The Full Cost of Cryptanalytic Attacks

Michael J. Wiener

Presented by Ilya Efanov
02.06.2013

Content

Introduction

- Question:

Some algorithms with input n may require $\Theta(n)$ steps and $\Theta(n)$ memory elements.

What is the algorithms' overall cost ?

Introduction

- Question:

Some algorithms with input n may require $\Theta(n)$ steps and $\Theta(n)$ memory elements.

What is the algorithms' overall cost ?

$\Theta(n)$, $\Theta(n^2)$, or something else ?

Introduction

- How do we measure the cost of an algorithm ?

Introduction

- How do we measure the cost of an algorithm ?
- Current practice in stating the cost of an algorithm is very processor centric - total number of operations performed by all processors.

# processors

✖

# steps
or
run-time
per processor

Introduction

W/O additional memory (exhaustive search)
2^k x 1 processor
2^{k-1} x 2 processors
2^{k-2} x 4 processors
2^{k-3} x 8 processors
…

Introduction

W/O additional memory (exhaustive search)
2^k x 1 processor
2^{k-1} x 2 processors
2^{k-2} x 4 processors
2^{k-3} x 8 processors
…

With additional large memory
2^k x 1 processor / 2^m memory
2^{k-1} x 2 processors / 2^m memory
2^{k-2} x 4 processors / 2^m memory
2^{k-3} x 8 processors / 2^m memory
…

Introduction

- Is there any problem with this metric ?

Introduction

- Is there any problem with this metric ?
- It ignores all hardware components except processors

Introduction

- We want to find more useful metric that considers all hardware costs.

Full Cost

- Definition:

The full cost of an algorithm run on a collection of hardware is the number of components multiplied by the duration of their use.

# h/w components
or
total h/w cost

$\times$

run-time

Used by
– [Amirazizi, Hellman, 1981, 1988]
– [Bernstein, 2001]
– [Lenstra, Shamir, Tomlinson, Tromer, 2002] called it "throughput cost"

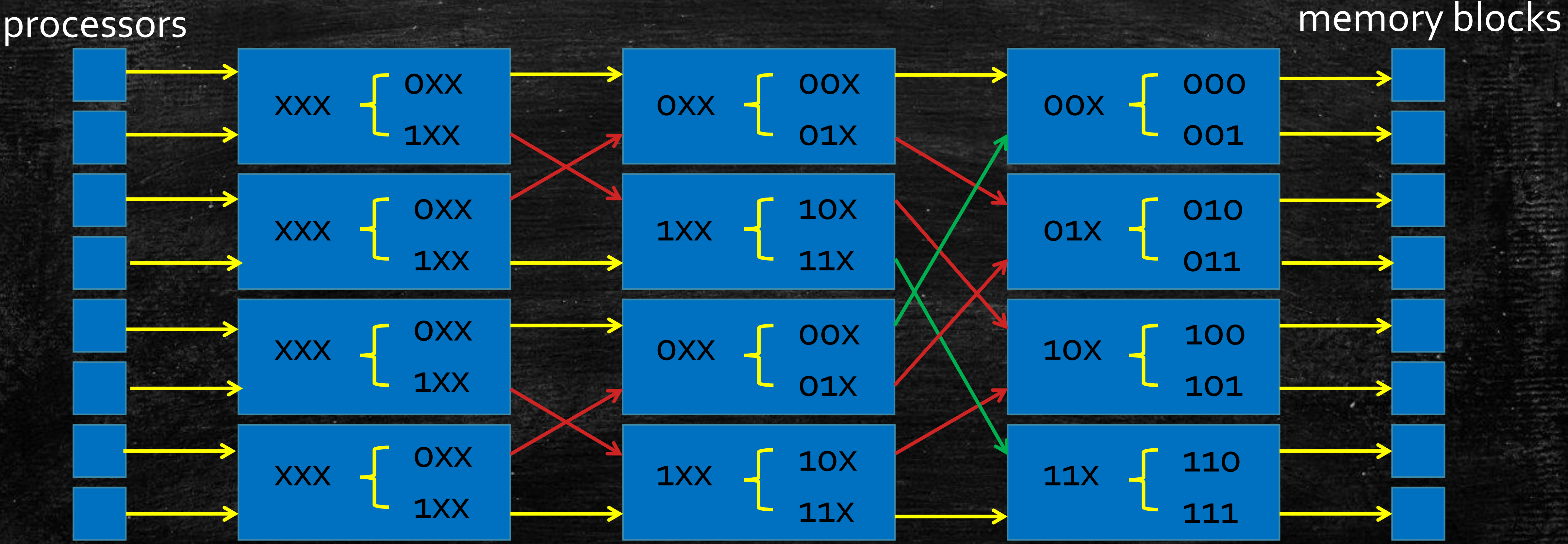
Full Cost of Connecting Many Processors to a Large Memory

- We are concerned with the case where the processors repeatedly access random locations in the large memory in parallel rather than the case where each processor is operating within its own small selection of memory
- **For instance:**

Given 1000 small processors accessing a large memory broken into 1000 blocks, where the processors generate a memory access every 100 ns, then the memory must support 1000 memory accesses every 100 ns.

Butterfly Network

- Connecting N Processors to N Blocks of Memory



Full Cost of Connecting Many Processors to a Large Memory

- We have:
  - $n = 2^k$ processors and memory blocks
  - $\Theta(nlogn)$ switching elements
  - $2^{k-1}\left(2^{k-1} - 1\right) = \Theta(n^2)$ total wire length

Full Cost of Connecting Many Processors to a Large Memory

- We have:
  - $n = 2^k$ processors and memory blocks
  - $\Theta(nlogn)$ switching elements
  - $2^{k-1}\left(2^{k-1} - 1\right) = \Theta(n^2)$ total wire length
- It's possible to pack the memory elements in 3D, so it reduces total wiring cost to $\Theta(n^{3/2})$

Full Cost of Connecting Many Processors to a Large Memory

- We have:
  - $n = 2^k$ processors and memory blocks
  - $\Theta(nlogn)$ switching elements
  - $2^{k-1}\left(2^{k-1} - 1\right) = \Theta(n^2)$ total wire length
- It's possible to pack the memory elements in 3D, so it reduces total wiring cost to $\Theta(n^{3/2})$
- We cannot pack wires any tighter that filling three dimensions

Full Cost of Connecting Many Processors to a Large Memory

- We have:
  - $n = 2^k$ processors and memory blocks
  - $\Theta(nlogn)$ switching elements
  - $2^{k-1}(2^{k-1} - 1) = \Theta(n^2)$ total wire length
- It's possible to pack the memory elements in 3D, so it reduces total wiring cost to $\Theta(n^{3/2})$
- We cannot pack wires any tighter that filling three dimensions
- Is it possible to reduce wiring cost below $\Theta(n^{3/2})$?

Full Cost of Connecting Many Processors to a Large Memory

- We have:
  - $n = 2^k$ processors and memory blocks
  - $\Theta(nlogn)$ switching elements
  - $2^{k-1}(2^{k-1} - 1) = \Theta(n^2)$ total wire length
- It's possible to pack the memory elements in 3D, so it reduces total wiring cost to $\Theta(n^{3/2})$
- We cannot pack wires any tighter that filling three dimensions
- Is it possible to reduce wiring cost below $\Theta(n^{3/2})$?
- No.

General Case

- Problem 1: Memory request rate may be not $\Theta(1)$ bits per unit time.
- For example, if each processor generates a request of size $logn$ every n units of time, we say that the memory access rate per processor is $logn/n$.
- Problem 2: The number of processors and number of memory blocks may not be equal.

General Case

- Theorem 1:

The total number of components required to allow each of p processors uniformly random access to m memory elements at a memory access rate r, including the components of processors, memory, switching elements and wires is $\Theta(p + m + (pr)^{3/2})$.

General Case

- Corollary 1:

For an algorithm where p processors access a memory of size m at rate r and the total number of processor operations T, the full cost of the algorithm is $F = \Theta((T/p)(p + m + (pr)^{3/2}))$.

General Case

- Corollary 1:

For an algorithm where p processors access a memory of size m at rate r and the total number of processor operations T, the full cost of the algorithm is $F = \Theta((T/p)(p + m + (pr)^{3/2}))$.

- Corollary 2:

For an algorithm where the rate r at which p processors access a memory of size m is high, $1/r = m^{o(1)}$, the memory size is independent of the number of processors and the number of processor operations is T, the full cost of the algorithm is a minimum of $F = \Theta(Trm^{1/3})$ where $p = \Theta(m^{2/3}/r)$

Full Cost vs. Processor Centric Cost

- If we rewrite the full cost from Corollary 1 as
$F = \Theta\left((T/p)\left(p + m + (pr)^{3/2}\right)\right) = \Theta\left(T\left(1 + m/p + p^{1/2}r^{3/2}\right)\right)$

So $F = \Omega(T)$.

And we are getting $F = \Theta(T)$ iff $p = \Omega(m)$ and $r = O(p^{-1/3})$

Full Cost vs. Processor Centric Cost

- If we rewrite the full cost from Corollary 1 as

$$F = \Theta\left((T/p)\left(p + m + (pr)^{3/2}\right)\right) = \Theta\left(T\left(1 + m/p + p^{1/2}r^{3/2}\right)\right)$$

So $F = \Omega(T)$.

And we are getting $F = \Theta(T)$ iff $p = \Omega(m)$ and $r = O(p^{-1/3})$

Conclusion: The full cost of an algorithm is never less than the traditional count of processor steps.

Applications

- Double encryption

More:

- Discrete logarithm
- Factoring
- Encryption
- Triple encryption
- Hash Collisions

Double encryption

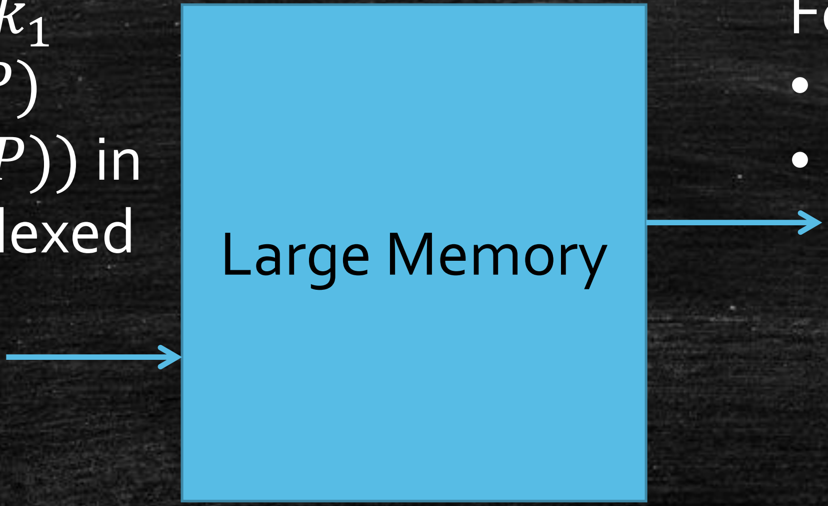
- $C = E_{k_2}(E_{k_1}(P))$
- Key space is n^2
  - Using techniques for recovering many keys at once with a chosen-plaintext attack reduces key space to n
- Given (P, C) find (k_1, k_2)
- Attack approaches
  - Simple meet-in-the-middle
  - Parallel collision search

Meet-In-The-Middle

- Obrevation $E_{k_1}(P) = E_{k_2}^{-1}(C)$

For each possible k_1

- Compute $E_{k_1}(P)$
- Store $(k_1, E_{k_1}(P))$ in a hash table indexed by $E_{k_1}(P)$



For each possible k_2

- Compute $E_{k_2}^{-1}(C)$
- Look it up in the table.

Whenever, $E_{k_2}^{-1}(C)$ is in the table, we have a candidate key pair (k_1, k_2) that can be tested on the other (P, C) pairs

Meet-In-The-Middle

- $m = \Theta(nlogn)$ – required memory size
- $t = \Omega(logn)\ or\ t = n^{o(1)}$ - encryption and decryption time
  - keys and text of size $\Theta(logn)$
- $r = \Theta(logn/t)$ – memory access rate
- $T = \Theta(nt)$

By Corollary 2 the full cost of a meet-in-the-middle attack on double encryption is $F = \Theta(Trm^{1/3})$ and then $F = \Theta(nlogn)^{4/3}$

Note: We will say that it's $n^{4/3+o(1)}$

Parallel Collision Search applied to MITM attacks

- $m = \Theta(wlogn)$ - required memory size
  - w memory locations, each of size $\Theta(logn)$
- $T = \Theta(n^{3/2}t/w^{1/2})$ - the total number of processor steps across p processors
- $r = \Theta(w^{1/2}(logn)/(n^{1/2}t))$ – memory access rate
  - memory access of size $\Theta(logn)$ is made every $(n/w)^{1/2}$ encryptions

Parallel Collision Search applied to MITM attacks

By Corollary 1, the full cost of the algorithm is

$F = \Theta((n^{3/2}t/w^{1/2}p)(p + m + (pr)^{3/2}))$.

- This cost is a minimum of $\Theta(n^{6/5}t^{2/5}(logn)^{4/5})$ when:
  - $p = \Theta(wlogn)$
  - $w = \Theta(n^{3/5}t^{6/5}/(logn)^{8/5})$
- Note: $n^{6/5+o(1)}$

Result Comparison

- The full cost of a known-plaintext attack on double encryption using a simple meet-in-the-middle attack is $n^{4/3+o(1)}$ and this can be reduced to $n^{6/5+o(1)}$ using parallel collision search.

Conclusion

In general:

- p - # processors
- m – memory size
- r – memory access rate
- T – processor steps

Then full cost is $F = \Theta(\underbrace{T}_{\text{processor costs}} + \underbrace{Tm/p}_{\text{memory}} + \underbrace{Tp^{1/2}r^{3/2}}_{\text{cost of connecting the memory}})$

Conclusion

- For the full cost to match the traditional measure of algorithm cost, i.e. $F = \Theta(T)$, we must have $p = \Omega(m)$ and the memory access rate must be low, $r = O(p^{-1/3})$.

Cryptanalytic problem	Attack Method	Processor Steps	Full Cost
One discrete logarithm (prime group order n)	Shanks Parallel Collision Search	$n^{1/2+o(1)}$ $n^{1/2+o(1)}$	$n^{2/3+o(1)}$ $n^{1/2+o(1)}$
n-th discrete logarithm (prime group order n)	Shanks Parallel Collision Search	$n^{o(1)}$ $n^{o(1)}$	$n^{1/3+o(1)}$ $n^{1/5+o(1)}$
Factoring	Number field sieve	$L(1.90188\ldots)$	$L(1.94961\ldots)$
Block cipher encrypting (|key space| = n)	All methods (first key) Table lookup (n-th key)	$n^{1+o(1)}$ $n^{o(1)}$	$n^{1+o(1)}$ $n^{1/3+o(1)}$
Double encryption	Meet-in-the-middle Parallel collision search	$n^{1+o(1)}$ $n^{1+o(1)}$	$n^{4/3+o(1)}$ $n^{6/5+o(1)}$
2-key triple encryption (|message space| = $u \geq n$)	Unlimited chosen texts w known texts ($w \leq u$)	$n^{1+o(1)}$ $n^{1+o(1)}u/w$	$n^{4/3+o(1)}$ $n^{1+o(1)}u/w^{\frac{2}{3}}$
3-key triple encryption	Meet-in-the-middle Parallel collision search	$n^{2+o(1)}$ $n^{2+o(1)}$	$n^{7/3+o(1)}$ $n^{11/5+o(1)}$
Hash collision (|output space| = n)	Meet-in-the-middle Parallel collision search	$n^{1/2+o(1)}$ $n^{1/2+o(1)}$	$n^{2/3+o(1)}$ $n^{1/2+o(1)}$

Thank you