# 4096-Bit RSA Implementation and Its Side Channel Attack Countermeasures on FPGA

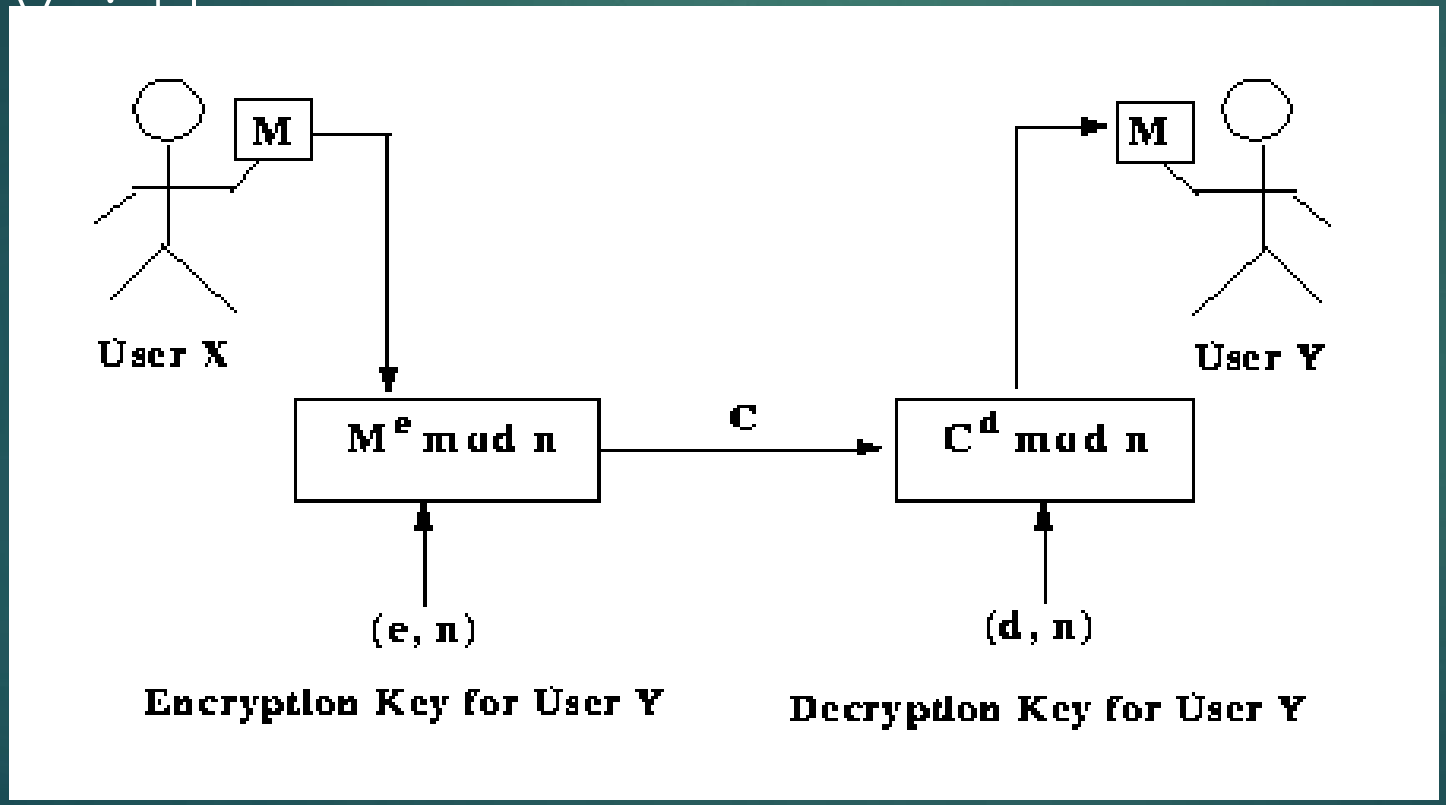
QIN ZHOU

ADVISORS: PROF. CETIN KOC, PROF. TIM SHERWOOD

# RSA Cryptosystem



- Well-known public-key cryptosystem[1].
-
-
-



# Why do we make $n$ 4096-bit?



- To guarantee security, large number is needed .
- 1024-bit RSA has been cracked[3] in 2010.
- According to RSA laboratory, 2048-bit is recommended since 1369-bit is possibly being cracked today[4].
- We believe 4096-bit can be secure for longer time. Our solution can make contributions to the industry.

# Why implement on FPGA (programmable hardware)?

- Reduce time usage (clock cycle usage) on large number computing
  - CPU today computes 64 bits in one clock cycle while FPGA can compute hundreds of bits in a cycle.
- Better security
  - Since it is another piece of hardware, normally it looks like a black box to hacker.
  - Computation is different compared to CPU implementation, harder to reverse engineering.
- Better analysis
  - On simulation tool every clock cycle is transparent.

# Basic implementation settings of 4096-Bit RSA ($m = c^d \ (mod\ n\ )$)

- The 4096-bit numbers are organized as $sw$ bits, where $s$ is the number of words and $w$ is the word length[5], $sw = 4096$.
- $s = 32, w = 128. n = n_{31}n_{30} \dots n_1n_0, \quad n_0 = n_{0,127}n_{0,126} \dots n_{0,1}n_{0,0}$
- Primary Input: $c$
- Secondary Input: $d, n, n_0' = -n_o^{-1}\ mod\ 2^w, r = 2^{sw}, t = 2^{2sw}$
- Output: $m$
- Using Verilog HDL, a hardware language
- IDE: Quartus II 12.1 64-Bit
- Target Device: Cyclone II EP2C50
  - A very resource limited device

# RSA Algorithm: The Binary ModExp (Modular Exponentiation)



- find the index $k < 4096$ of the leftmost 1 in $d$
- $\overline{c} = MonPro(c, t)$
- $\overline{m} = r$
- for $i = k - 1$ down to $0$
  - $\overline{m} = MonPro(\overline{m}, \overline{m})$ (square)
  - if $d_i = 1$ then $\overline{m} = MonPro(\overline{m}, \overline{c})$ (multiplication)
- $m = MonPro(\overline{m}, 1)$
- What is $MonPro$?

# MonPro: Montgomery Product



- $MonPro(a,b) = a \cdot b \cdot r^{-1} \ (mod\ n),\ (r = 2^{sw}, n < r)$
  - To get $u = a \cdot b \ (mod\ n)$
  - $\overline{u} = MonPro(a,b) = a \cdot b \cdot r^{-1} (mod\ n)$
  - $u = MonPro(\overline{u}, t) = \overline{u} \cdot t \cdot r^{-1} = a \cdot b \cdot r^{-1} \cdot r^2 \cdot r^{-1} = a \cdot b (mod\ n),\ t = 2^{2sw}$
- Steps[6] of $u = MonPro(a,b)$, $n$ is odd and $a, b, n < 2^{sw}$, no time-consuming modular operation.

  - $u = 0$
  - for $i = 0$ to $s - 1$
    - $u = u + a_i b$
    - $u = u + (-n_0^{-1}) \cdot u_0 \cdot n$
    - $u = u/2^w$
  - if $u \geq n$ then return $u - n$
  - else return $u$

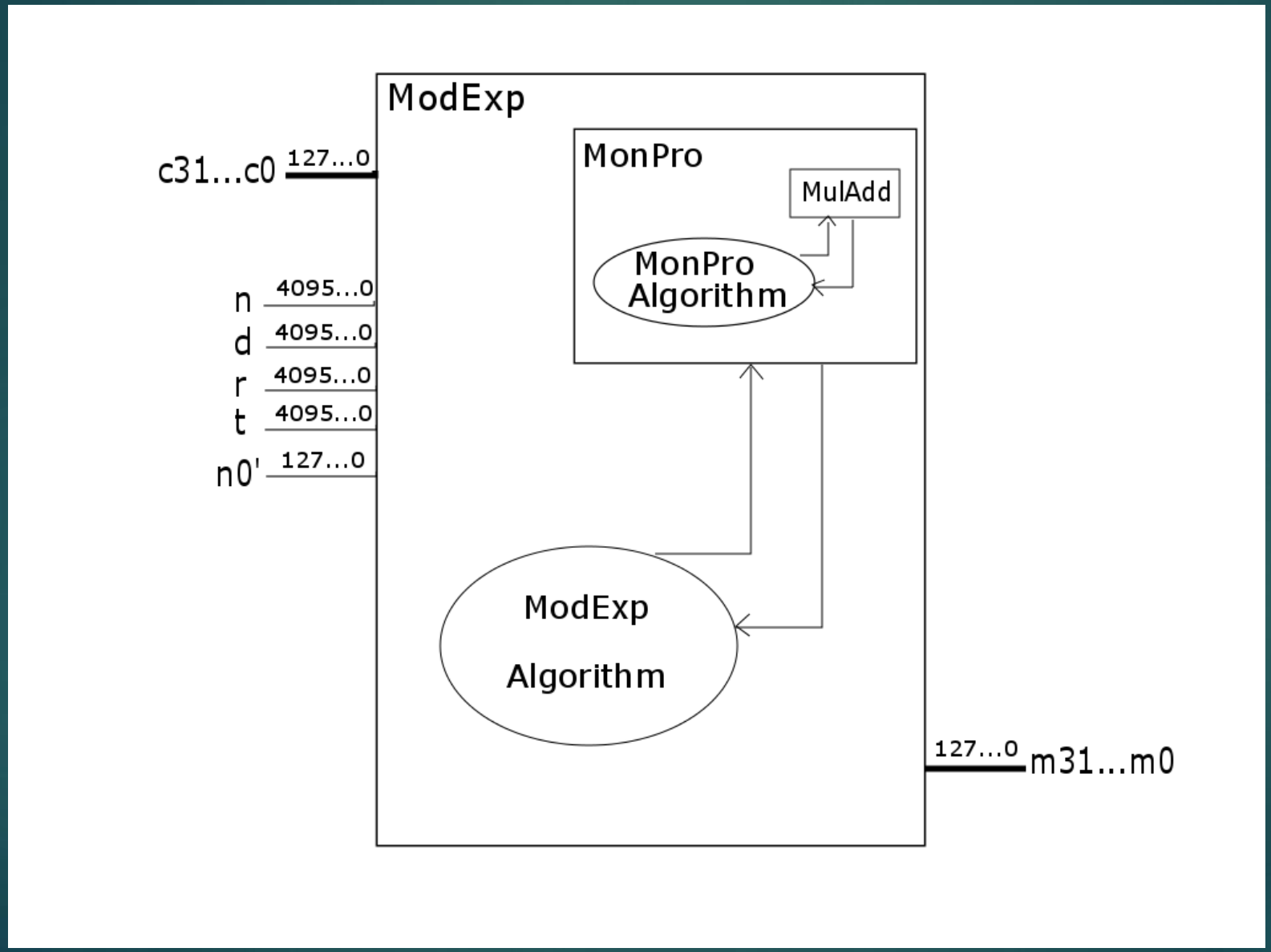
# Details of the Implementation



- Three modules: MulAdd, MonPro, ModExp
  - MulAdd handles basic multiplication and addition, used by MonPro.
  - MonPro does a Montgomery Product, used by ModExp.
  - ModExp implementes Modular Exponentiation.
- In ModExp, there is a 128-bit input wire for $c$ input and 128-bit output reg for $m$ output.
- All internal states are viewable through output pins
- Big numbers $d, n, n'_0, r, t$ are initialized before ModExp computation, through readmemh.

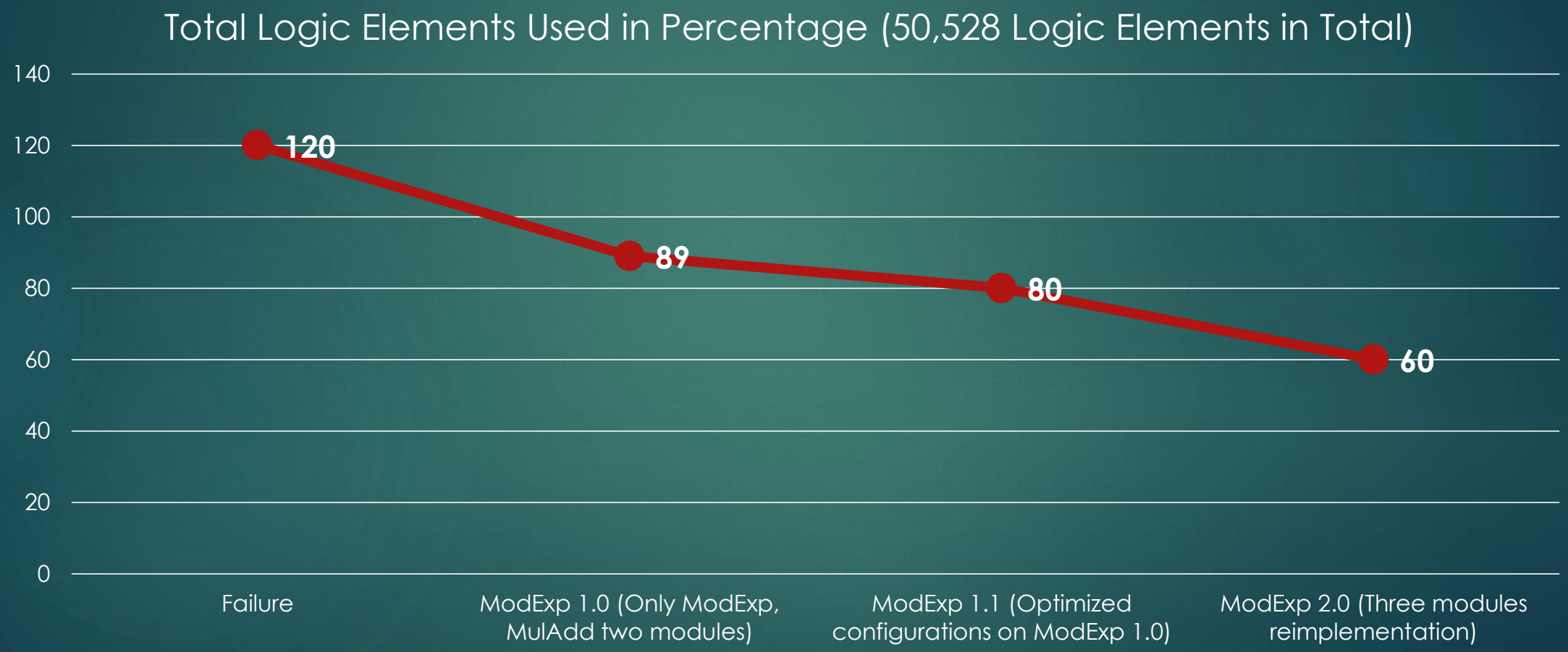
# Details of the Implementation





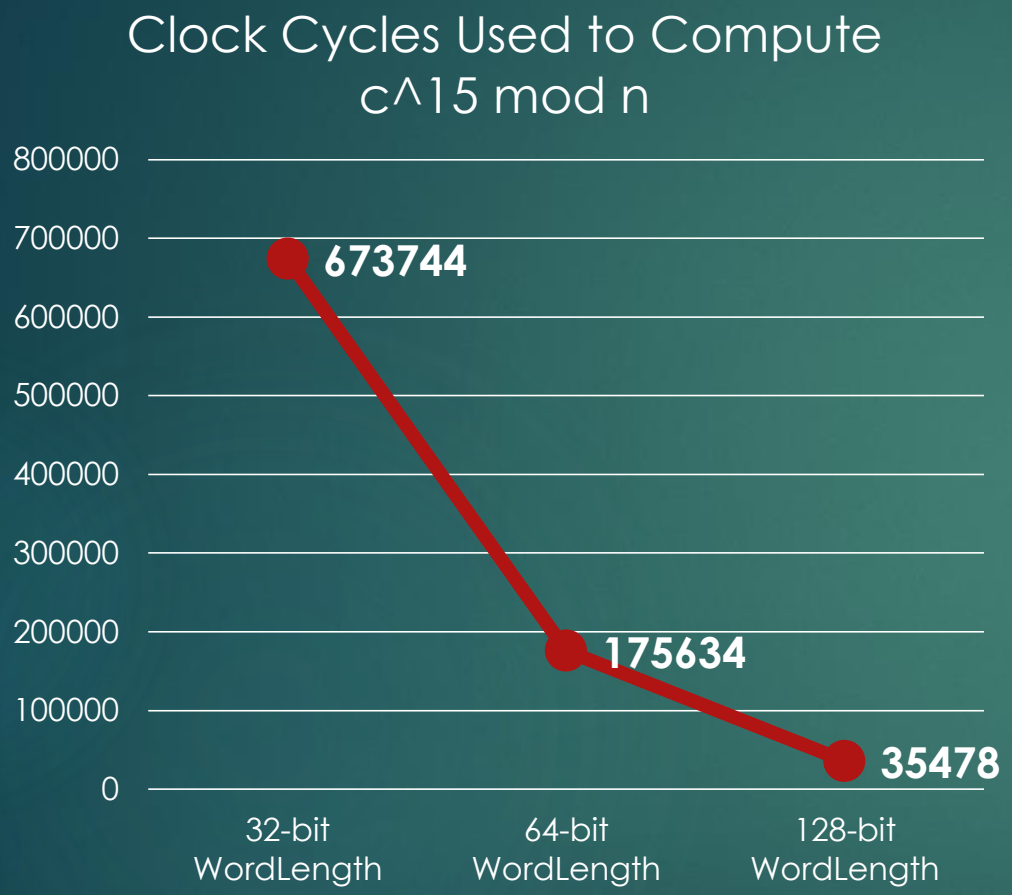
# Efforts of Fitting Code into Cyclone II EP2C50

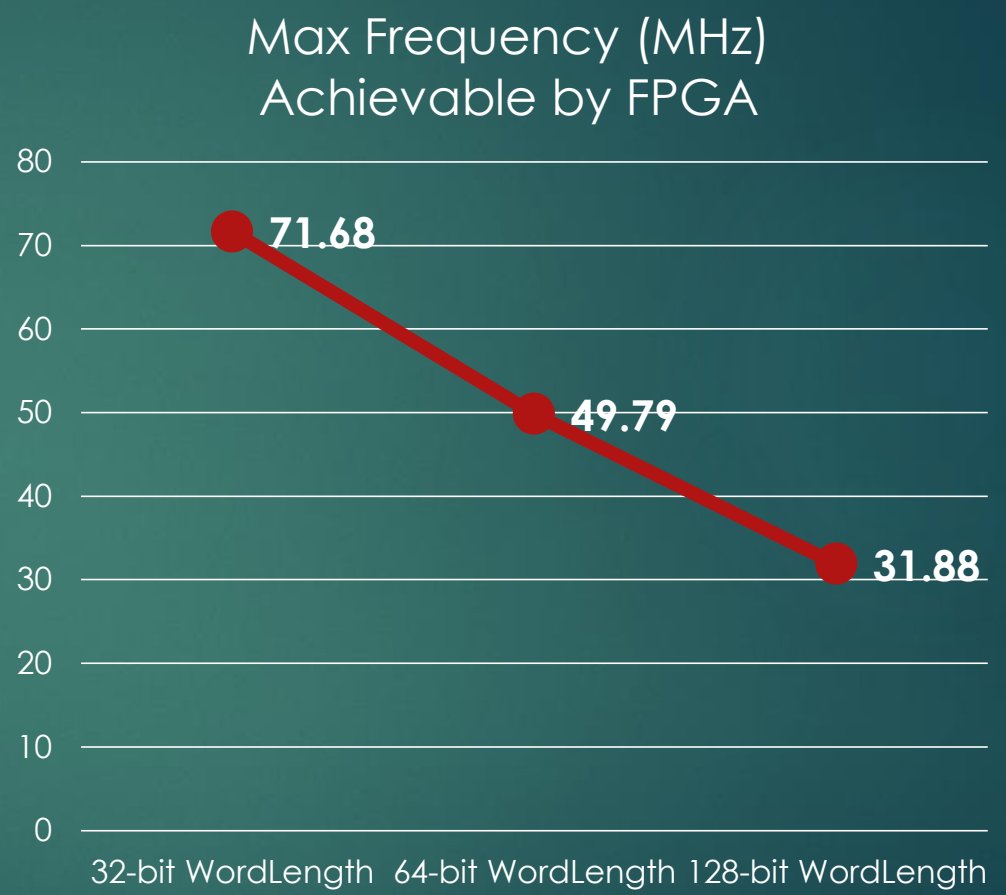




# Time Usage of Implementation





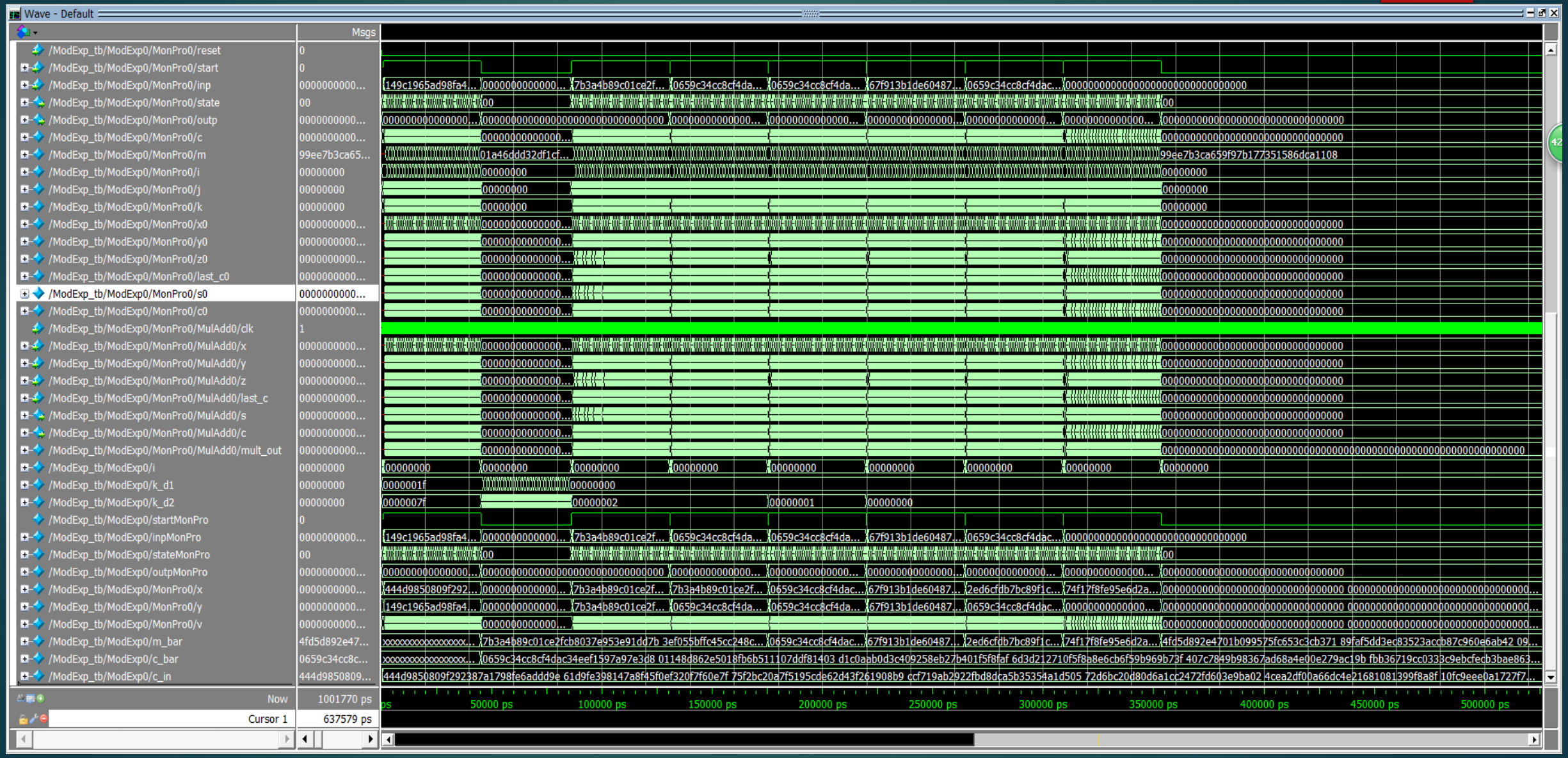




- $ClockCycles \propto 1/WordLength^2$, $TimeUsage = MaxFrequency \cdot ClockCycles$

↓ if word length increases

# RSA Simulation Demo

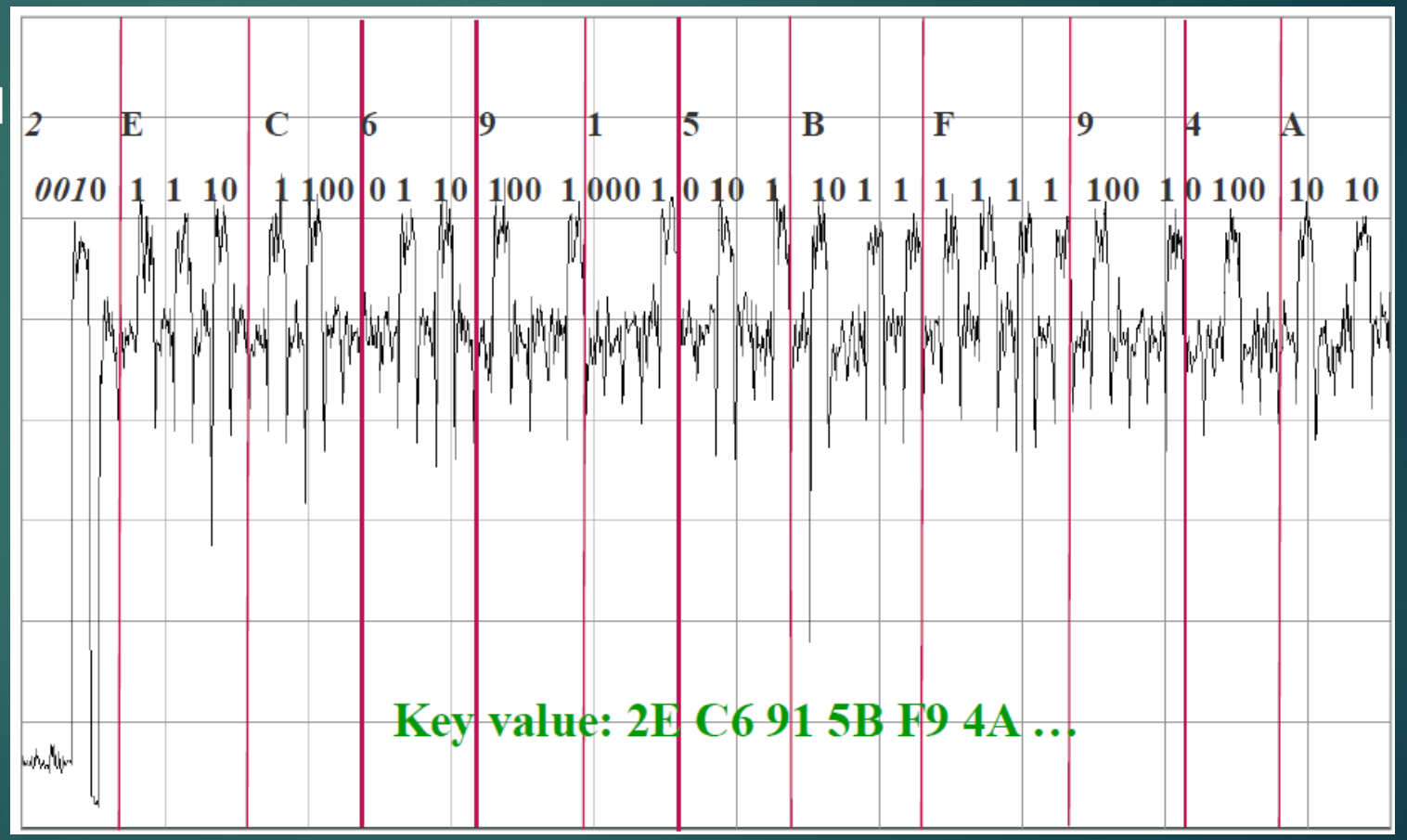




# Side Channel Attacks[8]: Simple Power Analysis (SPA)



- Attacks based on information gained from physical channels.
- SPA attack on RSA
- Doesn't work if signals contain too much noise!



# Side Channel Attacks[8]: Differential Power Analysis (DPA)



- Use signal processing and error correction to process records of RSA computation with different variables and exploit the secret key $d$.
- Many variations of DPA against RSA.
- My attack using DPA
  - Record power graph of RSA with different $c$ but same other settings; synthesize them to an average graph.
  - Record power graph of RSA with different $c$ and $d' = 0$, synthesize.
  - Record power graph of RSA with different $c$ and $d' = 1$, synthesize.
  - Compare three graphs and extract every bit of secret key $d$.

# Side Channel Attacks[10]: Timing Attack



- Suppose we can measure time precisely; we can try different $c$ many times. Goal is to crack entire $d$.
- Suppose we have guessed bits $\mathrm{d_{k-1}d_{k-2}} \ldots \mathrm{d_{i+1}}$ correctly, and we want to make a guess on $\mathrm{d_i}$.
- Construct two sets of $c$, $c$ in first set will trigger subtraction in MonPro if $d_i = 1$, the other set will not trigger subtraction in MonPro if $d_i = 1$.
- If $d_i = 1$, we will notice the time difference between two sets, otherwise if $d_i = 0$, two sets have same time usage distribution!

# Countermeasures: Montgomery Powering Ladder



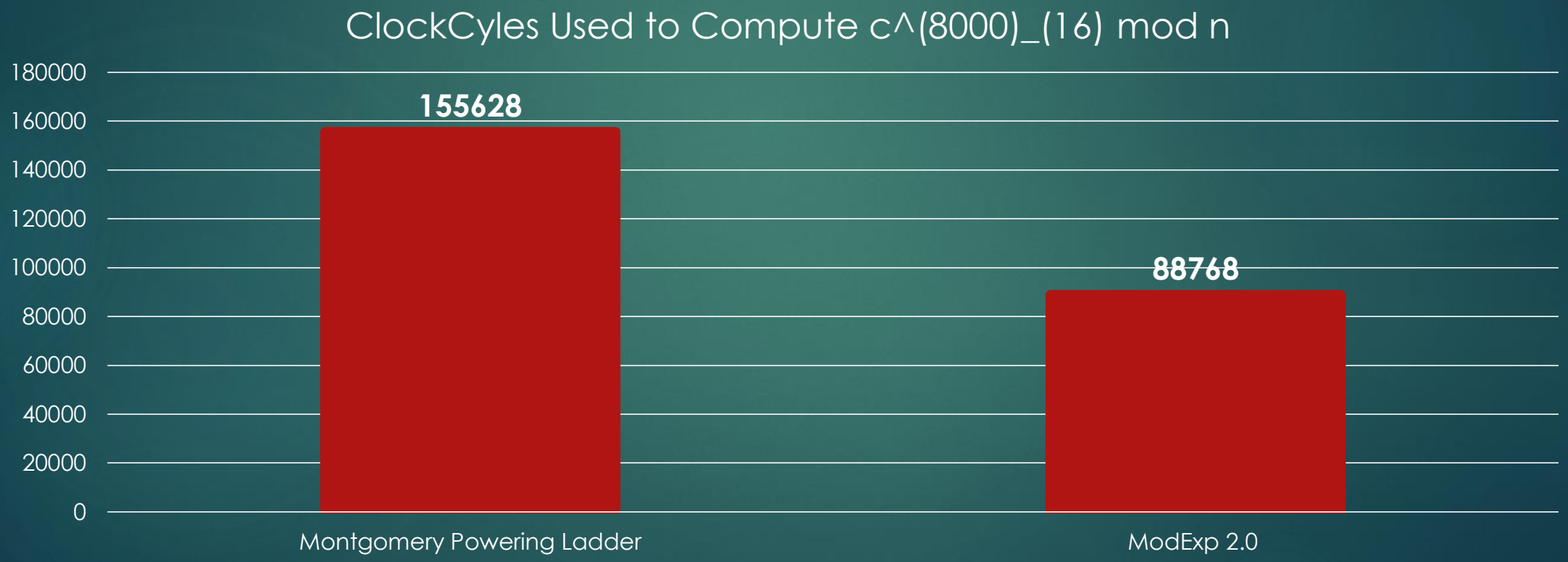
- For each bit $d_i$, ModExp behaves the same. Dummy operation is introduced
- Input: $c$, $d = (d_{k-1}, \ldots, d_0)_2$, $n$. Output: $m = c^d \bmod n$

- $R_0 \leftarrow 1$; $R_1 \leftarrow c$
- for $i = k - 1$ downto 0 do
  - $b \leftarrow 1 - d_i$
  - $R_b \leftarrow R_0 R_1$
  - $R_{d_i} \leftarrow R_{d_i}^2 \pmod n$
- return $R_0$

# Montgomery Powering Ladder: Drawback



- The time usage may double since dummy operation added, in the worst case scenario ($d = (100 \ldots 00)_2$).



# Countermeasures: Blinding

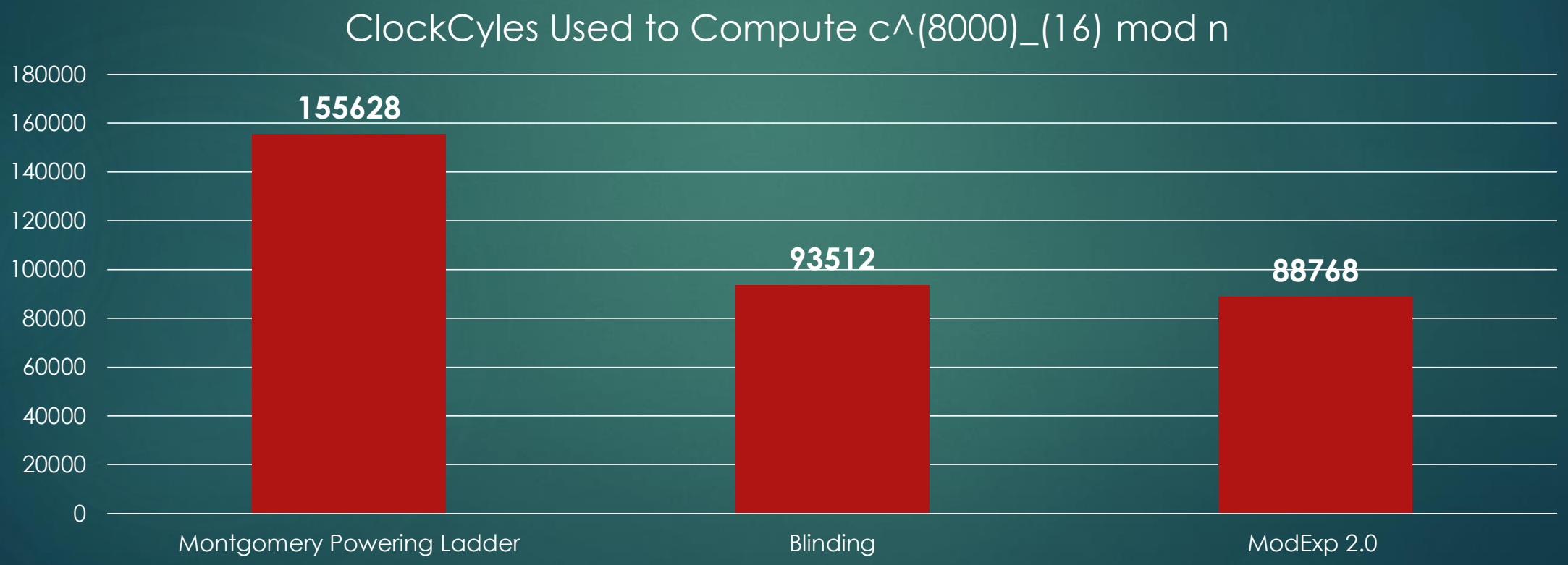


- Idea: do ModExp on blinded value then convert result to decrypted text. The signals in side channel has no meaningful information.
- $c' = c \cdot R^e \ (mod\ n)$, $R$ is a random number $< n$ and $\gcd(R, n) = 1$.
- $m' = c'^d \ (mod\ n)$
- $m = m' \cdot R^{-1} (mod\ n)$
- Drawback
  - Need a random number generator.
  - Need to compute an extra ModExp, extra MonPro, and inverse of R.

# Countermeasures: Blinding Implementation



- In my implementation, $R^e \ mod \ n$ and $R^{-1}$ are given. Only two extra MonPro operations are needed



# Future Work



- Investigate more side channel attacks.
- Create more side channel countermeasures; seek publication opportunity.

# Thank you!




- Special thanks to Prof. Cetin Koc and Prof. Tim Sherwood.

- Thanks to anyone who supported me during my study in UCSB!

# References




1. http://en.wikipedia.org/wiki/RSA_(cryptosystem)
2. http://courses.cs.vt.edu/~cs5204/fall00/protection/publicKey.html
3. http://www.techworld.com/news/security/rsa-1024-bit-private-key-encryption-cracked-3214360/
4. http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/key-size.htm
5. ModExp4096.pdf
6. Encyclopedia of Cryptography and Security, Volume 1. page 802. https://books.google.com/books?id=UuNKmgv70lMC&pg=PA802&lpg=PA802&dq=monpro+algorithm&source=bl&ots=X7yjmGHko3&sig=s3yVdrKh0zjWMFrfHhf4GyhVGDA&hl=en&sa=X&ei=dlH_VO_nEYyyogTnooG4Bw&ved=0CCAQ6AEwAA#v=onepage&q=monpro%20algorithm&f=false

# References




7. ICDv2.2.doc
8. Electronics Laboratories Advanced Engineering Course on Cryptographic Engineering. Lausanne, Switzerland Sep 8-12, 2008. Marc Joye, Thomson R&D, Cesson-Sevigne, France.
9. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. Paul C. Kocher.
10. Advanced Engineering Course on Cryptographic Engineering. Lausanne, Switzerland Jun 25-29, 2012. Marc Joye, Technicolor, France.