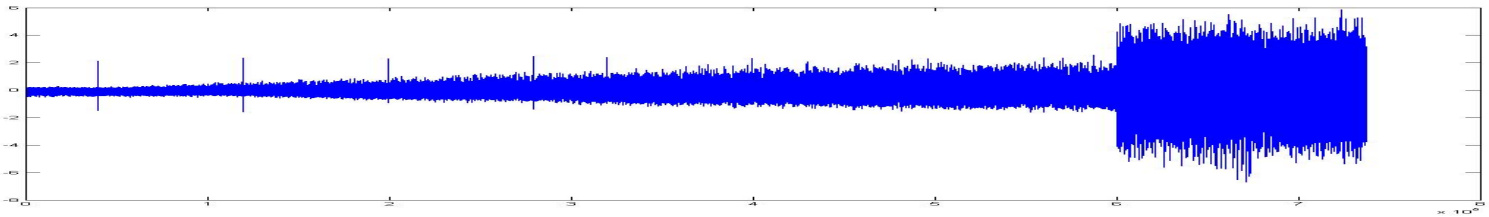


Figure: MESD Result of the third guess $e_g = 7BB...$

Attack SPA Protected Version m-ary method (1)

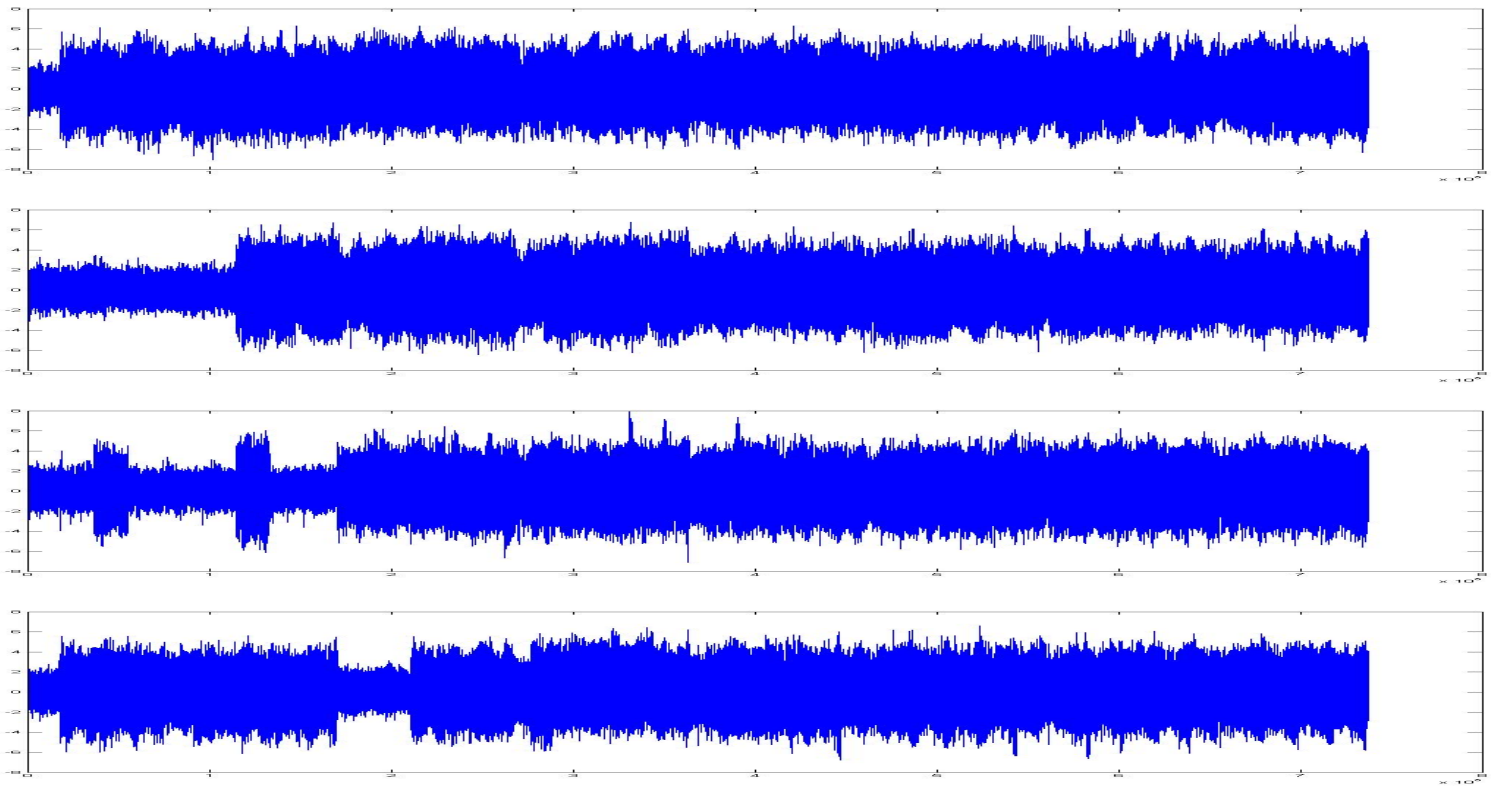


Figure: DPA bias traces from $e_g = 0...FF$ to $e_g = 3...FF$

Attack SPA Protected Version m-ary method (2)

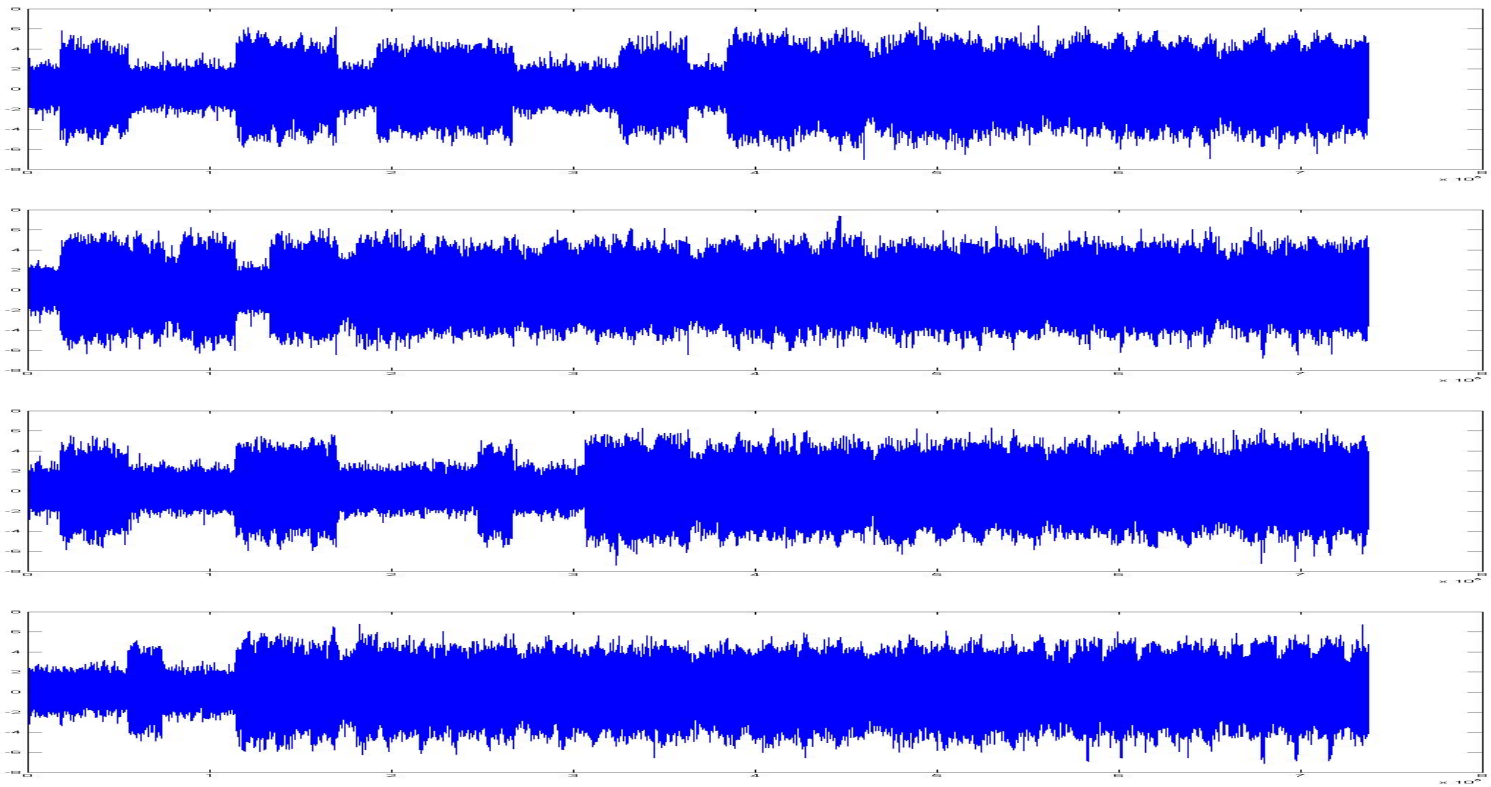


Figure: DPA bias traces from $e_g = 4...FF$ to $e_g = 7...FF$

Attack SPA Protected Version m-ary method (3)

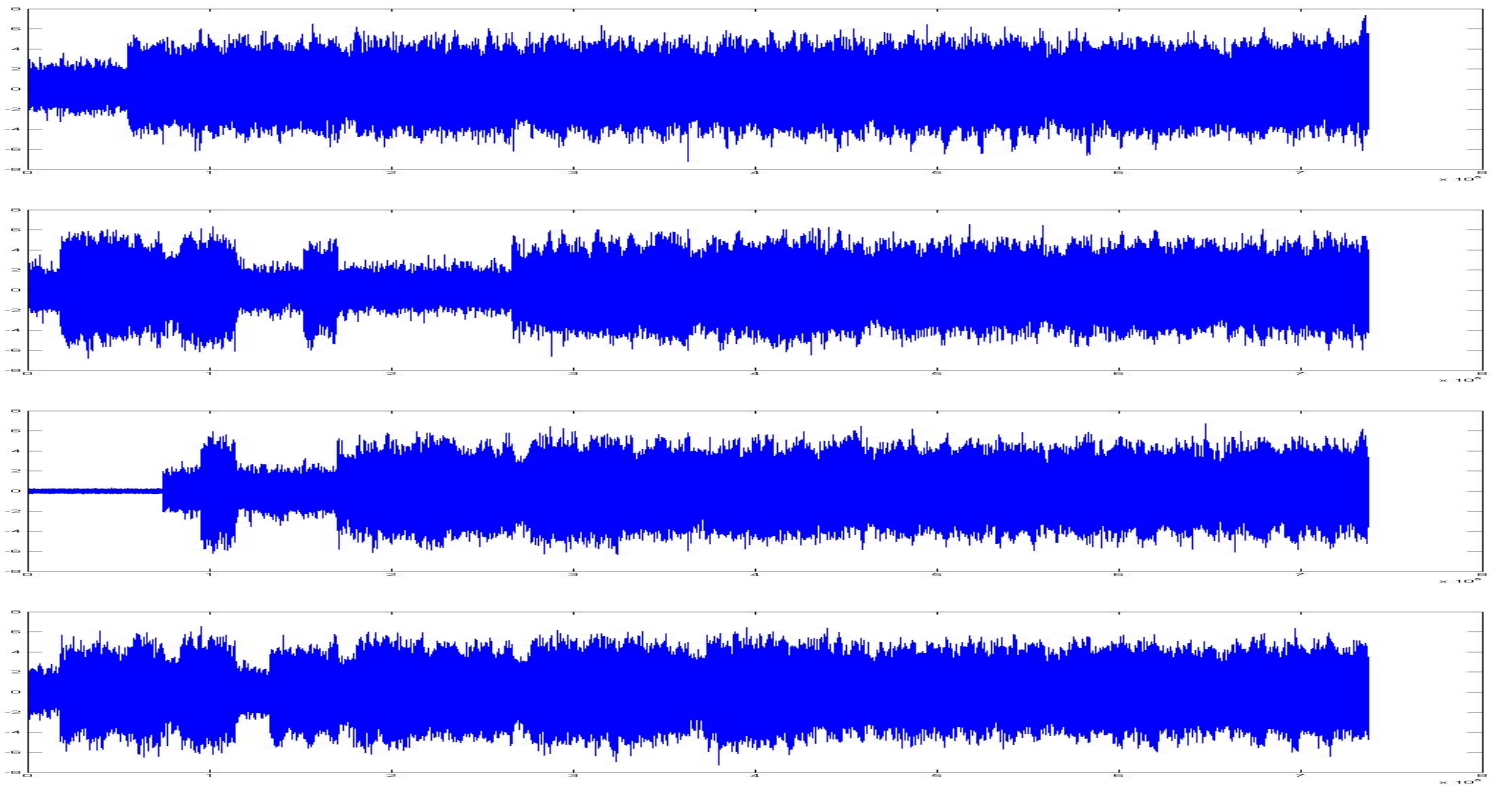


Figure: DPA bias traces from $e_g = 8...FF$ to $e_g = B...FF$

Attack SPA Protected Version m-ary method(4)

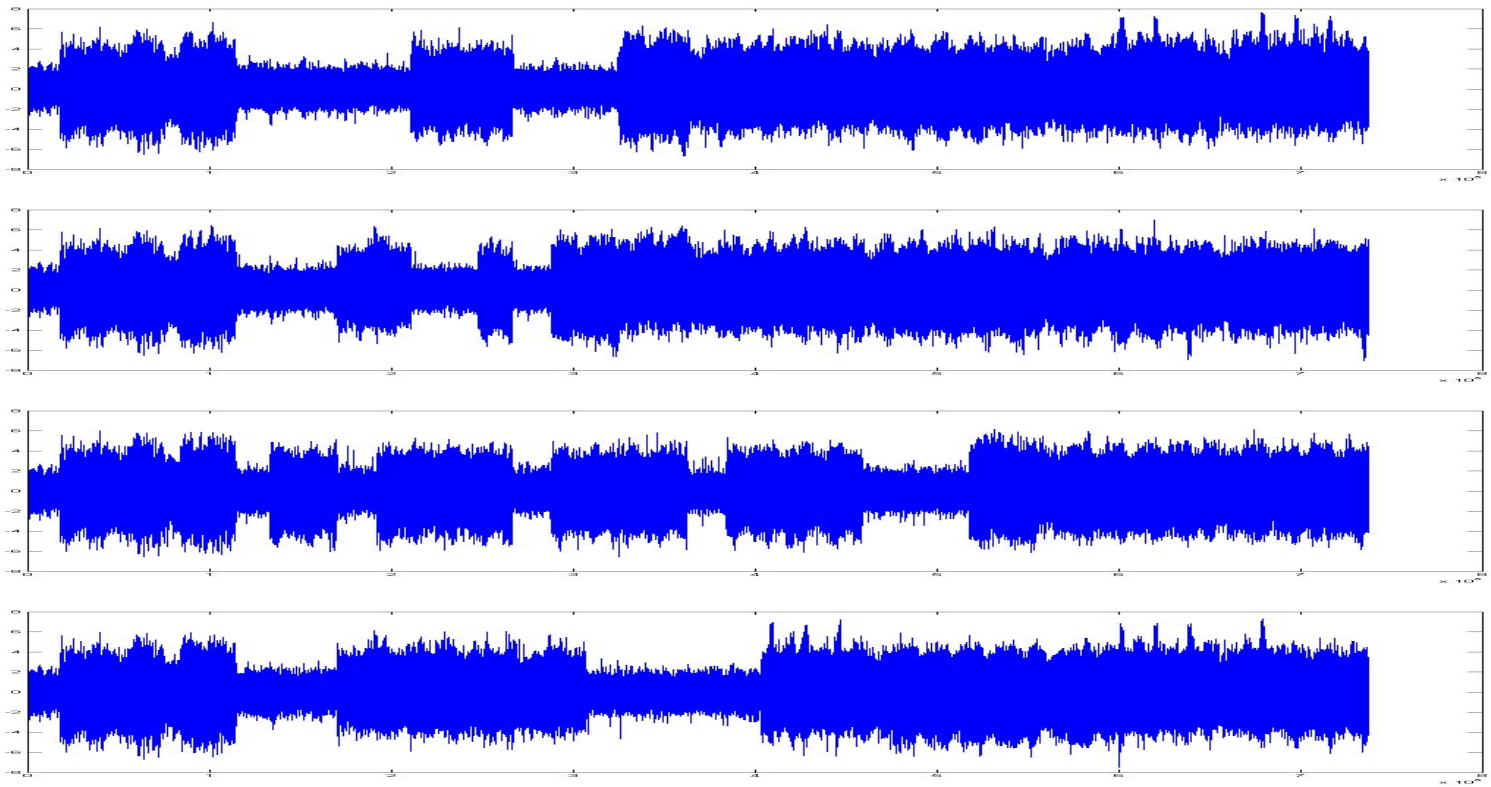


Figure: DPA bias traces from $e_g = C...FF$ to $e_g = F...FF$

MESD Attack

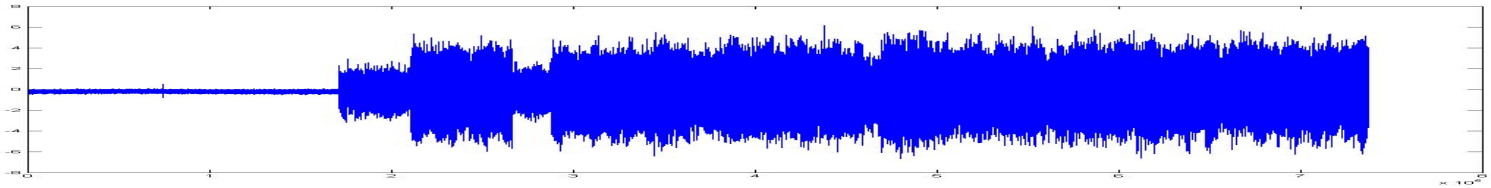


Figure: DPA bias trace with $e_g = A4...FF$

- Our MESD attack is successful, and this attack is significantly faster than the exhaustive key search.
- We can recover the secret exponent bit by bit for the square and multiply method (512 times for 256-RSA).
- We can recover the secret exponent window by window for the m-ary method, even the version with SPA countermeasures (1024 times for 256-RSA).

Countermeasures for DPA

- Blinding the message M and blinding the exponent e. [Kocher 96, Coron 99]
- The exponentiation process would go as follows:
  - Blind the message M using v_i co-prime with the modulus n: $M' = (v_i \times M) \bmod n$.
  - Blind the exponent e using a random number r (for 1024-bit RSA, the length of r is usually 32 bits)$e' = e + r\phi n$.
  - Do exponentiation after blinding:$C' = (M'^{e'}) \bmod n$.
  - After receiving the message C', unblind the message C, whereby $v_f = (v_i^{-1})^e \bmod n$: $C = v_f \times C' \bmod n$.

Conclusion

- Our contributions are the followings:
  - Our new hybrid multiplication saves add/adc and mov instructions, our 160×160-bit multiplication is 7.8% faster than Gura's.
  - Taking advantage of the CRT, m-ary method and our HSOS method, our RSA implementation represents a new speed record for 1024-RSA on 8-bit controllers.
  - Our efficient RSA implementation can resist power analysis attack (i.e. SPA, DPA) after adding low cost countermeasures.
  - Our implementation satisfies all requirements to expand the security protocols and services of the current Internet to the "Internet of Things".

Conclusion

- Our contributions are the followings:
  - Our new hybrid multiplication saves add/adc and mov instructions, our 160×160-bit multiplication is 7.8% faster than Gura's.
  - Taking advantage of the CRT, m-ary method and our HSOS method, our RSA implementation represents a new speed record for 1024-RSA on 8-bit controllers.
  - Our efficient RSA implementation can resist power analysis attack (i.e. SPA, DPA) after adding low cost countermeasures.
  - Our implementation satisfies all requirements to expand the security protocols and services of the current Internet to the "Internet of Things".

Conclusion

- Our contributions are the followings:
  - Our new hybrid multiplication saves add/adc and mov instructions, our 160×160-bit multiplication is 7.8% faster than Gura's.
  - Taking advantage of the CRT, m-ary method and our HSOS method, our RSA implementation represents a new speed record for 1024-RSA on 8-bit controllers.
  - Our efficient RSA implementation can resist power analysis attack (i.e. SPA, DPA) after adding low cost countermeasures.
  - Our implementation satisfies all requirements to expand the security protocols and services of the current Internet to the "Internet of Things".

Conclusion

- Our contributions are the followings:
  - Our new hybrid multiplication saves add/adc and mov instructions, our 160×160-bit multiplication is 7.8% faster than Gura's.
  - Taking advantage of the CRT, m-ary method and our HSOS method, our RSA implementation represents a new speed record for 1024-RSA on 8-bit controllers.
  - Our efficient RSA implementation can resist power analysis attack (i.e. SPA, DPA) after adding low cost countermeasures.
  - Our implementation satisfies all requirements to expand the security protocols and services of the current Internet to the "Internet of Things".

Conclusion

- Our contributions are the followings:
  - Our new hybrid multiplication saves add/adc and mov instructions, our 160×160-bit multiplication is 7.8% faster than Gura's.
  - Taking advantage of the CRT, m-ary method and our HSOS method, our RSA implementation represents a new speed record for 1024-RSA on 8-bit controllers.
  - Our efficient RSA implementation can resist power analysis attack (i.e. SPA, DPA) after adding low cost countermeasures.
  - Our implementation satisfies all requirements to expand the security protocols and services of the current Internet to the "Internet of Things".

Thanks and Questions

Thanks for your attention!

Questions?