COMPARATIVE ANALYSIS OF SOFTWARE LIBRARIES FOR PUBLIC KEY CRYPTOGRAPHY

Ashraf Abusharekh
Kris Gaj
Department of Electrical & Computer Engineering
George Mason University

OBJECTIVE

- Evaluation of Multi-precision Arithmetic Libraries for Use in Public Key Cryptography

- Practical Recommendations

  Which library is best for a particular application?

PUBLIC KEY SCHEMES OPERATIONS(1)

Signature Scheme	RSA	DSA	ECDSA
System Parameters	N/A	**Primality Testing**	**EC Generation** **Point Counting**
Key Generation	**Modular Exponentiation** Multiplication GCD xGCD	**Modular Exponentiation**	**Scalar Multiplication**
Signature Generation	**Modular Exponentiation**	Addition Multiplication **Modular Exponentiation** xGCD	**Scalar Multiplication** Addition Multiplication xGCD
Signature Verification	**Modular Exponentiation**	Multiplication **Modular Exponentiation** xGCD	**Scalar Multiplication** Multiplication xGCD Point Addition3

PUBLIC KEY SCHEMES OPERATIONS(2)

- Large Integers (768 – 2048)
  - Addition, Multiplication, Modular Exponentiation, GCD, xGCD, Primality Testing
- Elliptic Curve Points (140 – 224)
  - Elliptic Curve Point Addition and Scalar Multiplication

Problem: Operations are difficult to implement.

Solution: Use existing libraries.

LIBRARIES

- Many arithmetic and number theoretic libraries, commercial and in the public domain have been developed to perform these multi-precision arithmetic operations efficiently
  - Beecrypt, BIGNUM, Botan, bnlib, CLN, cryptolib, CryptoPP, freelip, GMP, Libgcrypt, LiDIA, linteger, MIRACL, nettle, NTL, OpenSSL, PARI, PIOLOGIE, zen ……..

- **Problem**: Which one to use?

Evaluated Libraries

Library	Category	License	Version used
CLN	Number theoretic	GNU GPL	1.1.5
CryptoPP	Cryptographic	Copyrighted as a compilation	5.1
GMP	Multi-precision, Number theoretic	GNU GPL	4.1.2
LiDIA	Number theoretic	LiDIA group	2.1pre7
MIRACL	Cryptographic	Shamus Software Ltd.	4.82
NTL	Number theoretic	GNU GPL	5.3.1
OpenSSL	Cryptographic	Apache-style license	0.9.7c
PIOLOGIE	Multi-precision, Number theoretic	www.hipilib.de	1.3.2

GEORGE MASON UNIVERSITY

PLATFORMS

Processor/hardware	Operating System	Compiler
2.00 GHz Pentium IV Processor, 512 MB RAM	Windows XP Cygwin	GNU C/C++ 3.3.1
	RedHat Linux 9.0	GNU C/C++ 3.3.1
Sun: 2x 400 MHz UltraSPARC-Solaris-II, 4-MB E-cache, 2048 MB RAM	Solaris 5.8	GNU C/C++ 2.95.2

EVALUATION CRITERIA

PRIMARY

- Performance of Primitive Operations
- Support for Public Key Cryptosystems
  - Primitive Operations
  - Public Key Schemes

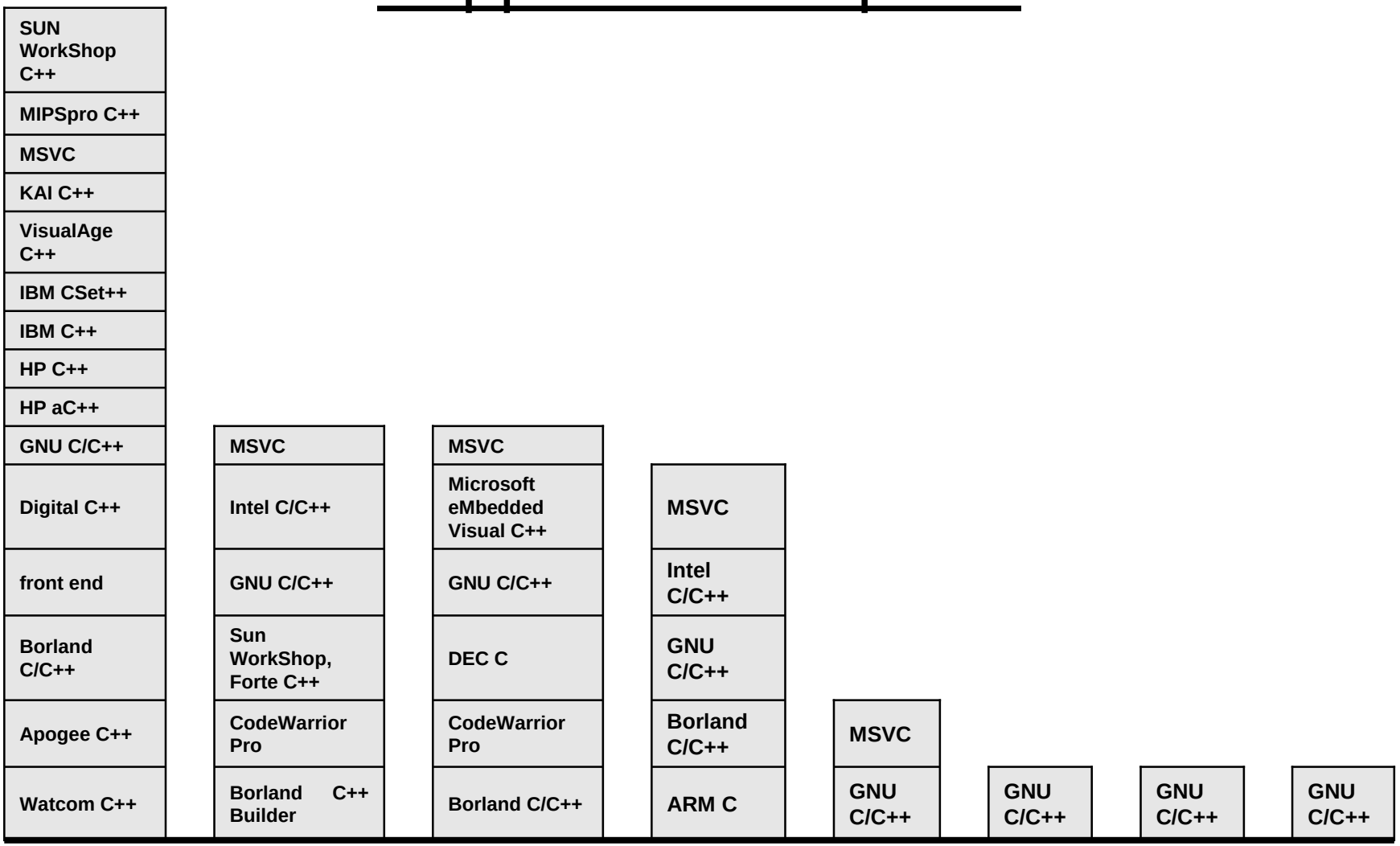
SECONDARY

- Documentation & Ease of Use
- Supported Compilers

Evaluation Criteria
Documentation & Ease of Use

Documentation \ Ease of use	Hard		Easy
Sufficient		GMP LiDIA MIRACL NTL	**Best**
		CLN	PIOLOGIE
Insufficient	CryptoPP **Worst**	OpenSSL	

Evaluation Criteria
Supported Compilers



Operations, Operand Sizes

- Large Integers (768/1024/2048)
  - Multiplication, Modular Exponentiation, GCD, xGCD, Primality Testing
- $E(F_P)$ (162/224/384)
  - Addition, Point Scalar Multiplication
- $E(F_{2^n})$ (163/233/409)
  - Addition, Scalar Multiplication

METHODOLOGY(1)

- Measuring Performance of Operations
  - RDTSC Method: Clock Cycles
    - Pentium IV: RDTSC Instruction
  - Timing Method: Milliseconds
    - UltraSPARC-II, "***`gettimeofday()`***"
  - 100 execution times for each operation

METHODOLOGY(2)

Operation	OP	Group	Comments
Multiplication	MUL	A	$I_i \times J_i$
MOD Exp E = 3	E_3	A	I_i^3 MOD K_i
MOD Exp E = 65537	E_{65537}	A	I_i^{65537} MOD K_i
MOD Exp E, size of modulus	E	A	$I_i^{J_i}$ MOD K_i
Greatest Common Divisor	GCD	B, A	GCD(P_i^j, K_i)
Extended GCD	xGCD	B, A	xGCD(P_i^j, K_i)

- Operands
  - Large Integers:
    - Random I_n, J_n, K_n, n = {768, 1024, 2048}
    - Random Primes $P_n^{(j)}$, n = {768, 1024, 2048}, j = [0, 9]
  - $E(F_{2\hat{}n})$
    - SEC 2 recommended 163, 233, 409.
    - Random Points T_n, S_n, n = {163, 233, 409}
  - $E(F_P)$
    - Random, 162, 226, 386.
    - Random Points T_n, S_n, n = {162, 226, 386}

METHODOLOGY(3)

➢Operation Ranking

P4-WinXP/MULTIPLICATION(Clock Cycles)			
Library	$result_{768}$	$result_{1024}$	$result_{2048}$
CLN	8,940	11,763	29,133
CryptoPP	38,432	37,928	78,755
GMP	3,423	5,364	17,605
LiDIA	3,573	6,047	18,722
MIRACL	10,974	18,613	71,512
NTL	3,381	5,426	17,722
OpenSSL	9,055	15,218	48,438
Piologie	29,910	41,163	113,977
Min_n	3,381	5,364	17,605

P4-WinXP/MULTIPLICATION Ranking				
Library	R_{768}	R_{1024}	R_{2048}	Rank
CLN	2.64	2.19	1.65	2.12
CryptoPP	11.37	7.07	4.47	7.11
GMP	1.01	1.00	1.00	1.00
LiDIA	1.06	1.13	1.06	1.08
MIRACL	3.25	3.47	4.06	3.58
NTL	1.00	1.01	1.01	1.01
OpenSSL	2.68	2.84	2.75	2.75
Piologie	8.85	7.67	6.47	7.60

$$\text{Operation Rank/Lib/OS} = \sqrt[3]{\prod_{n=768,1024,2048} R_n} \quad \text{where } R_n = result_n / Min_n$$

METHODOLOGY(4)

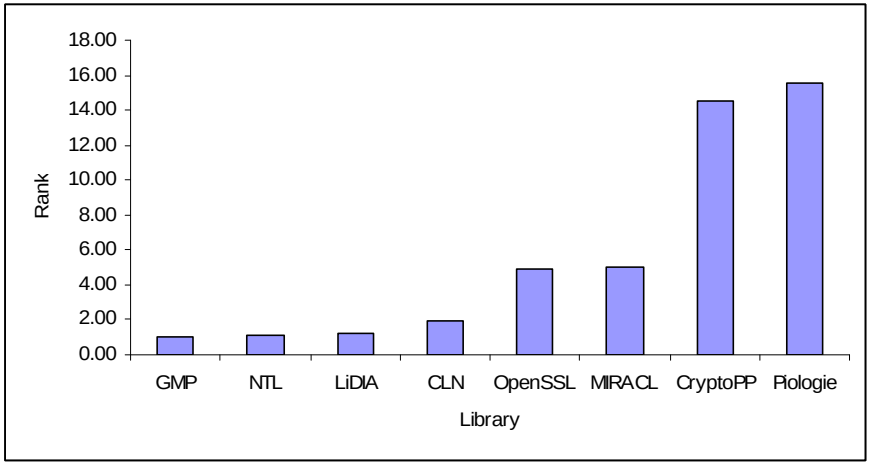
- Library Ranking

	X_1	X_2	X_3	X_4	X_5	X_6	
Library	**MUL**	**E = 3**	**E = 65537**	**Large E**	**GCD**	**xGCD**	**P4-WinXP Rank**
CLN	**2.12**	**2.23**	**2.25**	**2.79**	**1.34**	**1.37**	**1.95**
CryptoPP	**7.11**	**15.17**	**4.71**	**4.04**	**464.90**	**9.99**	**14.56**
GMP	**1.00**	**1.00**	**1.00**	**1.00**	**1.01**	**1.08**	**1.01**
LiDIA	**1.08**	**1.45**	**1.08**	**1.65**	**1.03**	**1.10**	**1.21**
MIRACL	**3.58**	**22.40**	**4.56**	**2.62**	**5.15**	**3.15**	**5.00**
NTL	**1.01**	**1.42**	**1.17**	**1.18**	**1.00**	**1.00**	**1.12**
OpenSSL	**2.75**	**8.07**	**2.65**	**2.33**	**8.31**	**12.17**	**4.90**
PIOLOGIE	**7.60**	**7.40**	**6.63**	**10.65**	**16.41**	**213.30**	**15.51**

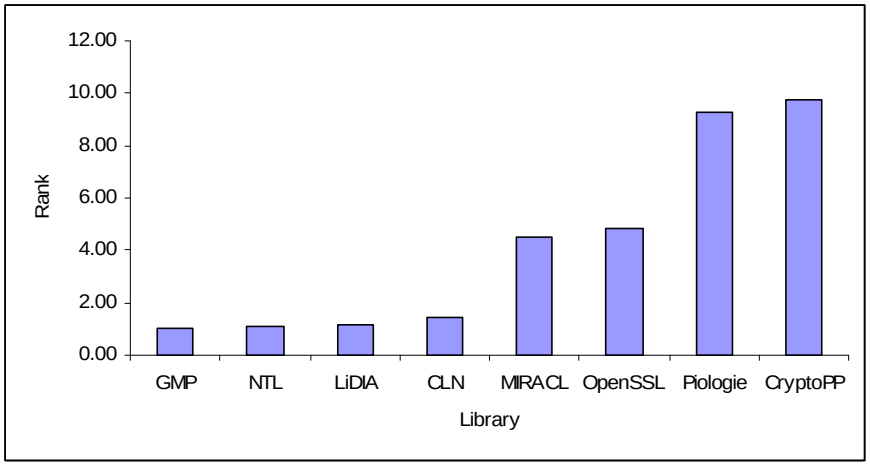
$$\text{Library Rank/OS} = \sqrt[N]{\prod_{k=1}^{N} X_k}$$

RESULTS LARGE INTEGERS

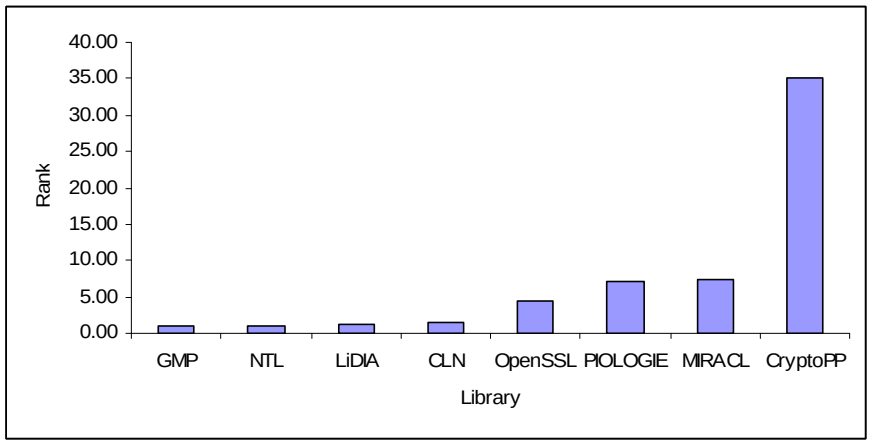
RESULTS P4-WinXP



RESULTS RedHat

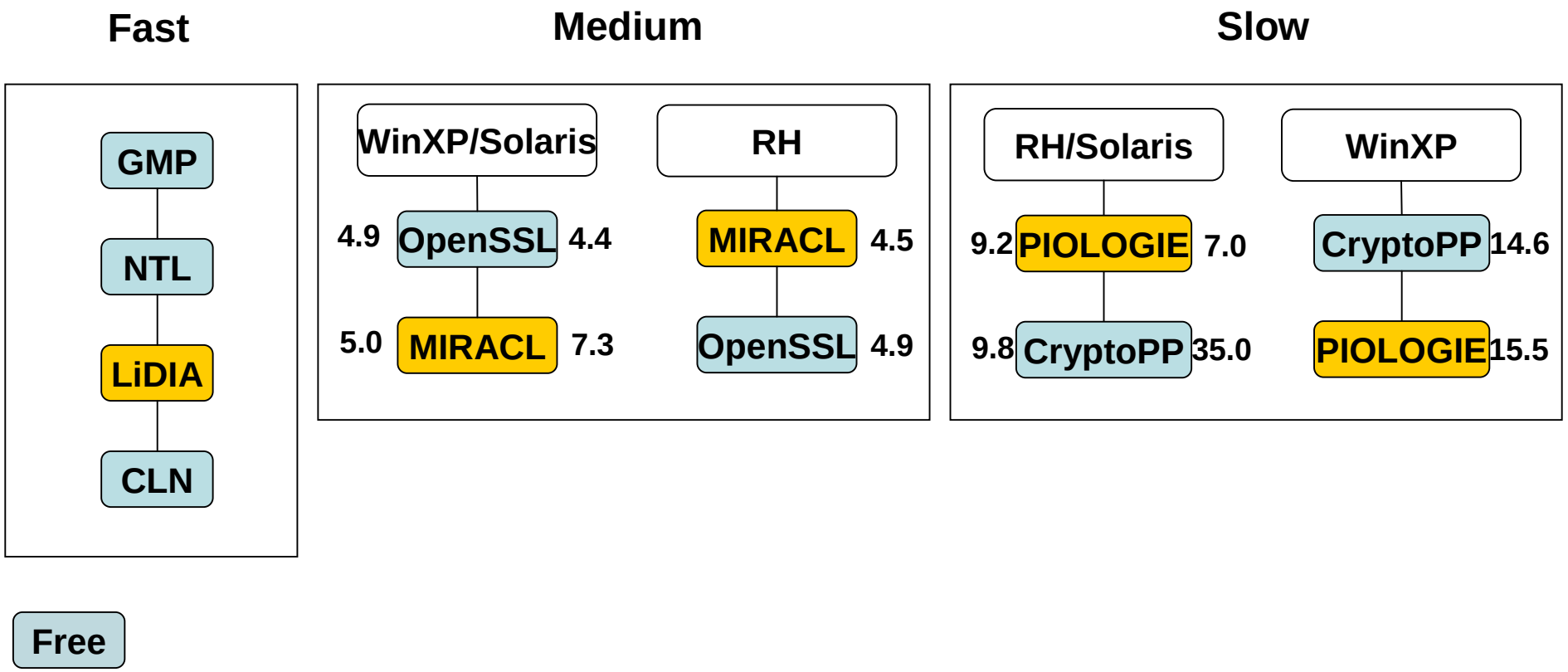


RESULTS UltraSPARC-Solaris



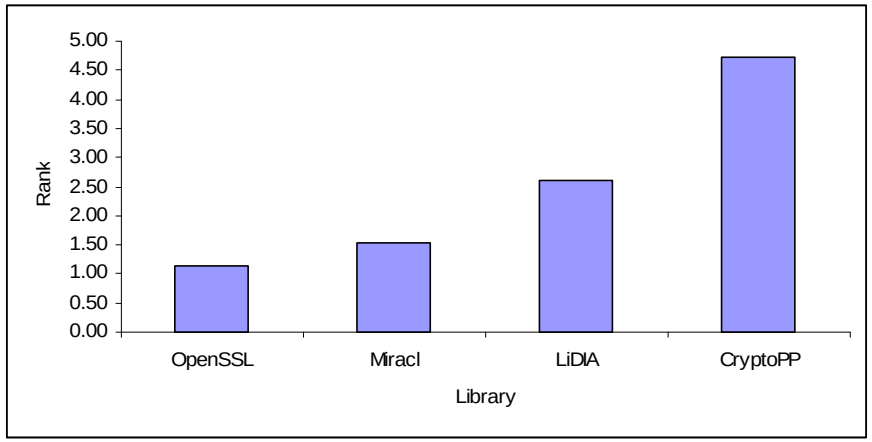
Results

Operations On Large Integers

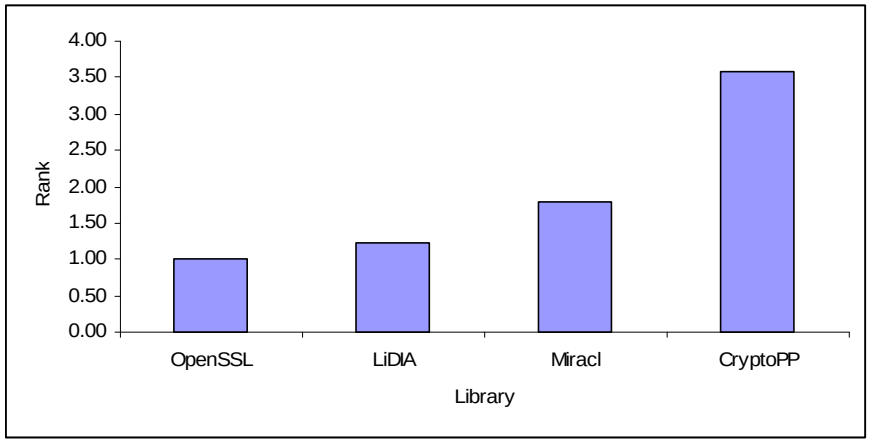


RESULTS $E(F_p)$

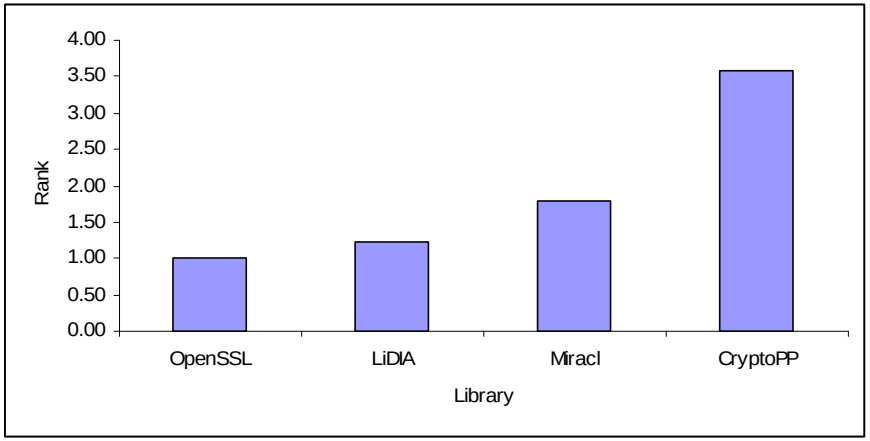
RESULTS P4-WinXP



RESULTS RedHat

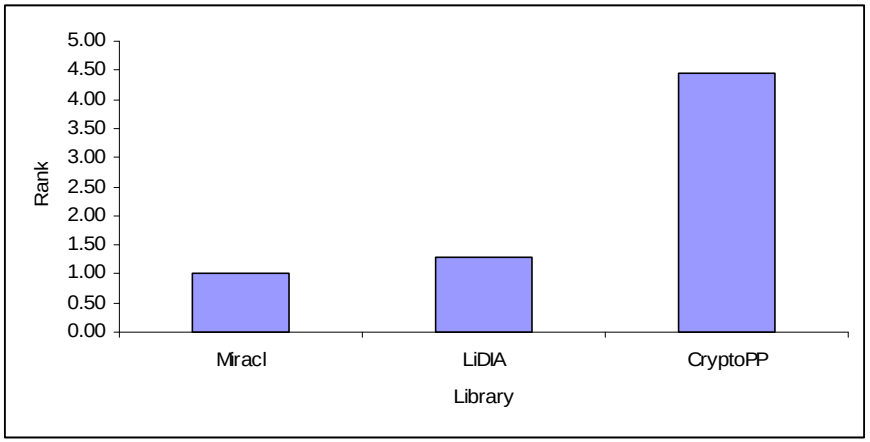


RESULTS UltraSPARC-Solaris

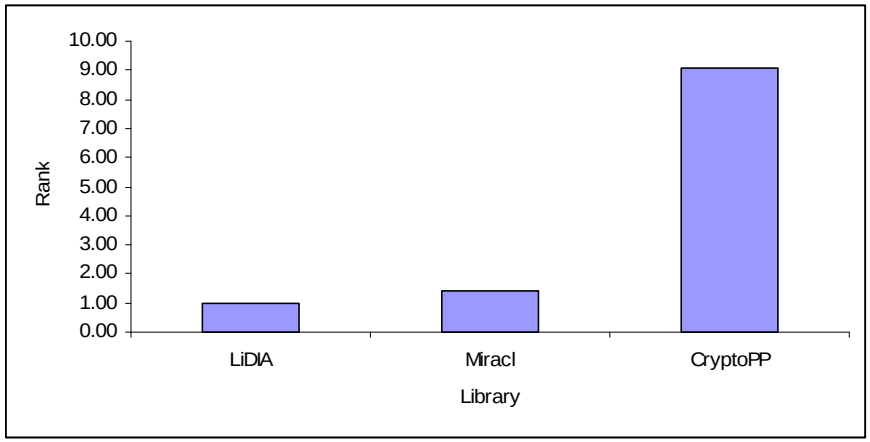


RESULTS $E(F_{2^n})$

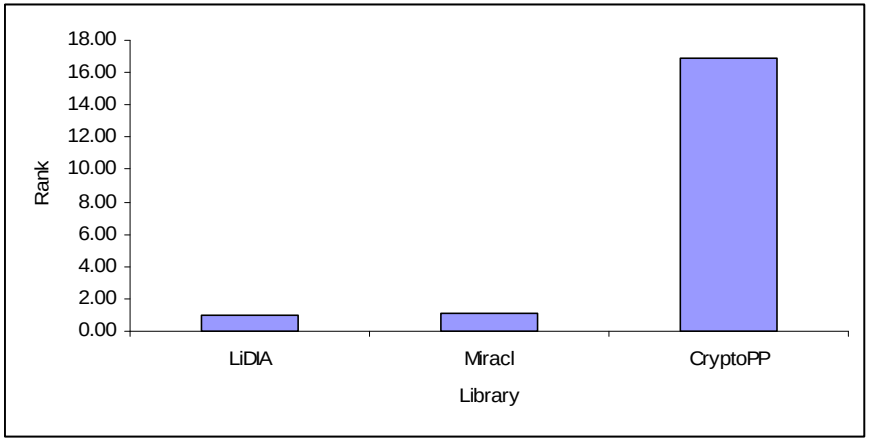
RESULTS P4-WinXP



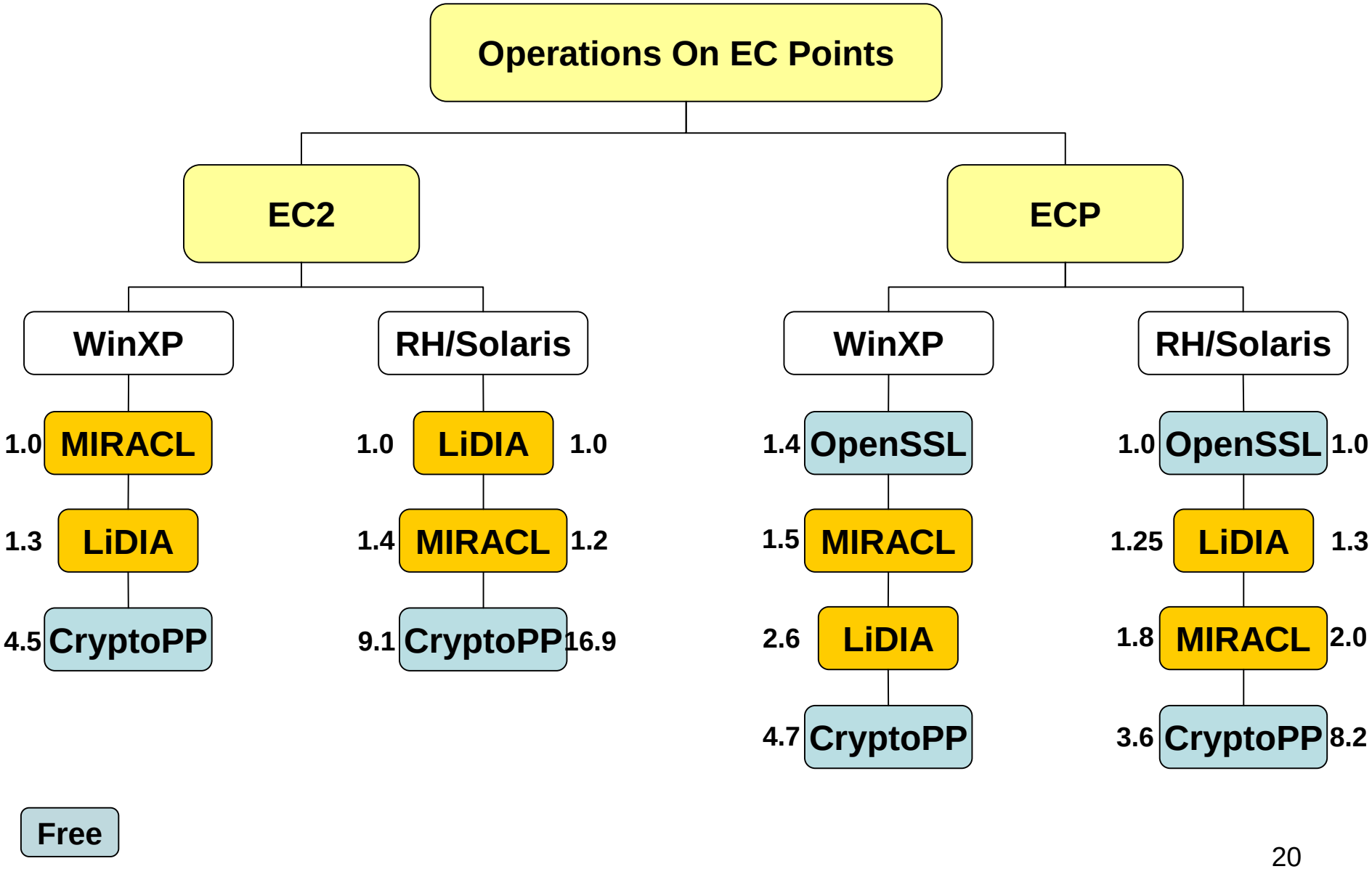
RESULTS RedHat



RESULTS UltraSPARC-Solaris



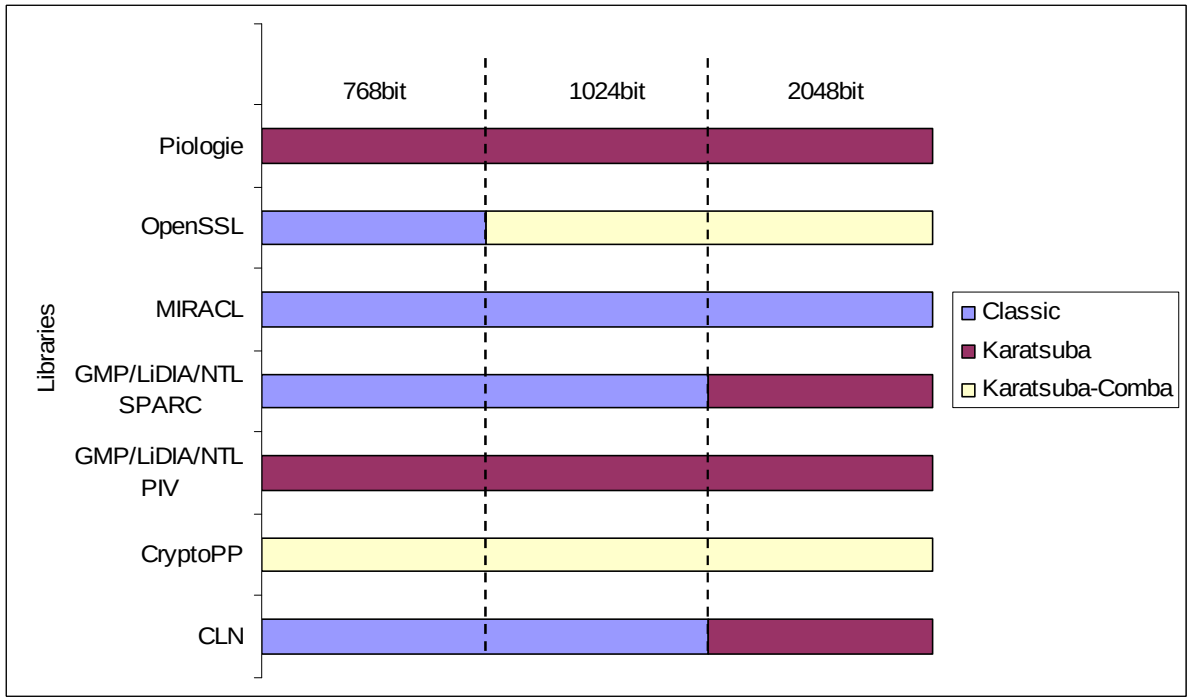
Results


Operations On EC Points
EC2
ECP
WinXP
RH/Solaris
WinXP
RH/Solaris
1.0 MIRACL
1.3 LiDIA
4.5 CryptoPP
1.0 LiDIA 1.0
1.4 MIRACL 1.2
9.1 CryptoPP 16.9
1.4 OpenSSL
1.5 MIRACL
2.6 LiDIA
4.7 CryptoPP
1.0 OpenSSL 1.0
1.25 LiDIA 1.3
1.8 MIRACL 2.0
3.6 CryptoPP 8.2
Free

FACTORS AFFECTING PERFORMANCE

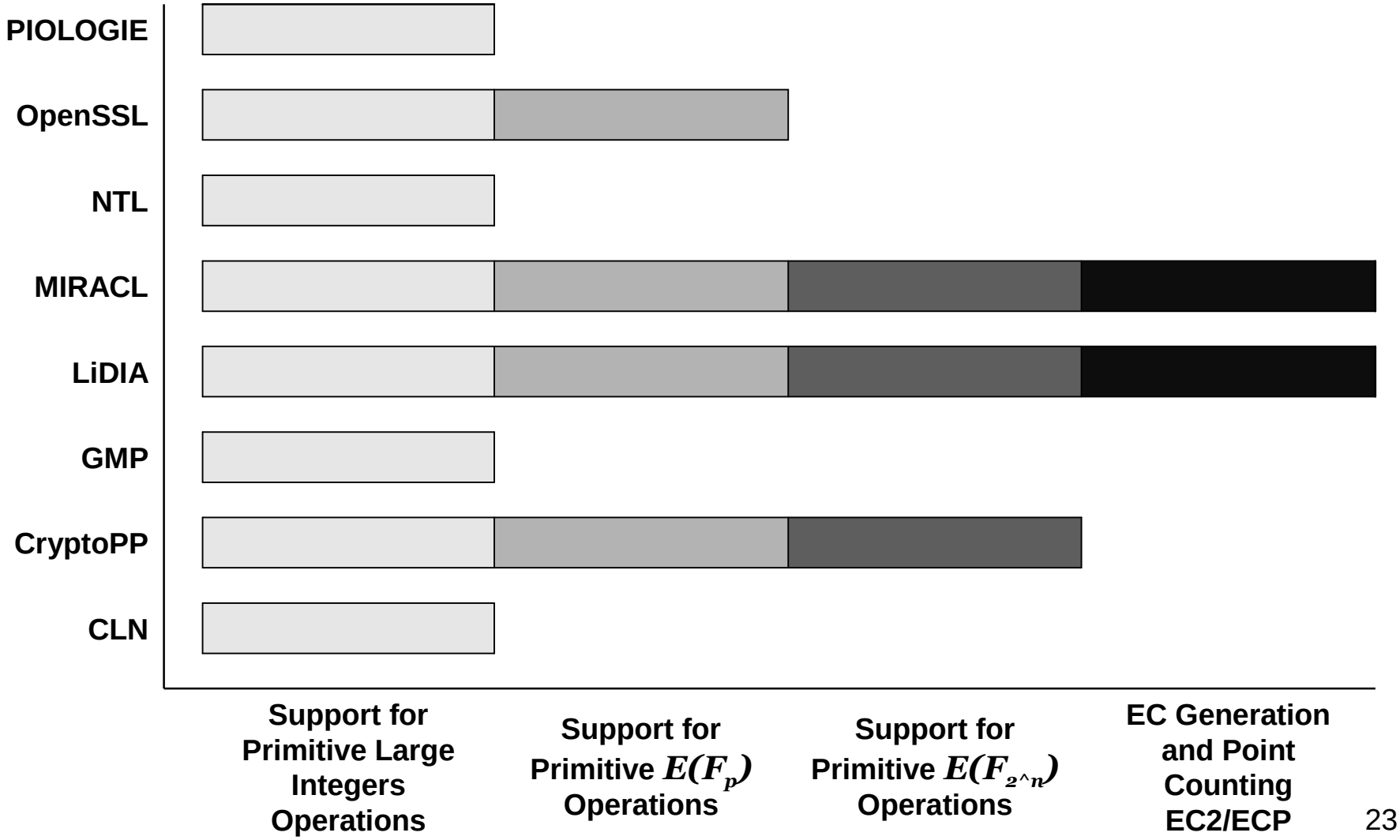
- Low level routines
  - Targeting Pentium 4 and UltraSPARC
- Algorithms
  - Choice of algorithm and algorithm parameters
  - Different implementations

Factors Affecting Performance Examples

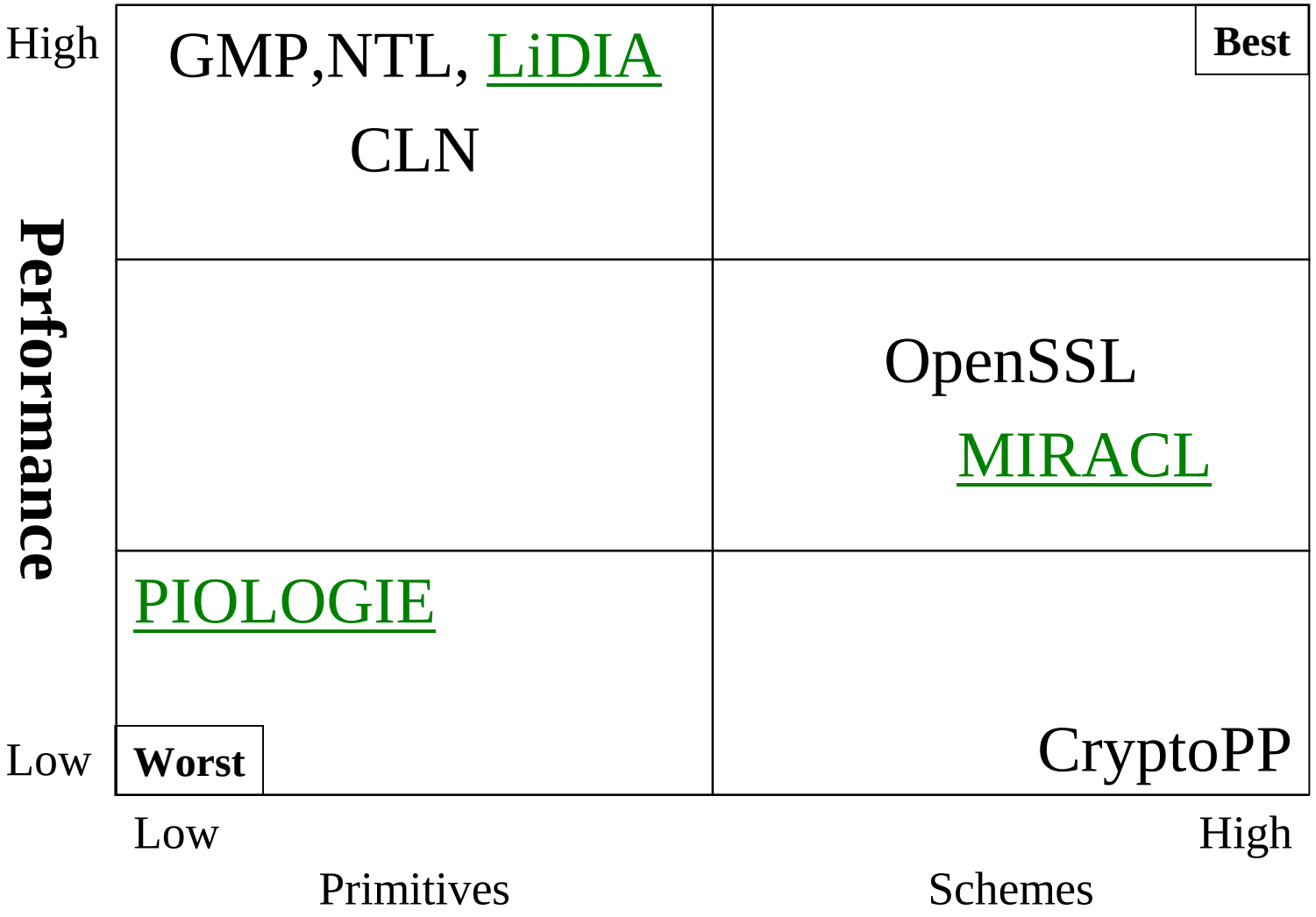


P4-WinXP/MULTIPLICATION (Clock Cycles)			
Library	$result_{768}$	$result_{1024}$	$result_{2048}$
CLN	**8,940**	**11,763**	**29,133**
CryptoPP	**38,432**	**37,928**	**78,755**
GMP	**3,423**	**5,364**	**17,605**
LiDIA	**3,573**	**6,047**	**18,722**
MIRACL	**10,974**	**18,613**	**71,512**
NTL	**3,381**	**5,426**	**17,722**
OpenSSL	**9,055**	**15,218**	**48,438**
Piologie	**29,910**	**41,163**	**113,977**
Min_n	**3,381**	**5,364**	**17,605**

Support For Public Key Cryptosystems



Which library is best for implementing PK Schemes operating on LARGE INTEGERS?



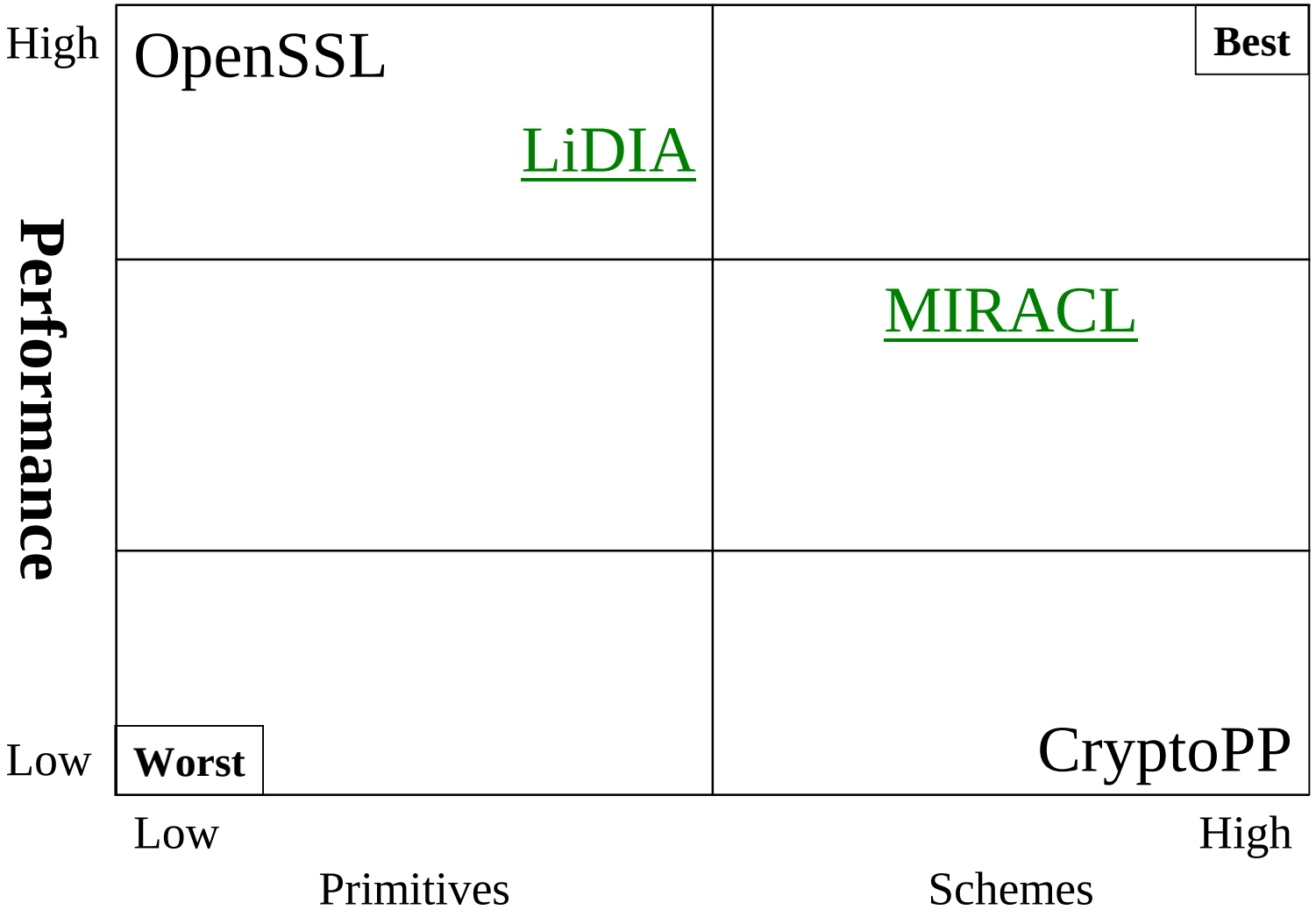
CONCLUSION(1)

- Support for operations on large integers,
  - Group A)
    - {GMP(fastest), NTL, LiDIA, CLN}: best performance under all platforms tested.
    - Trade off: amount of time and effort needed for implementation.
  - Group B)
    - {OpenSSL, MIRACL} trail libraries from group A in terms of overall performance. Support implementations of cryptographic schemes => faster development.
  - {CryptoPP} is the best choice for the fast development based on the wide range of cryptographic schemes implemented.
    - Trade off: performance as compared to other libraries.

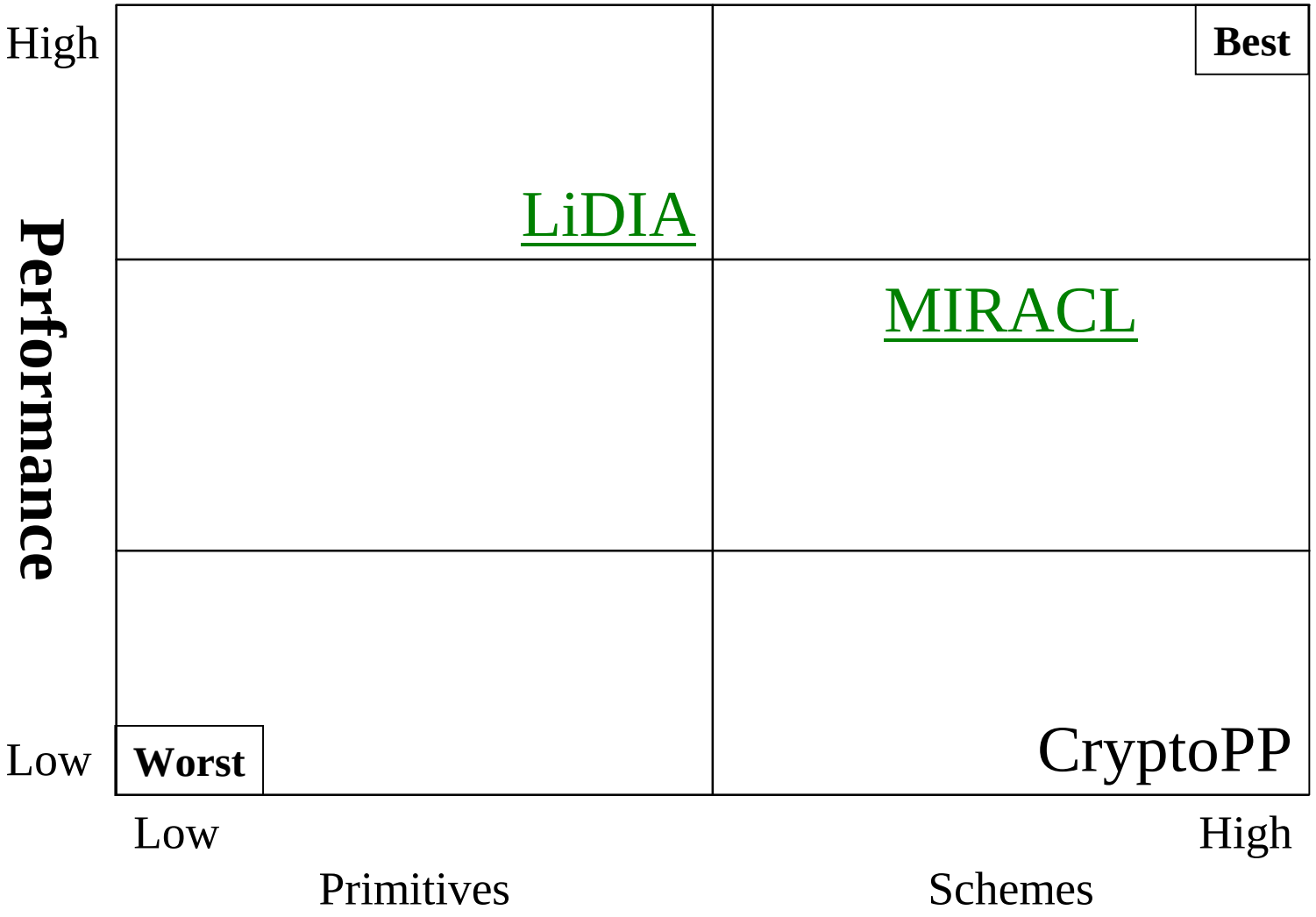
Which library is best for implementing PK Schemes operating on LARGE INTEGERS?

- High Performance :
  - A lot of time devoted for development (low support)
  - Free
    - Use GMP.
- Medium Performance/Medium Time Devoted For Development :
  - Free
    - Use OpenSSL.
- High Support/Not Enough Time to Develop
  - Free
    - Use CryptoPP.

Which library is best for implementing PK Schemes operating on ECP?



Which library is best for implementing PK Schemes operating on EC2?



CONCLUSION(2)

- Support for $E(F_{2^n})$,
  - LiDIA has the best performance under Pentium IV-RedHat 9.0 and UltraSPARC-Solaris. Under Pentium IV, Windows XP, MIRACL has the best performance. CryptoPP is the slowest under all platforms.
- Support for $E(F_p)$,
  - OpenSSL has the best performance under all platforms, LiDIA performance is better than MIRACL on Pentium IVRedHat 9.0 and UltraSPARC-Solaris, while under Pentium IV-Windows XP, MIRACL is better than LiDIA. CryptoPP is the slowest.

Which library is best for implementing PK Schemes operating on ECP?

- High Performance :
  - A lot of time devoted for development (low support)
  - Free
    - Use OpenSSL.
- Medium Performance/Good Support
  - Not Free
    - Use MIRACL.
- High Support/Not Enough Time to Develop
  - Free
    - Use CryptoPP.

Which library is best for implementing PK Schemes operating on EC2?

- High Performance :
  - A lot of time devoted for development (low support)
  - Not Free
    - Use LiDIA.
- Medium Performance/Good Support
  - Not Free
    - Use MIRACL.
- High Support/Not Enough Time to Develop
  - Free
    - Use CryptoPP.

Thank You