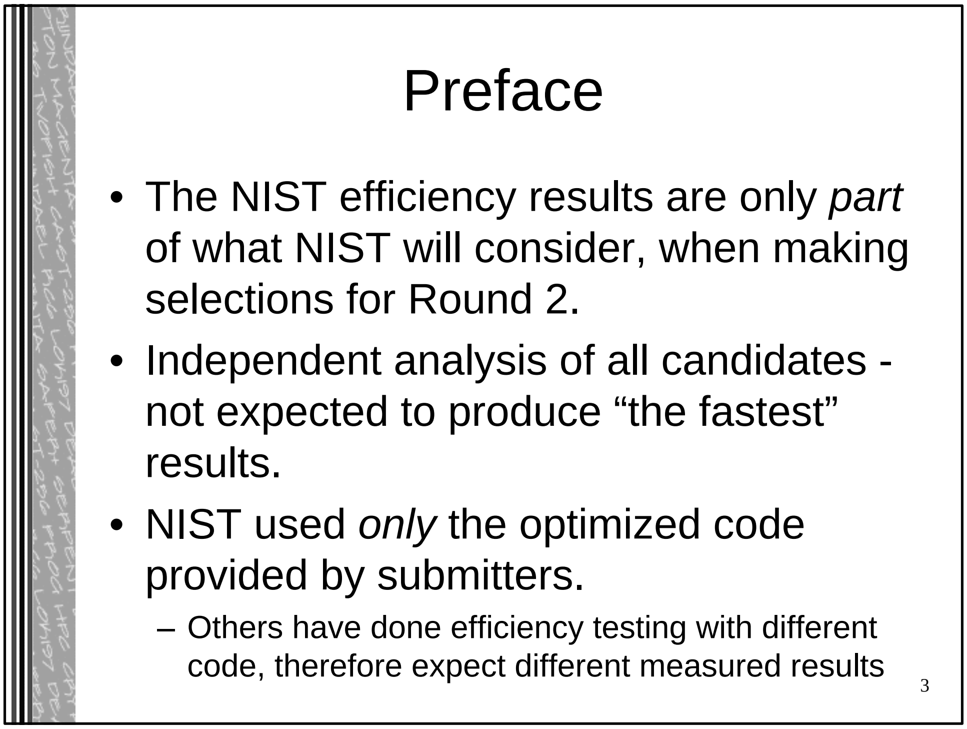
NIST’s Efficiency Testing for Round1 AES Candidates

Security Technology Group
Information Technology Laboratory
NIST

http://www.nist.gov/aes

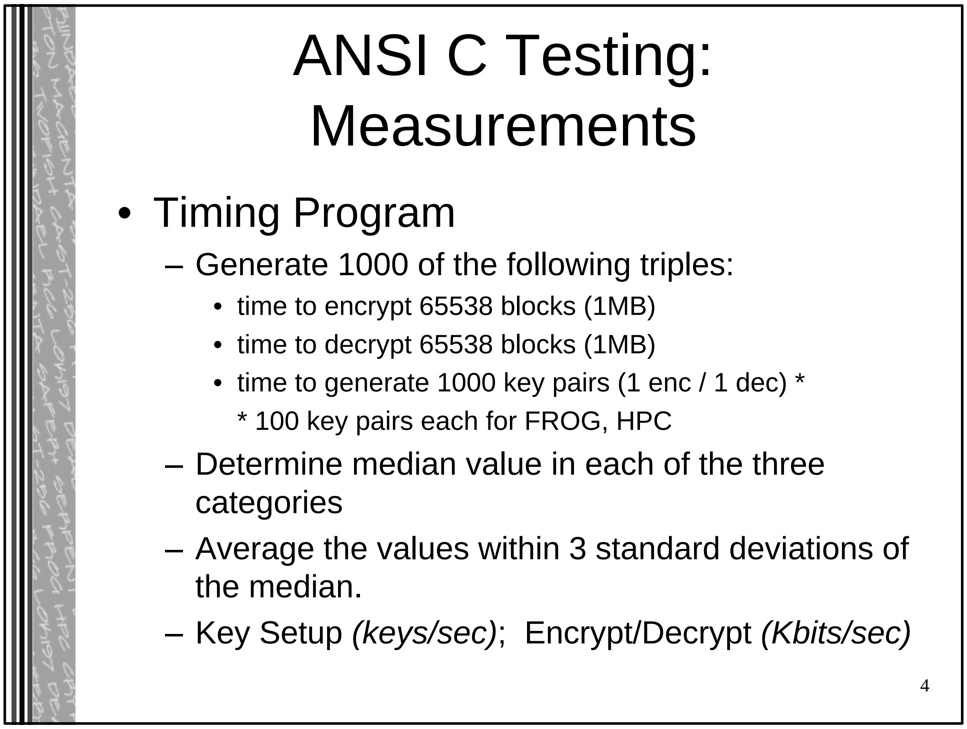
Overview

- ANSI C Testing:
  - – Configurations & measurement techniques
  - – “Reference” platform efficiency (speed) testing results
  - – Comparison of NIST results with other surveys
  - – Average performance, with multiple compilers on multiple platforms
- Java Testing
  - – Speed and Memory measurements



Preface

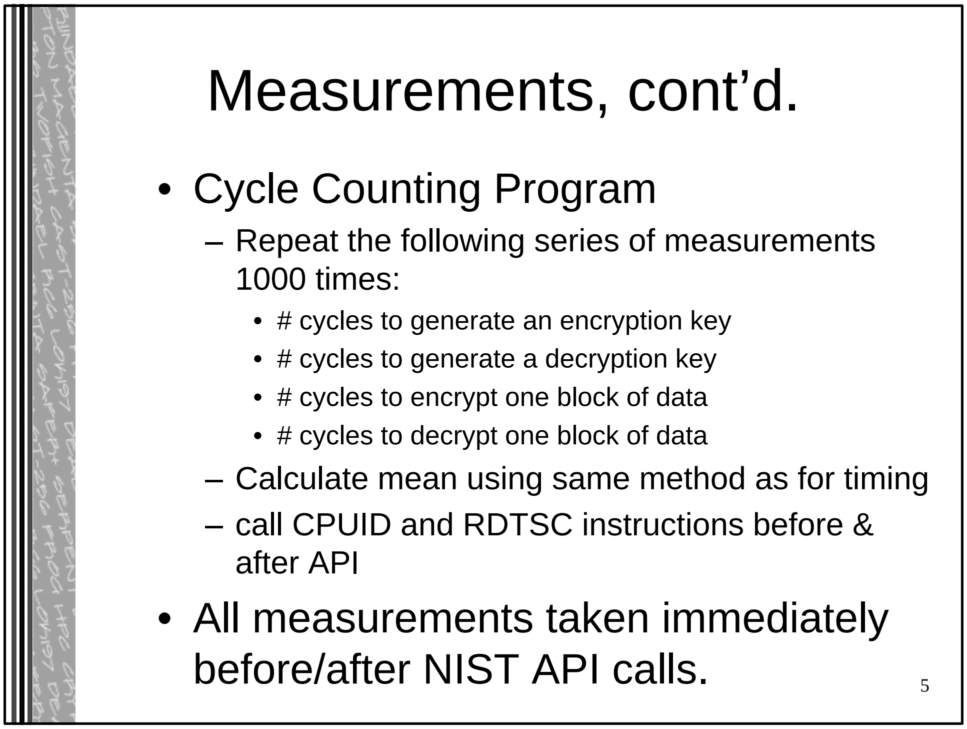
- The NIST efficiency results are only *part* of what NIST will consider, when making selections for Round 2.
- Independent analysis of all candidates - not expected to produce “the fastest” results.
- NIST used *only* the optimized code provided by submitters.
  - Others have done efficiency testing with different code, therefore expect different measured results



ANSI C Testing: Measurements

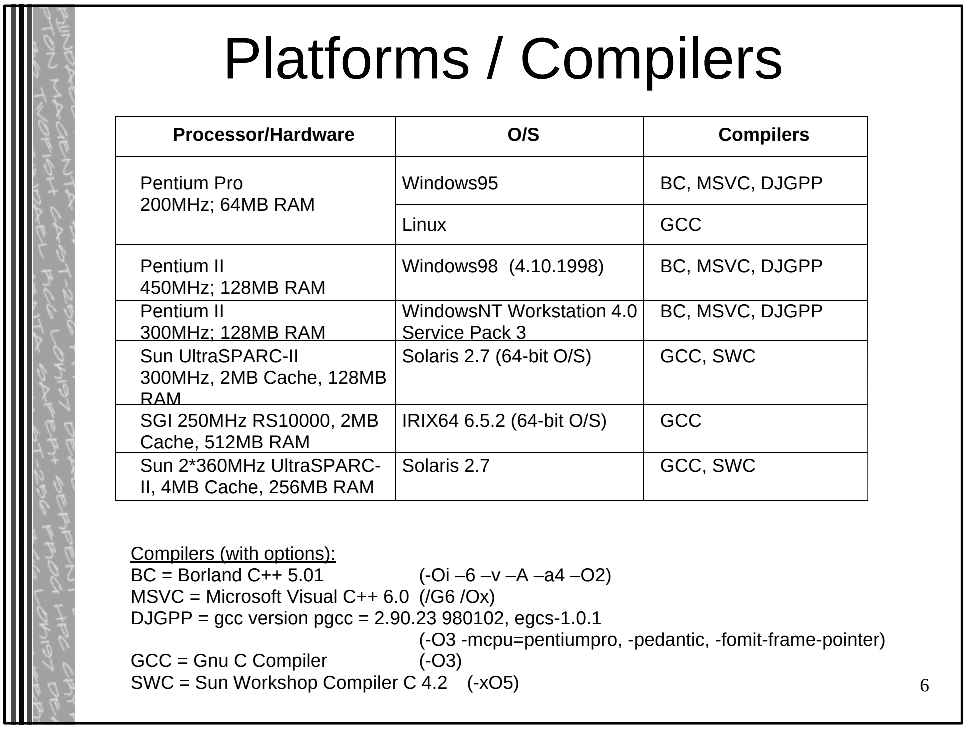
- Timing Program
  - Generate 1000 of the following triples:
    - time to encrypt 65538 blocks (1MB)
    - time to decrypt 65538 blocks (1MB)
    - time to generate 1000 key pairs (1 enc / 1 dec) *

      * 100 key pairs each for FROG, HPC
  - Determine median value in each of the three categories
  - Average the values within 3 standard deviations of the median.
  - Key Setup *(keys/sec)*; Encrypt/Decrypt *(Kbits/sec)*



Measurements, cont'd.

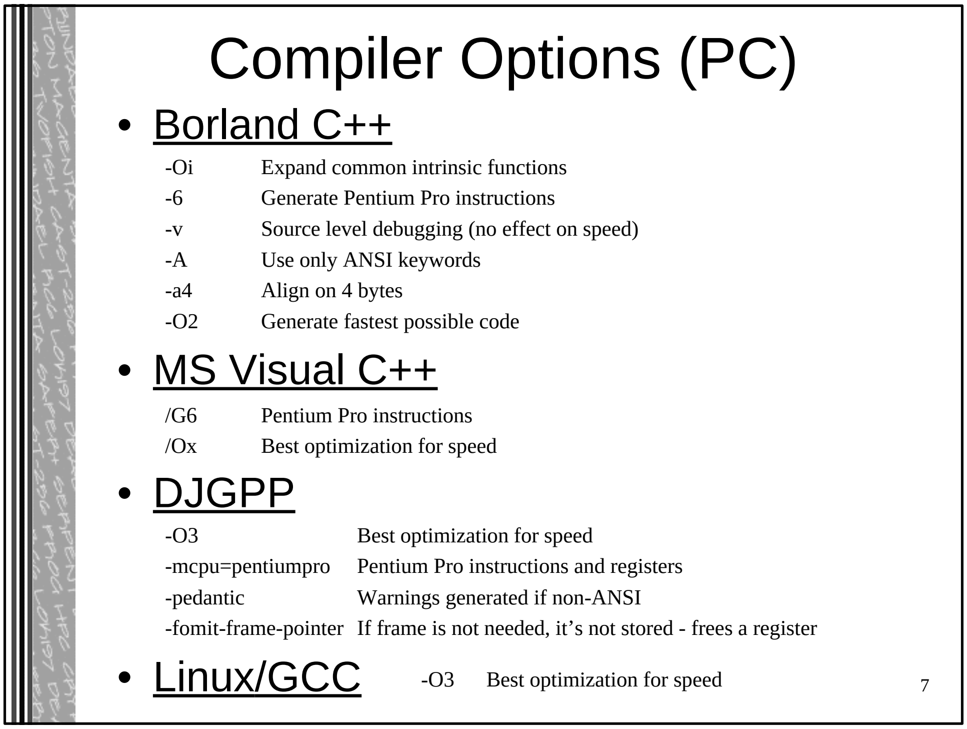
- Cycle Counting Program
  - Repeat the following series of measurements 1000 times:
    - # cycles to generate an encryption key
    - # cycles to generate a decryption key
    - # cycles to encrypt one block of data
    - # cycles to decrypt one block of data
  - Calculate mean using same method as for timing
  - call CPUID and RDTSC instructions before & after API
- All measurements taken immediately before/after NIST API calls.



Platforms / Compilers

Processor/Hardware	O/S	Compilers
Pentium Pro 200MHz; 64MB RAM	Windows95	BC, MSVC, DJGPP
	Linux	GCC
Pentium II 450MHz; 128MB RAM	Windows98 (4.10.1998)	BC, MSVC, DJGPP
Pentium II 300MHz; 128MB RAM	WindowsNT Workstation 4.0 Service Pack 3	BC, MSVC, DJGPP
Sun UltraSPARC-II 300MHz, 2MB Cache, 128MB RAM	Solaris 2.7 (64-bit O/S)	GCC, SWC
SGI 250MHz RS10000, 2MB Cache, 512MB RAM	IRIX64 6.5.2 (64-bit O/S)	GCC
Sun 2*360MHz UltraSPARC-II, 4MB Cache, 256MB RAM	Solaris 2.7	GCC, SWC

Compilers (with options):
BC = Borland C++ 5.01 (-Oi –6 –v –A –a4 –O2)
MSVC = Microsoft Visual C++ 6.0 (/G6 /Ox)
DJGPP = gcc version pgcc = 2.90.23 980102, egcs-1.0.1
(-O3 -mcpu=pentiumpro, -pedantic, -fomit-frame-pointer)
GCC = Gnu C Compiler (-O3)
SWC = Sun Workshop Compiler C 4.2 (-xO5)



Compiler Options (PC)

- <u>Borland C++</u>

-Oi	Expand common intrinsic functions
-6	Generate Pentium Pro instructions
-v	Source level debugging (no effect on speed)
-A	Use only ANSI keywords
-a4	Align on 4 bytes
-O2	Generate fastest possible code

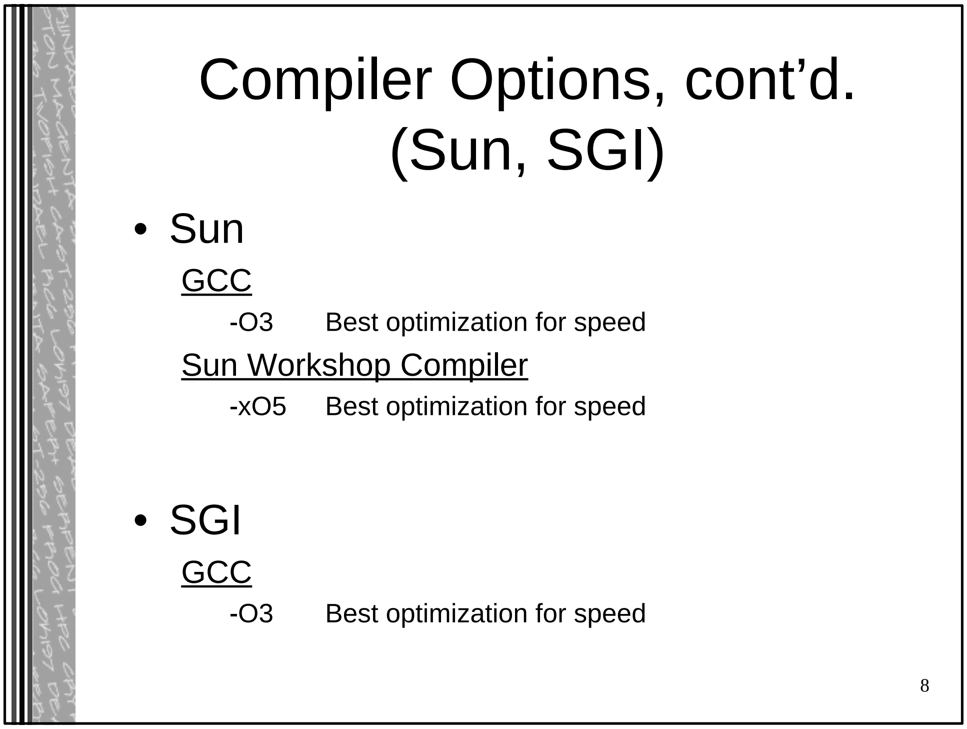
- <u>MS Visual C++</u>

/G6	Pentium Pro instructions
/Ox	Best optimization for speed

- <u>DJGPP</u>

-O3	Best optimization for speed
-mcpu=pentiumpro	Pentium Pro instructions and registers
-pedantic	Warnings generated if non-ANSI
-fomit-frame-pointer	If frame is not needed, it's not stored - frees a register

- <u>Linux/GCC</u> -O3 Best optimization for speed



Compiler Options, cont'd. (Sun, SGI)

- Sun

<u>GCC</u>

-O3	Best optimization for speed

<u>Sun Workshop Compiler</u>

-xO5	Best optimization for speed

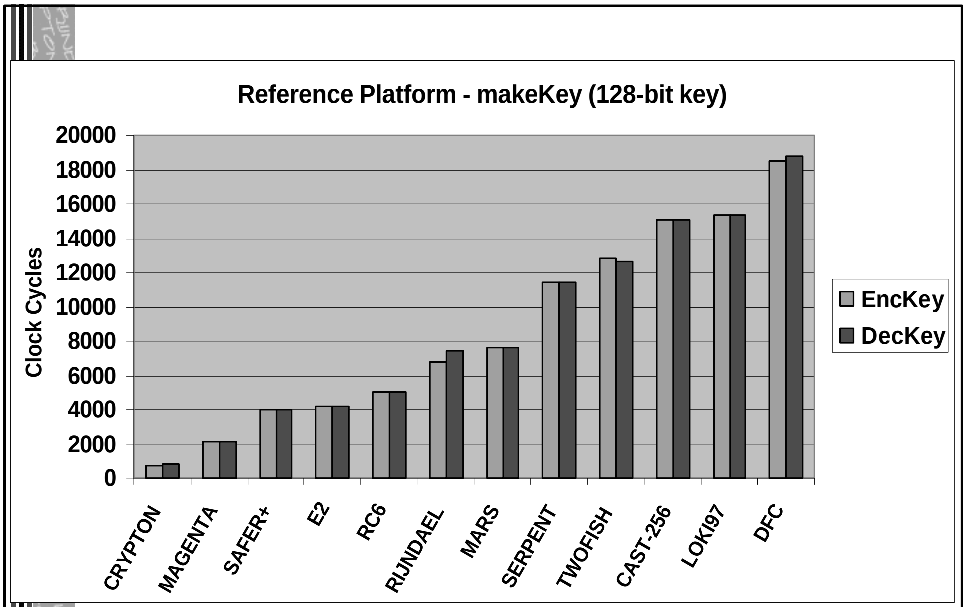
- SGI

<u>GCC</u>

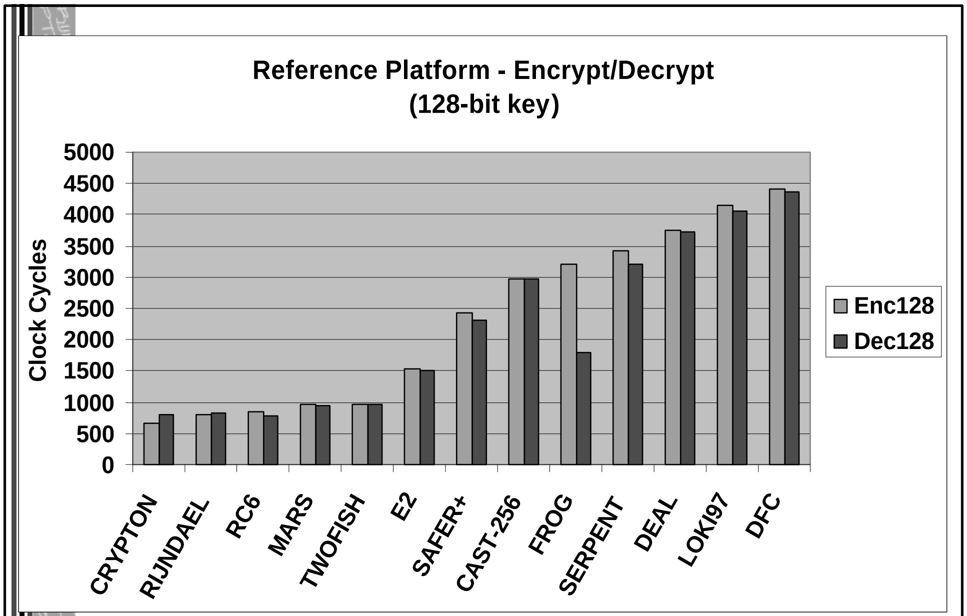
-O3	Best optimization for speed

NIST's "Reference" Configuration

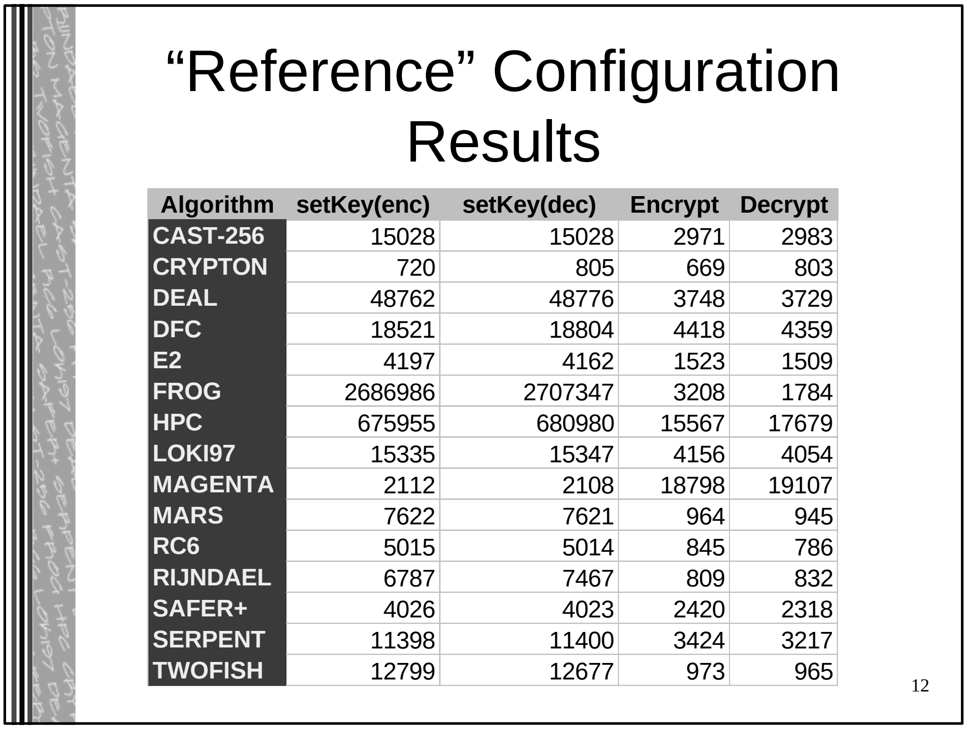
- NIST specified its minimum testing configuration in the call for candidate algorithms:
  - – Pentium Pro, 200MHz, 64MB RAM, Windows95
  - – Borland C++ 5.0 compiler *(everyone's favorite)*
  - – Key Setup, Encryption, Decryption
  - – Round 1: focus on 128-bit key size



*Not shown:	
DEAL	~48700
HPC	~680000
FROG	~2700000

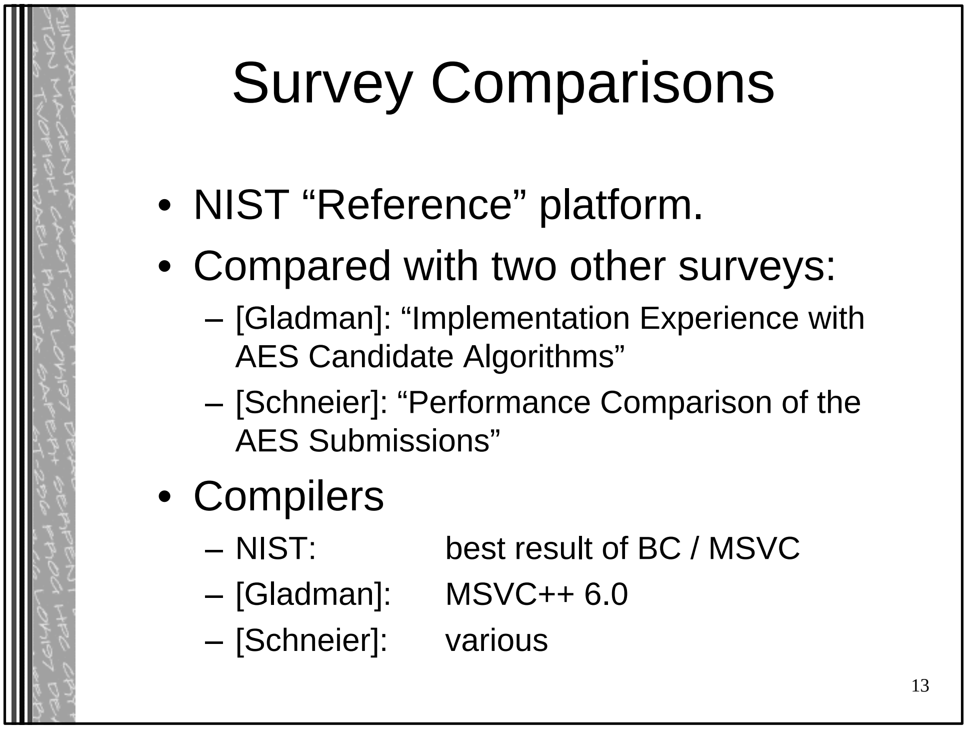


		Encrypt	Decrypt
*Not shown:	HPC	15567	17679
	MAGENTA	18798	19107



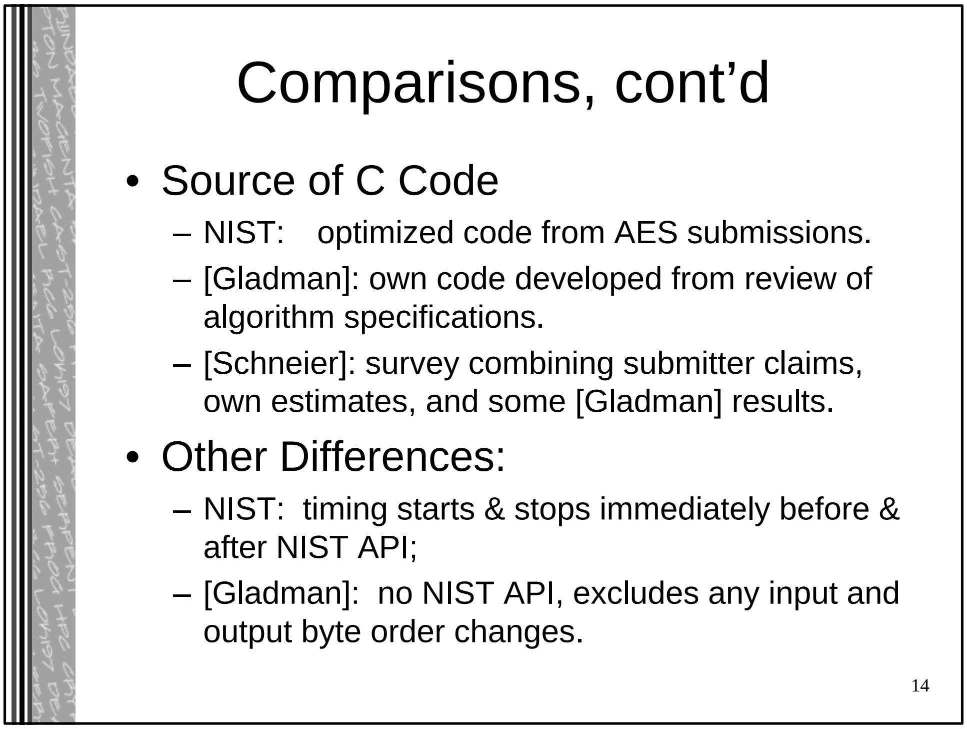
“Reference” Configuration Results

Algorithm	setKey(enc)	setKey(dec)	Encrypt	Decrypt
CAST-256	15028	15028	2971	2983
CRYPTON	720	805	669	803
DEAL	48762	48776	3748	3729
DFC	18521	18804	4418	4359
E2	4197	4162	1523	1509
FROG	2686986	2707347	3208	1784
HPC	675955	680980	15567	17679
LOKI97	15335	15347	4156	4054
MAGENTA	2112	2108	18798	19107
MARS	7622	7621	964	945
RC6	5015	5014	845	786
RIJNDAEL	6787	7467	809	832
SAFER+	4026	4023	2420	2318
SERPENT	11398	11400	3424	3217
TWOFISH	12799	12677	973	965



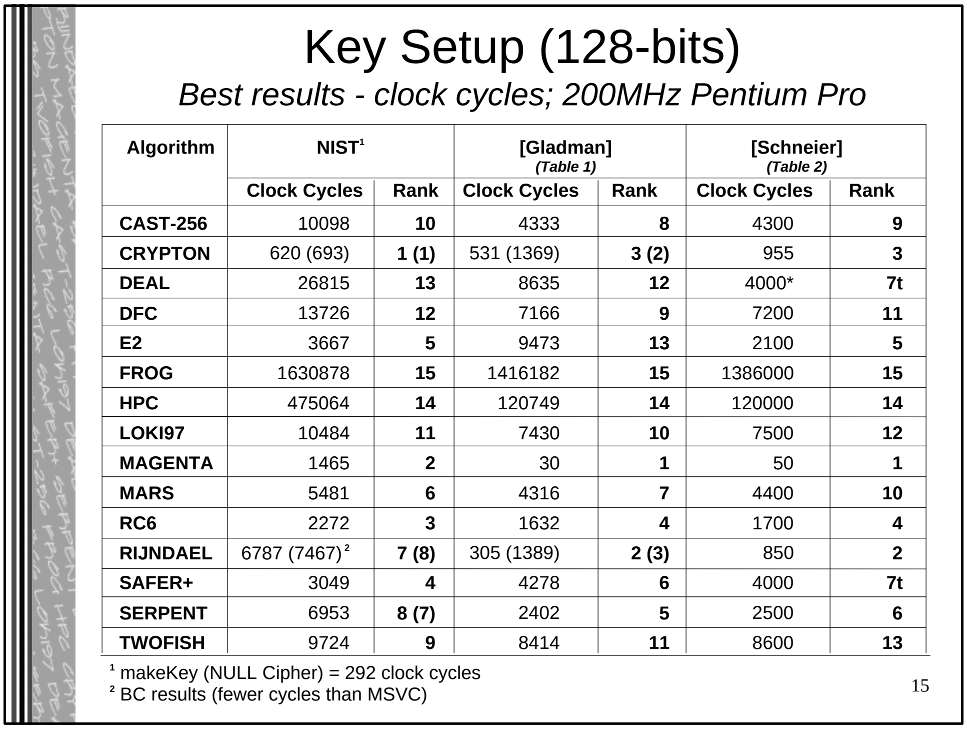
Survey Comparisons

- NIST “Reference” platform.
- Compared with two other surveys:
  - [Gladman]: “Implementation Experience with AES Candidate Algorithms”
  - [Schneier]: “Performance Comparison of the AES Submissions”
- Compilers
  - NIST: best result of BC / MSVC
  - [Gladman]: MSVC++ 6.0
  - [Schneier]: various



Comparisons, cont’d

- Source of C Code
  - NIST: optimized code from AES submissions.
  - [Gladman]: own code developed from review of algorithm specifications.
  - [Schneier]: survey combining submitter claims, own estimates, and some [Gladman] results.
- Other Differences:
  - NIST: timing starts & stops immediately before & after NIST API;
  - [Gladman]: no NIST API, excludes any input and output byte order changes.

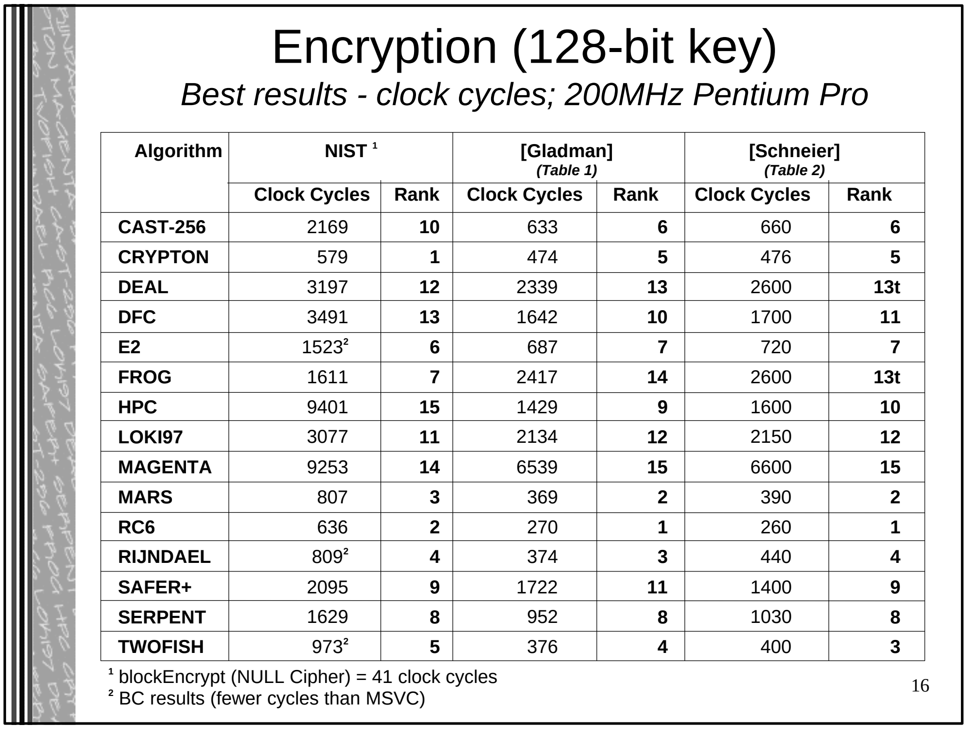


Key Setup (128-bits)

Best results - clock cycles; 200MHz Pentium Pro

Algorithm	**NIST[1]**		**[Gladman]** ***(Table 1)***		**[Schneier]** ***(Table 2)***	
	Clock Cycles	**Rank**	**Clock Cycles**	**Rank**	**Clock Cycles**	**Rank**
CAST-256	10098	**10**	4333	**8**	4300	**9**
CRYPTON	620 (693)	**1 (1)**	531 (1369)	**3 (2)**	955	**3**
DEAL	26815	**13**	8635	**12**	4000*	**7t**
DFC	13726	**12**	7166	**9**	7200	**11**
E2	3667	**5**	9473	**13**	2100	**5**
FROG	1630878	**15**	1416182	**15**	1386000	**15**
HPC	475064	**14**	120749	**14**	120000	**14**
LOKI97	10484	**11**	7430	**10**	7500	**12**
MAGENTA	1465	**2**	30	**1**	50	**1**
MARS	5481	**6**	4316	**7**	4400	**10**
RC6	2272	**3**	1632	**4**	1700	**4**
RIJNDAEL	6787 (7467)[2]	**7 (8)**	305 (1389)	**2 (3)**	850	**2**
SAFER+	3049	**4**	4278	**6**	4000	**7t**
SERPENT	6953	**8 (7)**	2402	**5**	2500	**6**
TWOFISH	9724	**9**	8414	**11**	8600	**13**

[1] makeKey (NULL Cipher) = 292 clock cycles
[2] BC results (fewer cycles than MSVC)

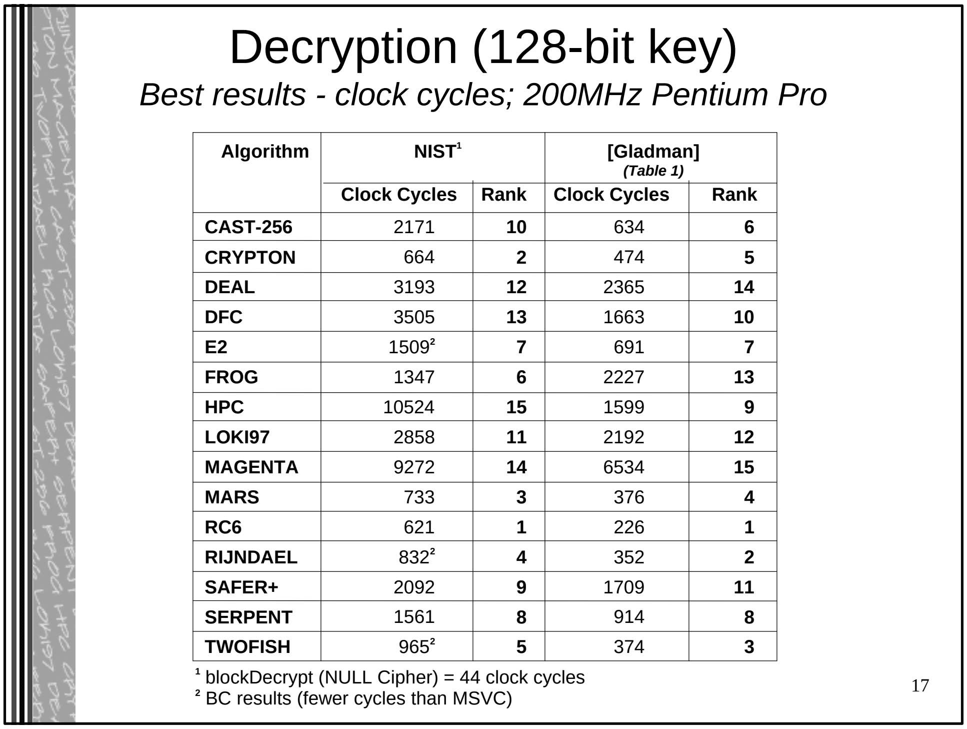


Encryption (128-bit key)

Best results - clock cycles; 200MHz Pentium Pro

Algorithm	**NIST [1]**		**[Gladman]** ***(Table 1)***		**[Schneier]** ***(Table 2)***	
	Clock Cycles	**Rank**	**Clock Cycles**	**Rank**	**Clock Cycles**	**Rank**
CAST-256	2169	**10**	633	**6**	660	**6**
CRYPTON	579	**1**	474	**5**	476	**5**
DEAL	3197	**12**	2339	**13**	2600	**13t**
DFC	3491	**13**	1642	**10**	1700	**11**
E2	1523[2]	**6**	687	**7**	720	**7**
FROG	1611	**7**	2417	**14**	2600	**13t**
HPC	9401	**15**	1429	**9**	1600	**10**
LOKI97	3077	**11**	2134	**12**	2150	**12**
MAGENTA	9253	**14**	6539	**15**	6600	**15**
MARS	807	**3**	369	**2**	390	**2**
RC6	636	**2**	270	**1**	260	**1**
RIJNDAEL	809[2]	**4**	374	**3**	440	**4**
SAFER+	2095	**9**	1722	**11**	1400	**9**
SERPENT	1629	**8**	952	**8**	1030	**8**
TWOFISH	973[2]	**5**	376	**4**	400	**3**

[1] blockEncrypt (NULL Cipher) = 41 clock cycles
[2] BC results (fewer cycles than MSVC)

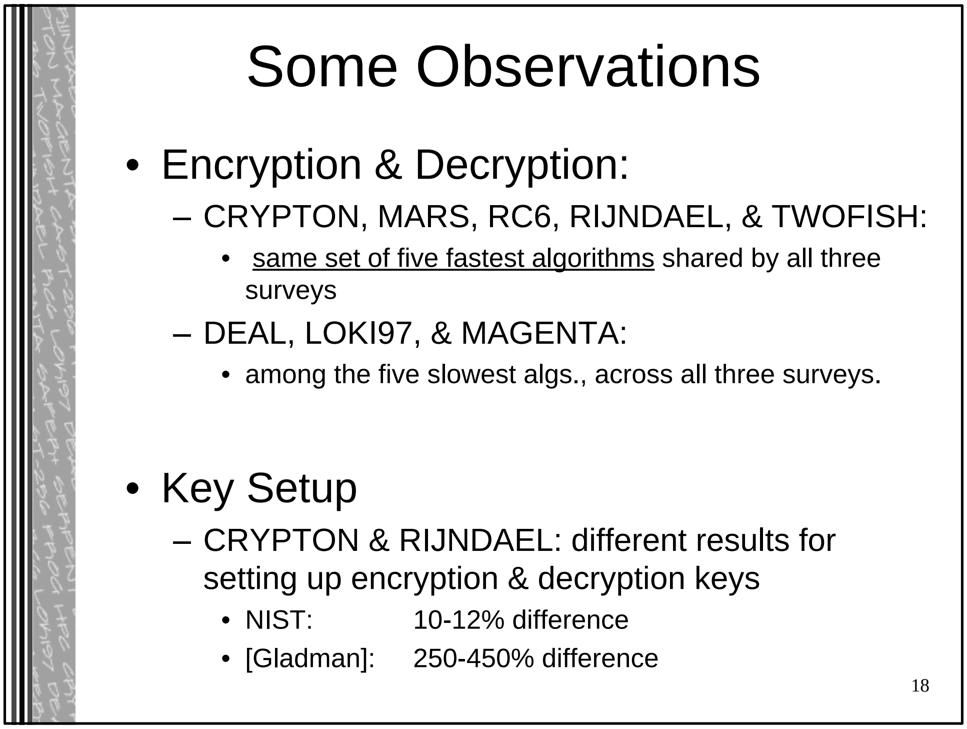


Decryption (128-bit key)

Best results - clock cycles; 200MHz Pentium Pro

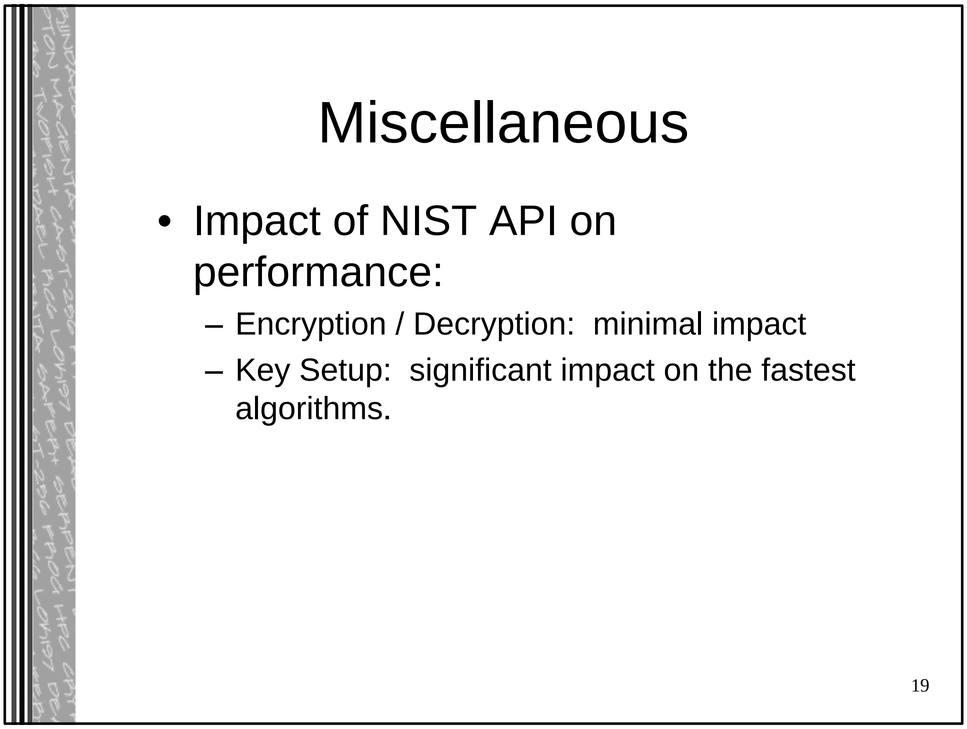
Algorithm	NIST[1]		[Gladman] *(Table 1)*	
	Clock Cycles	Rank	Clock Cycles	Rank
CAST-256	2171	**10**	634	**6**
CRYPTON	664	**2**	474	**5**
DEAL	3193	**12**	2365	**14**
DFC	3505	**13**	1663	**10**
E2	1509[2]	**7**	691	**7**
FROG	1347	**6**	2227	**13**
HPC	10524	**15**	1599	**9**
LOKI97	2858	**11**	2192	**12**
MAGENTA	9272	**14**	6534	**15**
MARS	733	**3**	376	**4**
RC6	621	**1**	226	**1**
RIJNDAEL	832[2]	**4**	352	**2**
SAFER+	2092	**9**	1709	**11**
SERPENT	1561	**8**	914	**8**
TWOFISH	965[2]	**5**	374	**3**

[1] blockDecrypt (NULL Cipher) = 44 clock cycles
[2] BC results (fewer cycles than MSVC)



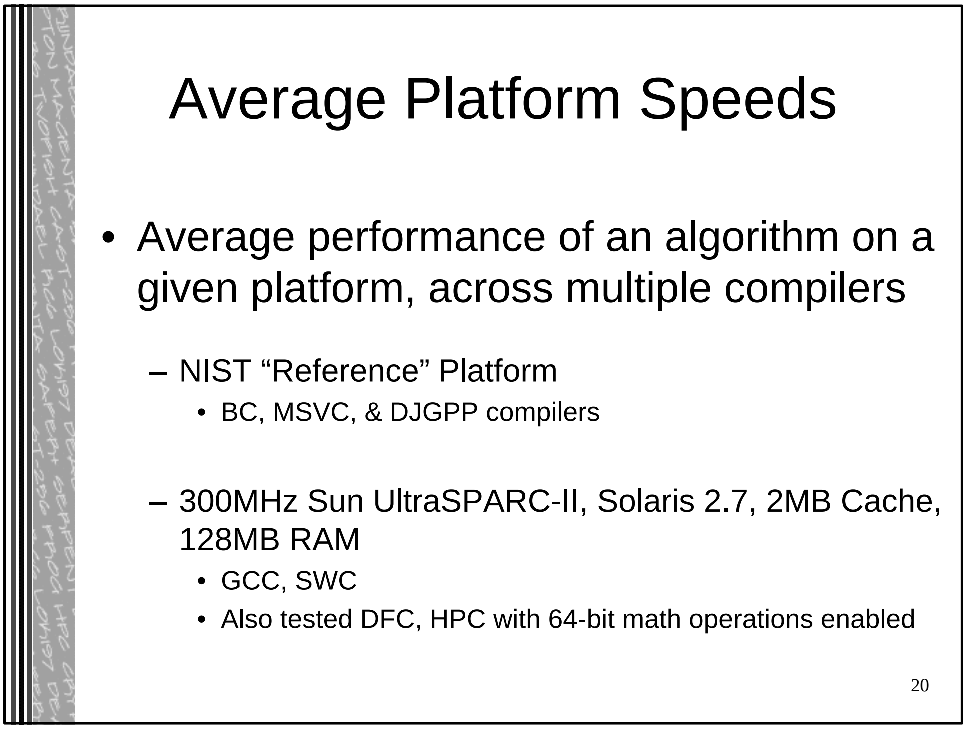
Some Observations

- Encryption & Decryption:
  - CRYPTON, MARS, RC6, RIJNDAEL, & TWOFISH:
    - same set of five fastest algorithms shared by all three surveys
  - DEAL, LOKI97, & MAGENTA:
    - among the five slowest algs., across all three surveys.
- Key Setup
  - CRYPTON & RIJNDAEL: different results for setting up encryption & decryption keys
    - NIST: 10-12% difference
    - [Gladman]: 250-450% difference



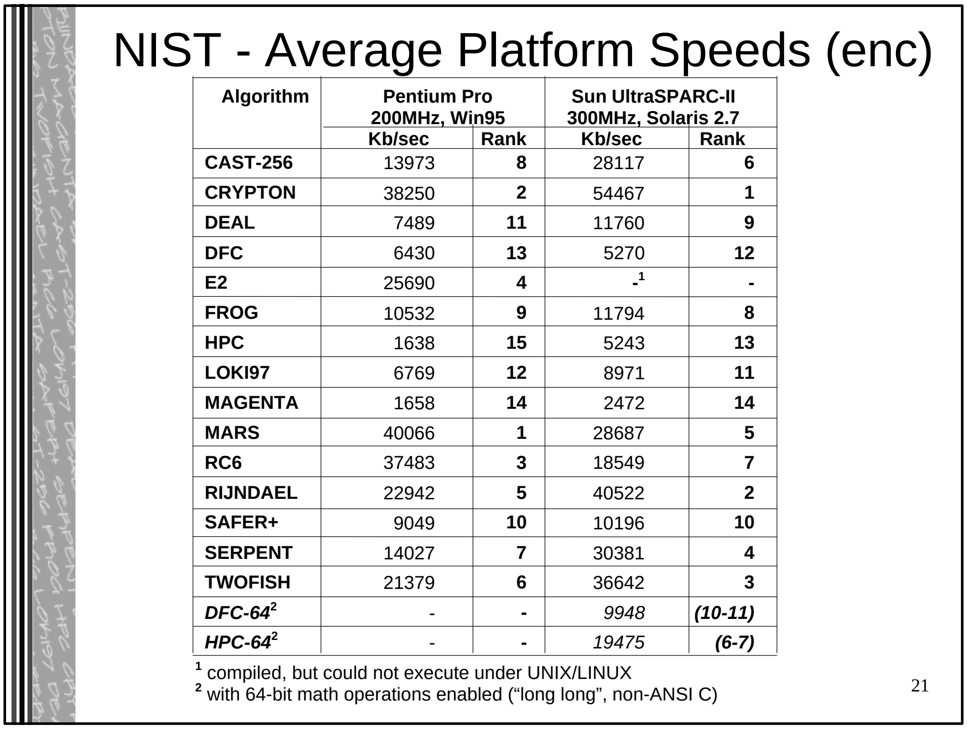
Miscellaneous

- Impact of NIST API on performance:
  - – Encryption / Decryption: minimal impact
  - – Key Setup: significant impact on the fastest algorithms.



Average Platform Speeds

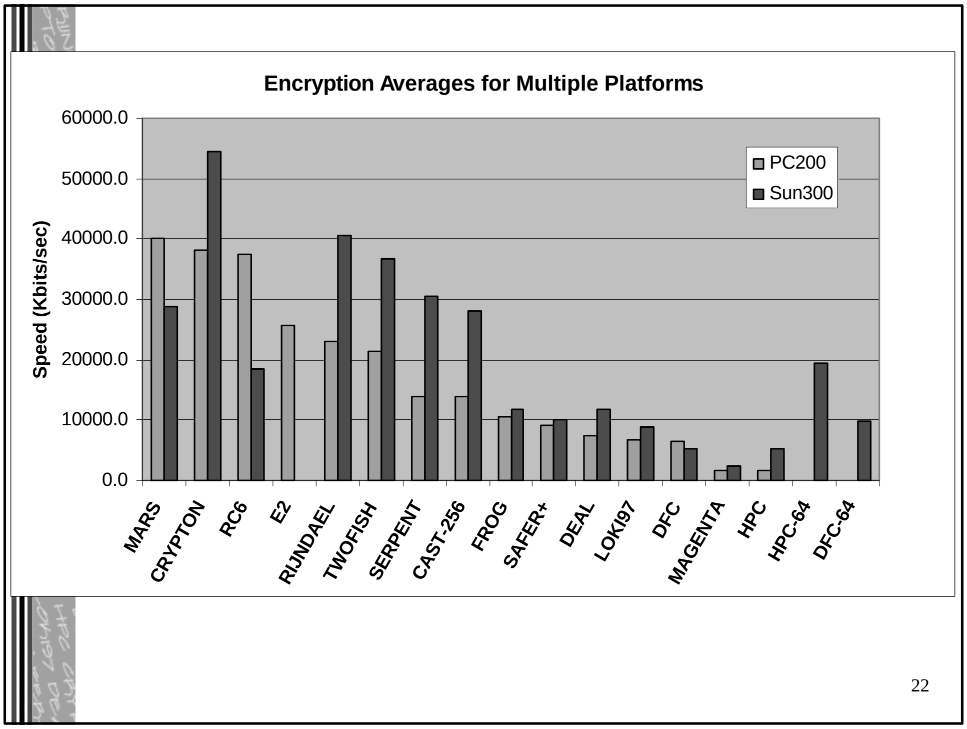
- Average performance of an algorithm on a given platform, across multiple compilers
  - – NIST “Reference” Platform
    - BC, MSVC, & DJGPP compilers
  - – 300MHz Sun UltraSPARC-II, Solaris 2.7, 2MB Cache, 128MB RAM
    - GCC, SWC
    - Also tested DFC, HPC with 64-bit math operations enabled



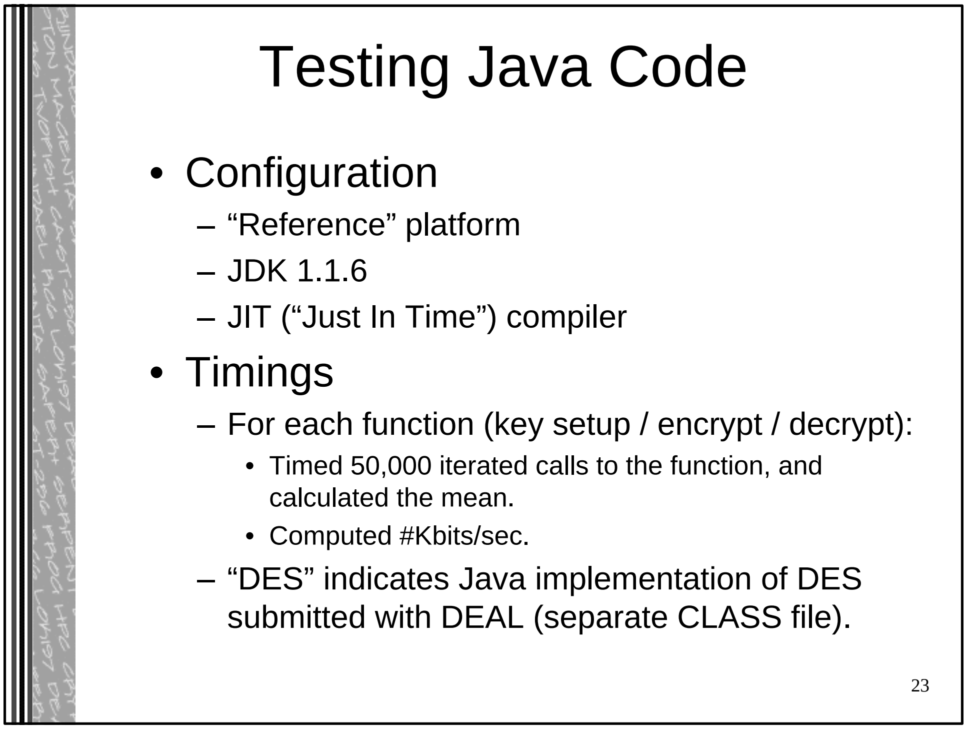
NIST - Average Platform Speeds (enc)

Algorithm	Pentium Pro 200MHz, Win95		Sun UltraSPARC-II 300MHz, Solaris 2.7	
	Kb/sec	Rank	Kb/sec	Rank
CAST-256	13973	**8**	28117	**6**
CRYPTON	38250	**2**	54467	**1**
DEAL	7489	**11**	11760	**9**
DFC	6430	**13**	5270	**12**
E2	25690	**4**	-[1]	**-**
FROG	10532	**9**	11794	**8**
HPC	1638	**15**	5243	**13**
LOKI97	6769	**12**	8971	**11**
MAGENTA	1658	**14**	2472	**14**
MARS	40066	**1**	28687	**5**
RC6	37483	**3**	18549	**7**
RIJNDAEL	22942	**5**	40522	**2**
SAFER+	9049	**10**	10196	**10**
SERPENT	14027	**7**	30381	**4**
TWOFISH	21379	**6**	36642	**3**
DFC-64[2]	-	**-**	*9948*	***(10-11)***
HPC-64[2]	-	**-**	*19475*	***(6-7)***

[1] compiled, but could not execute under UNIX/LINUX
[2] with 64-bit math operations enabled (“long long”, non-ANSI C)



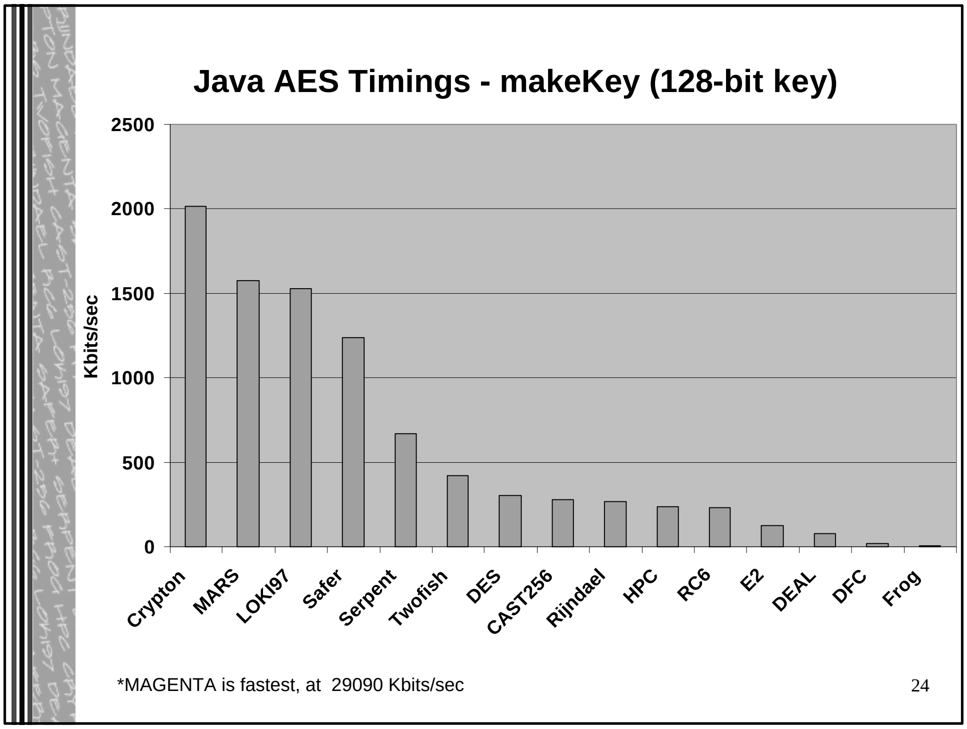
Encryption Averages for Multiple Platforms



Testing Java Code

- Configuration
  - “Reference” platform
  - JDK 1.1.6
  - JIT (“Just In Time”) compiler
- Timings
  - For each function (key setup / encrypt / decrypt):
    - Timed 50,000 iterated calls to the function, and calculated the mean.
    - Computed #Kbits/sec.
  - “DES” indicates Java implementation of DES submitted with DEAL (separate CLASS file).

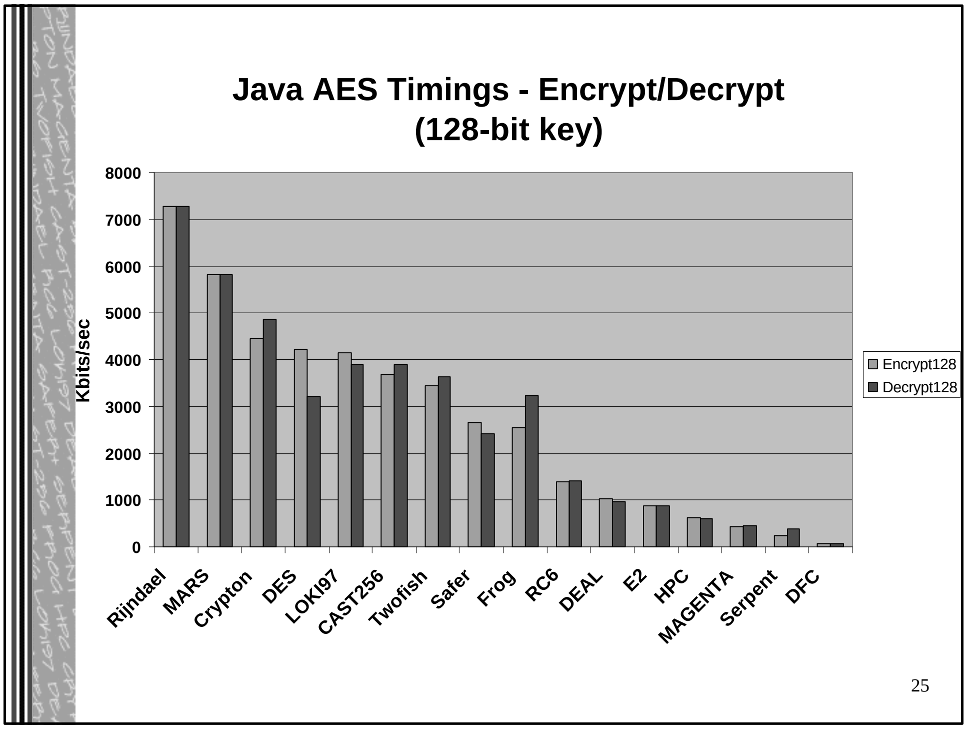
23



Java AES Timings - makeKey (128-bit key)

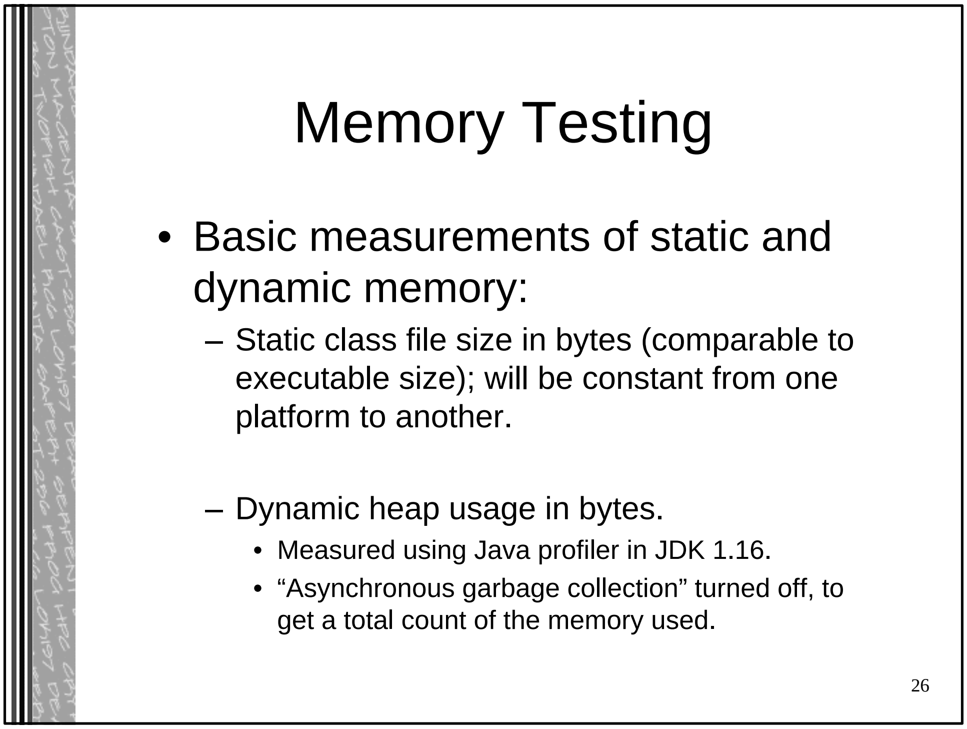
*MAGENTA is fastest, at 29090 Kbits/sec

24



Java AES Timings - Encrypt/Decrypt (128-bit key)

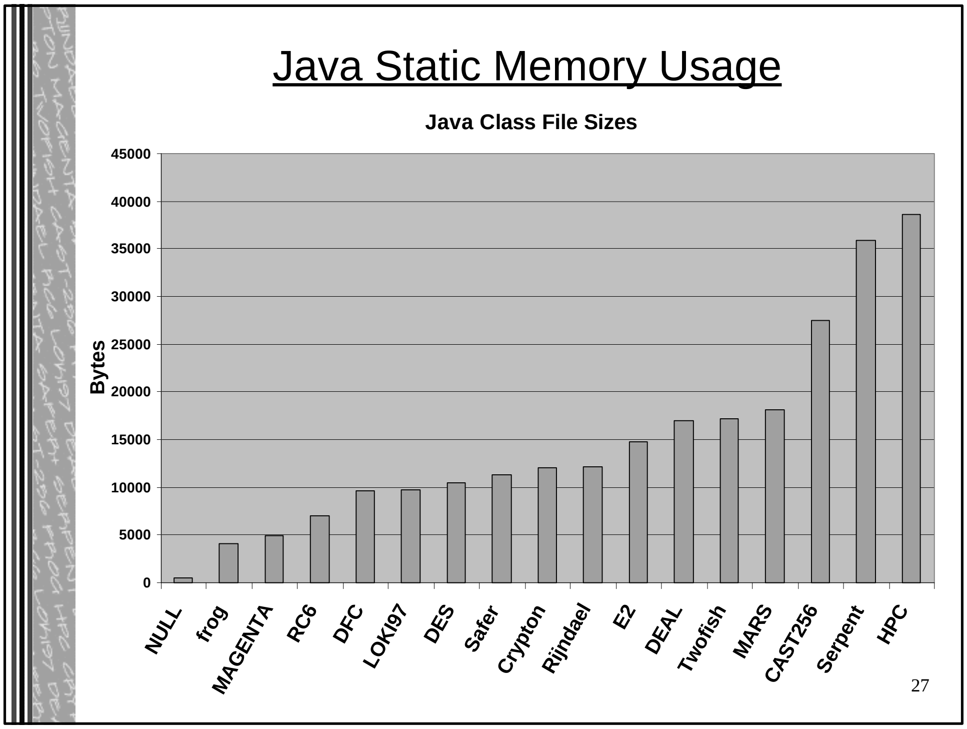
25



Memory Testing

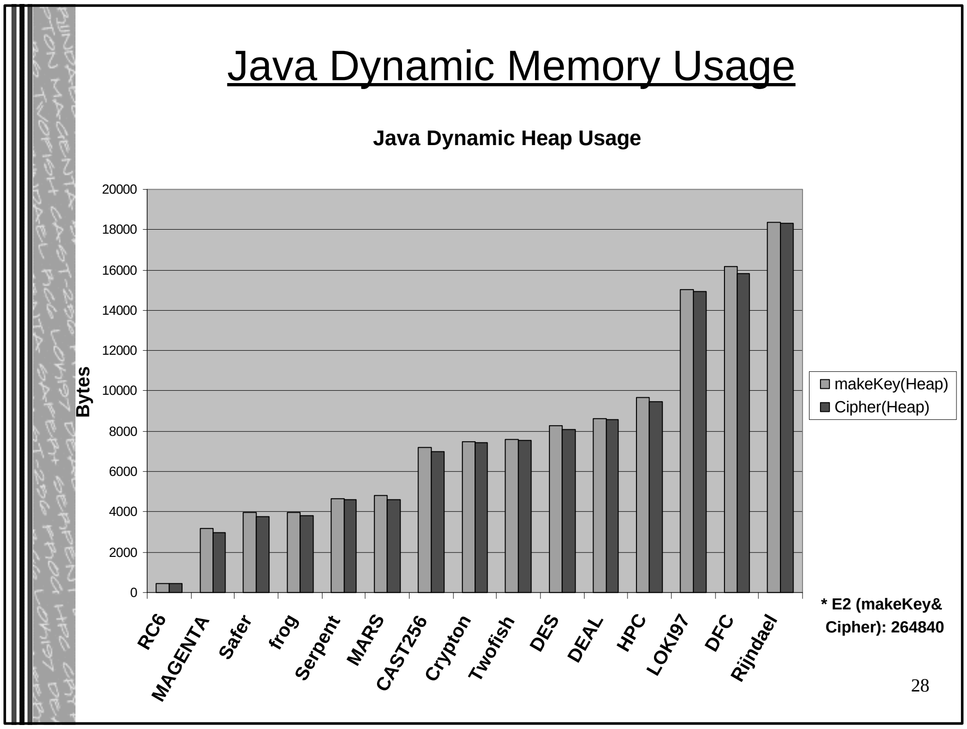
- Basic measurements of static and dynamic memory:
  - – Static class file size in bytes (comparable to executable size); will be constant from one platform to another.
  - – Dynamic heap usage in bytes.
    - Measured using Java profiler in JDK 1.16.
    - “Asynchronous garbage collection” turned off, to get a total count of the memory used.

26



Java Static Memory Usage

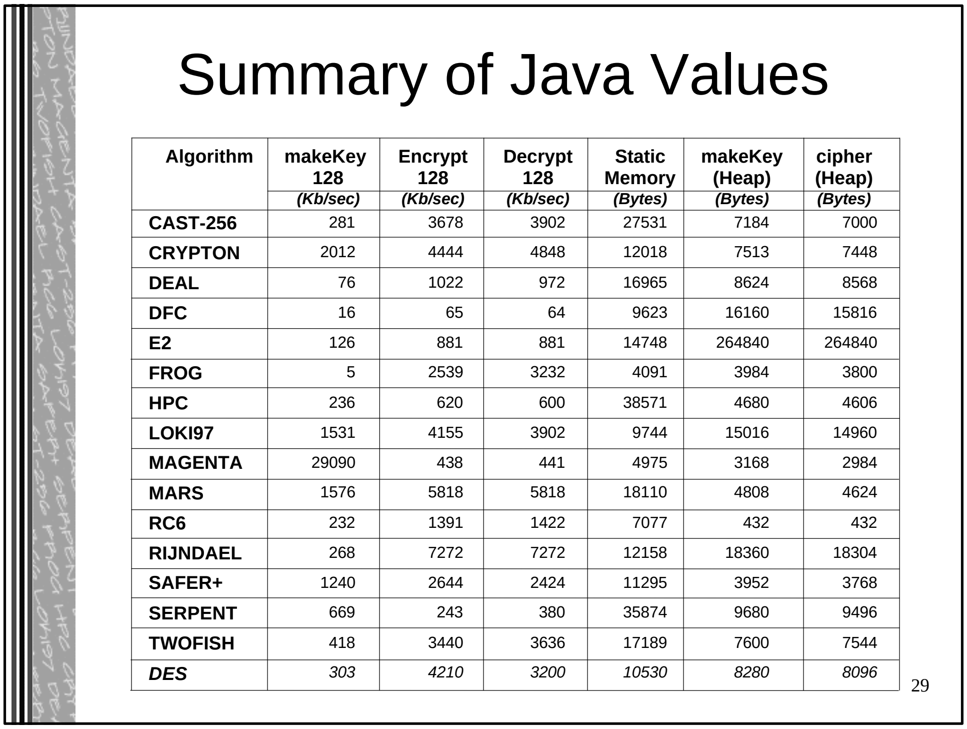
27



Java Dynamic Memory Usage

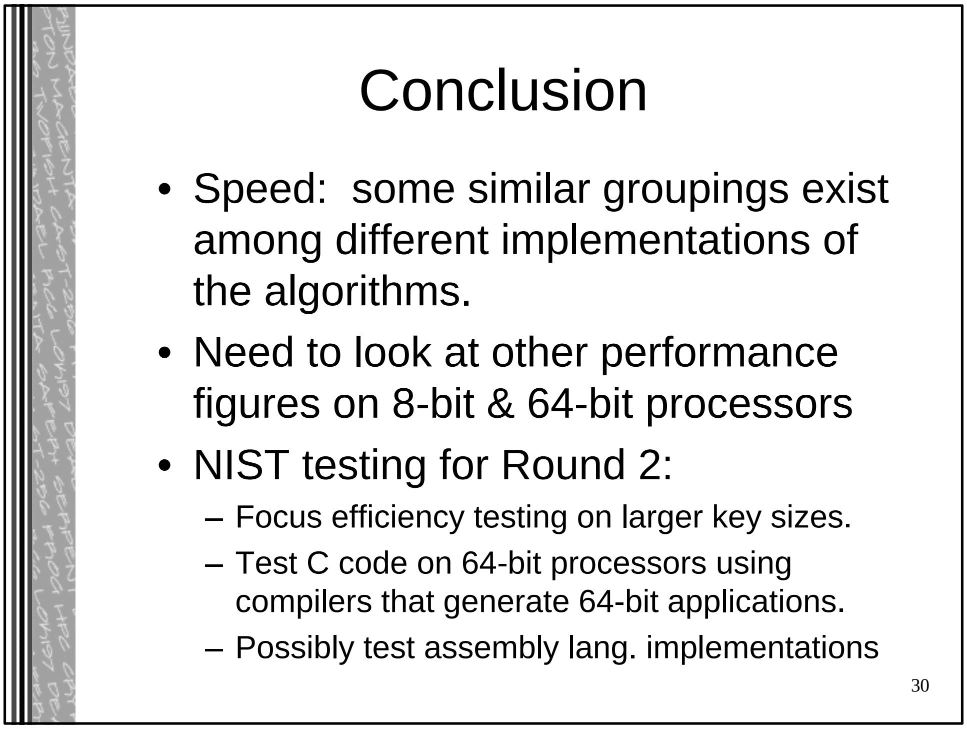
* E2 (makeKey& Cipher): 264840

28



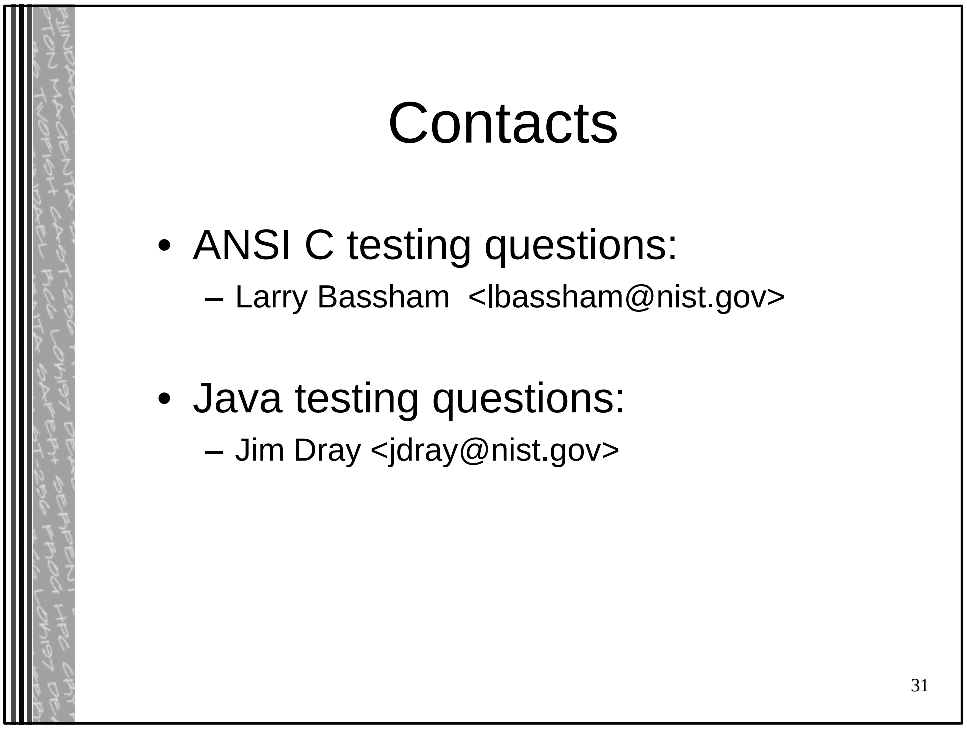
Summary of Java Values

Algorithm	makeKey 128	Encrypt 128	Decrypt 128	Static Memory	makeKey (Heap)	cipher (Heap)
	(Kb/sec)	*(Kb/sec)*	*(Kb/sec)*	*(Bytes)*	*(Bytes)*	*(Bytes)*
CAST-256	281	3678	3902	27531	7184	7000
CRYPTON	2012	4444	4848	12018	7513	7448
DEAL	76	1022	972	16965	8624	8568
DFC	16	65	64	9623	16160	15816
E2	126	881	881	14748	264840	264840
FROG	5	2539	3232	4091	3984	3800
HPC	236	620	600	38571	4680	4606
LOKI97	1531	4155	3902	9744	15016	14960
MAGENTA	29090	438	441	4975	3168	2984
MARS	1576	5818	5818	18110	4808	4624
RC6	232	1391	1422	7077	432	432
RIJNDAEL	268	7272	7272	12158	18360	18304
SAFER+	1240	2644	2424	11295	3952	3768
SERPENT	669	243	380	35874	9680	9496
TWOFISH	418	3440	3636	17189	7600	7544
DES	*303*	*4210*	*3200*	*10530*	*8280*	*8096*



Conclusion

- Speed: some similar groupings exist among different implementations of the algorithms.
- Need to look at other performance figures on 8-bit & 64-bit processors
- NIST testing for Round 2:
  - Focus efficiency testing on larger key sizes.
  - Test C code on 64-bit processors using compilers that generate 64-bit applications.
  - Possibly test assembly lang. implementations



Contacts

- ANSI C testing questions:
  - Larry Bassham <lbassham@nist.gov>
- Java testing questions:
  - Jim Dray <jdray@nist.gov>