Securing Wireless Data : System Architecture Challenges

Srivaths Ravi, Anand Raghunathan and Nachiketh Potlapally

International Symposium on Systems Synthesis, ACM October 2002

林佑青 2003.1.3

Outline

- Introduction
- System design challenges
- Wireless security processing gap
- System design methodologies
- Performance
- Conclusion

Introduction

- The deployment of high-speed wireless data and multimedia communications ushers in new and greater security challenges.
- Wireless clients are much more constrained in their processing capabilities and energy supplies.
- Bottleneck：Security processing gap、Battery gap.
- Mobile security processing system design methodologies.

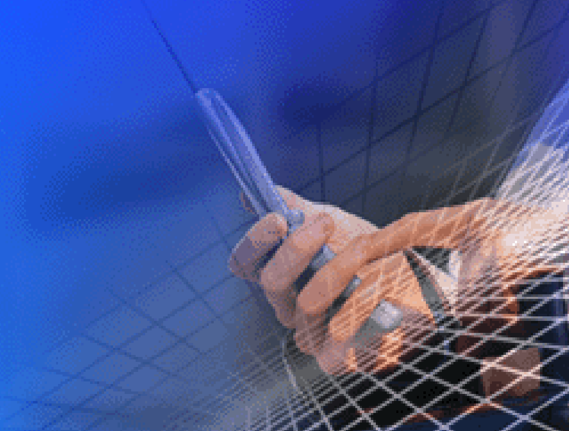
System design challenges

- Security processing gap
- Battery gap

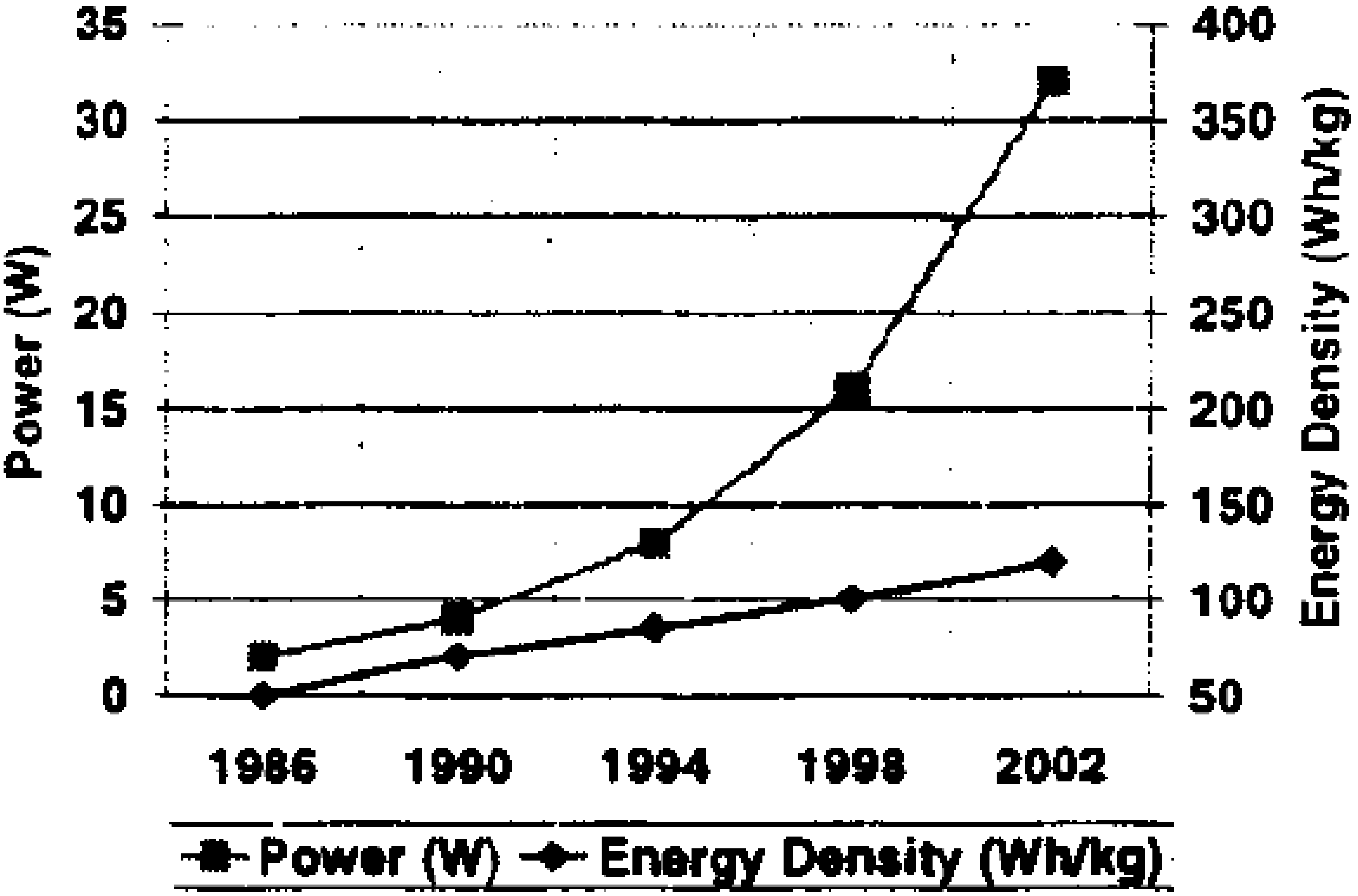
  growth is far slower than the processing requirements.
- Flexibility

  different protocol stack and networks
- Tamper-proof implementation

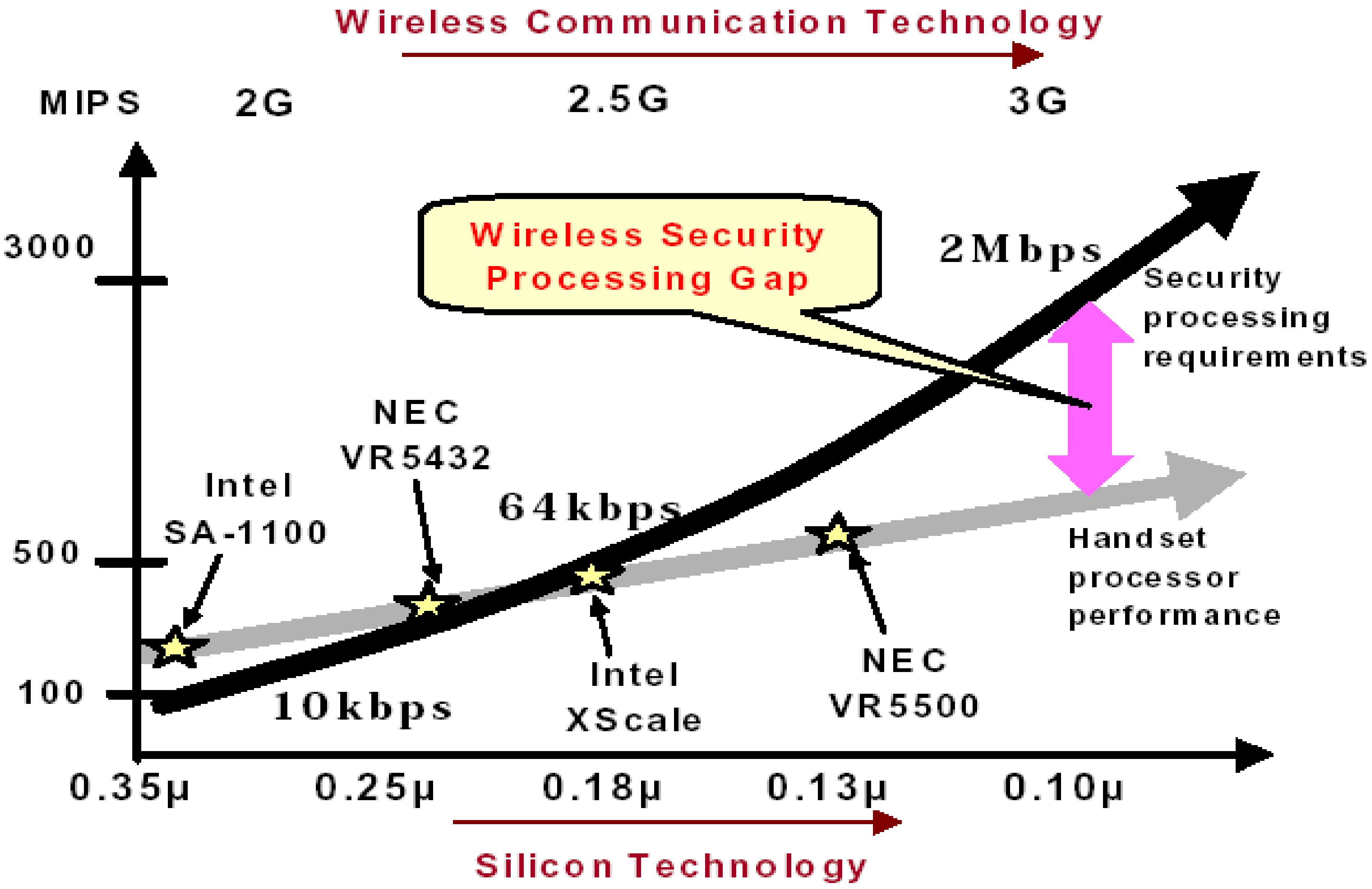
  Biometric identification techniques



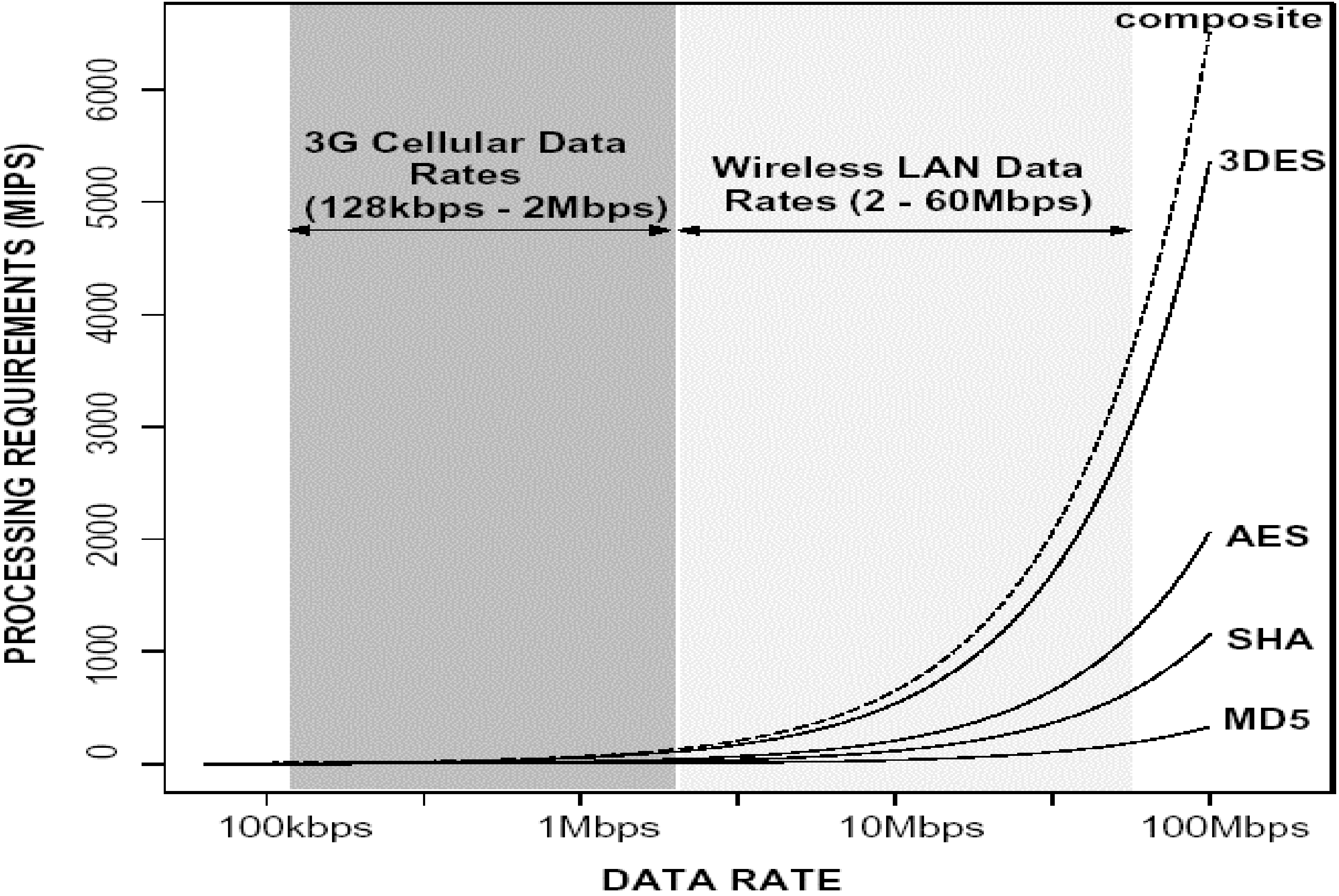
Battery gap


35
30
25
20
15
10
5
0
Power (W)
400
350
300
250
200
150
100
50
Energy Density (Wh/kg)
1986
1990
1994
1998
2002
Power (W)
Energy Density (Wh/kg)

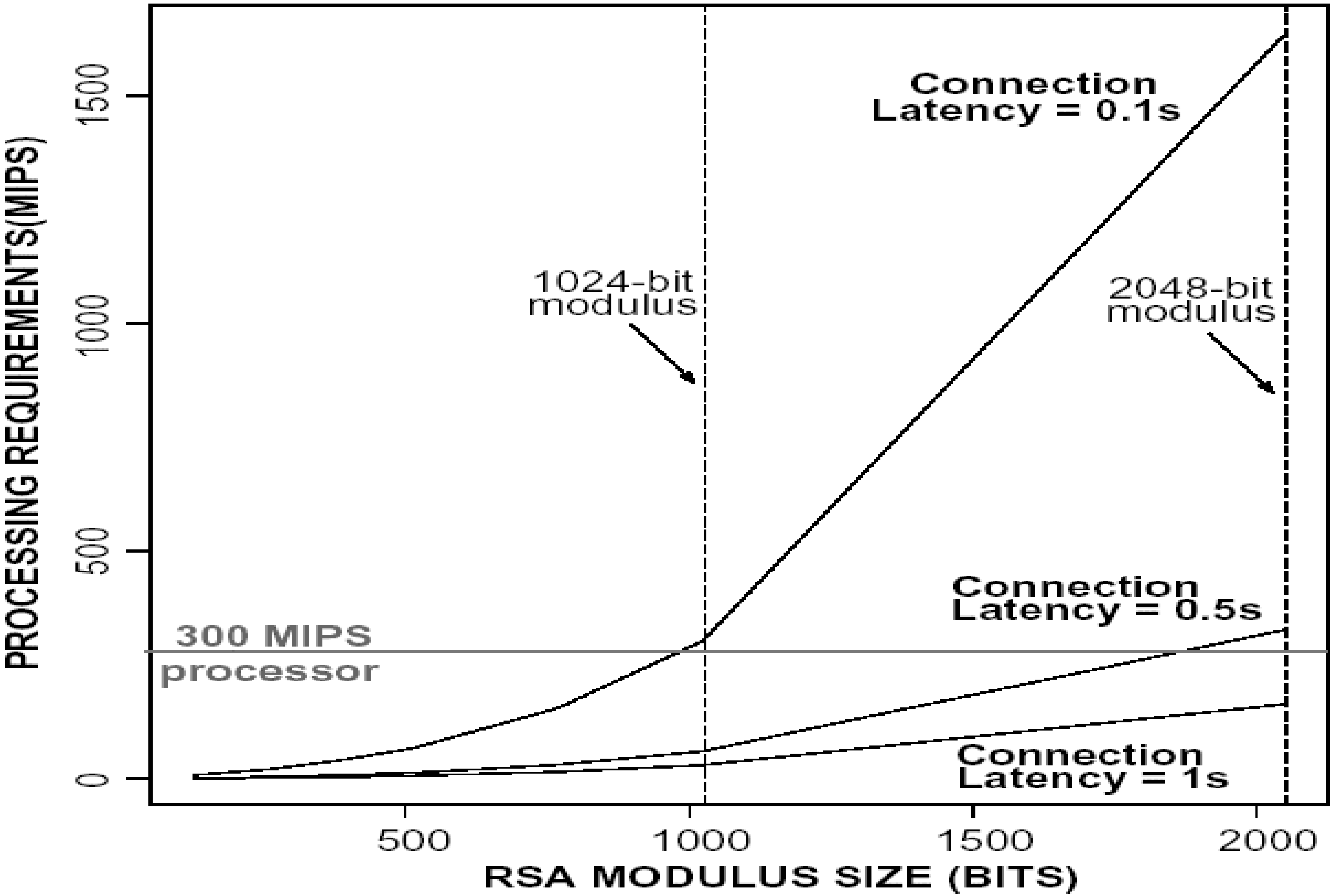
Security Processing Gap (1)

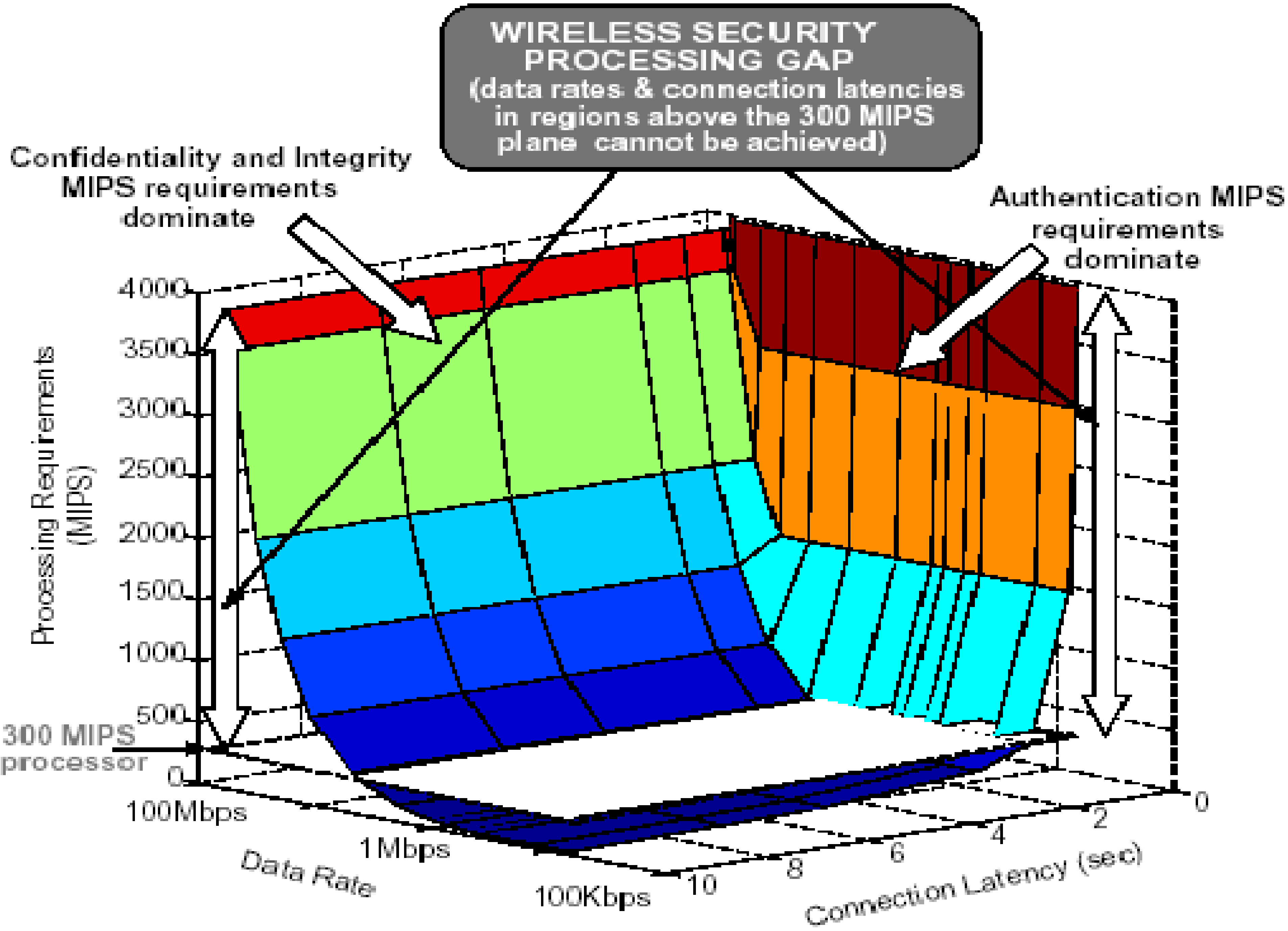


Security processing gap (2)



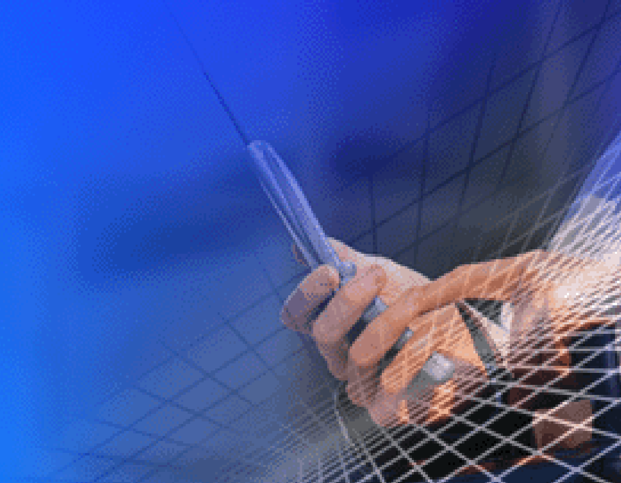
Processing requirements of RSA-based SSL handshake




WIRELESS SECURITY PROCESSING GAP
(data rates & connection latencies in regions above the 300 MIPS plane cannot be achieved)
Confidentiality and Integrity MIPS requirements dominate
Authentication MIPS requirements dominate
Processing Requirements (MIPS)
4000
3500
3000
2500
2000
1500
1000
500
0
300 MIPS processor
100Mbps
1Mbps
100Kbps
Data Rate
10
8
6
4
2
0
Connection Latency (sec)

Bridging the wireless security processing gap

- Low complexity security protocols and cryptographic algorithms.
- Embedded processors with enhanced security processing capabilities.
- MOSES : MObile SEcurity processing System



MOSES

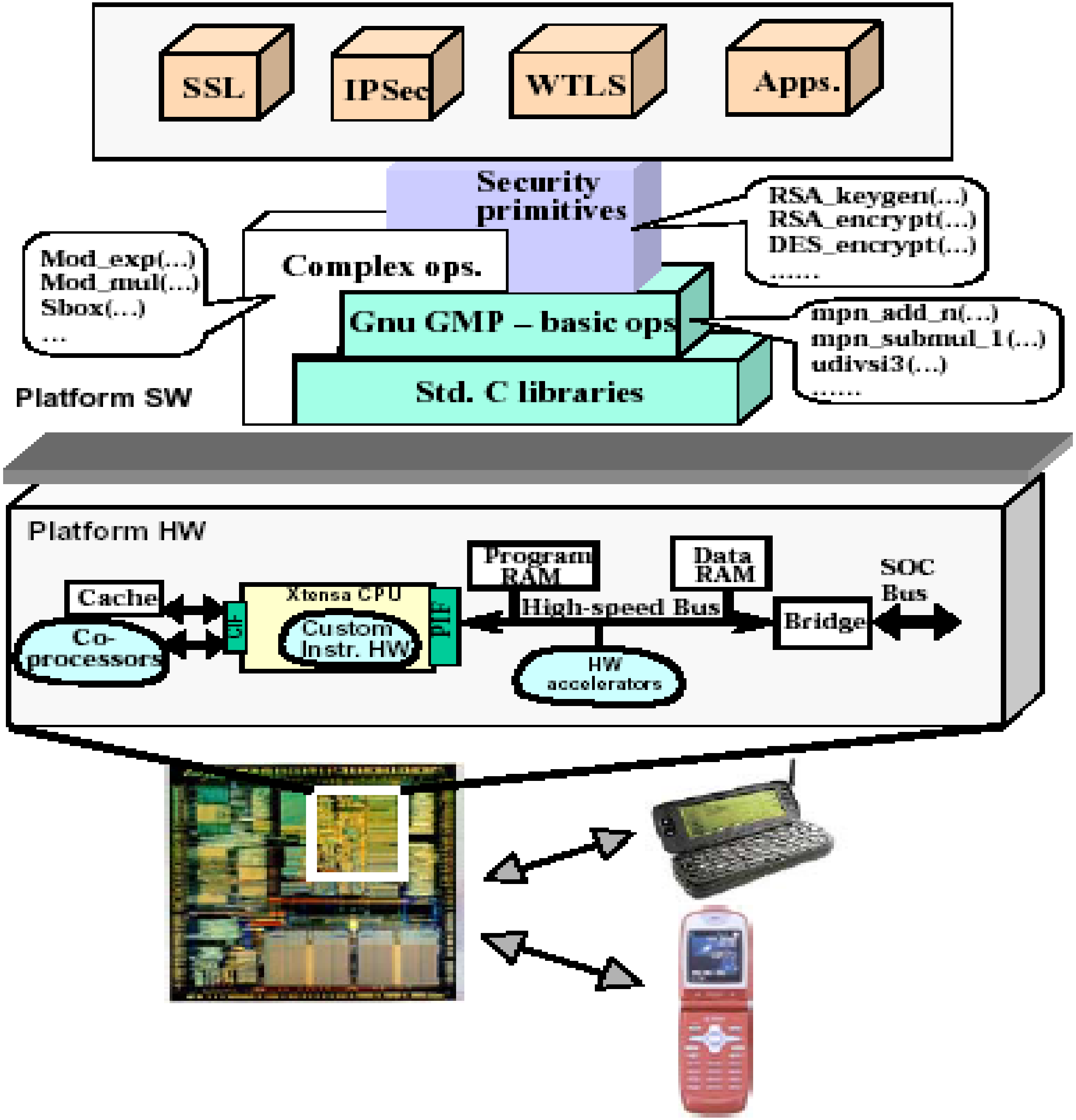
- A programmable security processor platform, to enable secure data and multi-media communications in next-generation wireless handsets.
- Employs a novel system-level design methodology to build the HW/SW platform.
- The objective is to address the wireless security processing gap.

Software Architecture

- Using a layered philosophy, like the design of network protocols.
- At the top level, the SW architecture provides a generic interface using security protocols and applications can be ported to the platform.
- Advantages : each SW layer can proceed concurrently.

Hardware platform architecture

- The instruction set of the processor is extended through the addition of custom instructions that speed up operation.
- The added instructions are executed by custom hardware, which is tightly integrated into the processor execution pipeline.



SSL

IPSec

WTLS

Apps.

Security primitives

RSA_keygen(...)
RSA_encrypt(...)
DES_encrypt(...)
......

Mod_exp(...)
Mod_mul(...)
Sbox(...)
...

Complex ops.

Gnu GMP – basic ops

mpn_add_n(...)
mpn_submul_1(...)
udivsi3(...)
......

Std. C libraries

Platform SW

Platform HW

Cache

Co-processors

CIF

Xtensa CPU

Custom Instr. HW

PIF

Program RAM

Data RAM

High-speed Bus

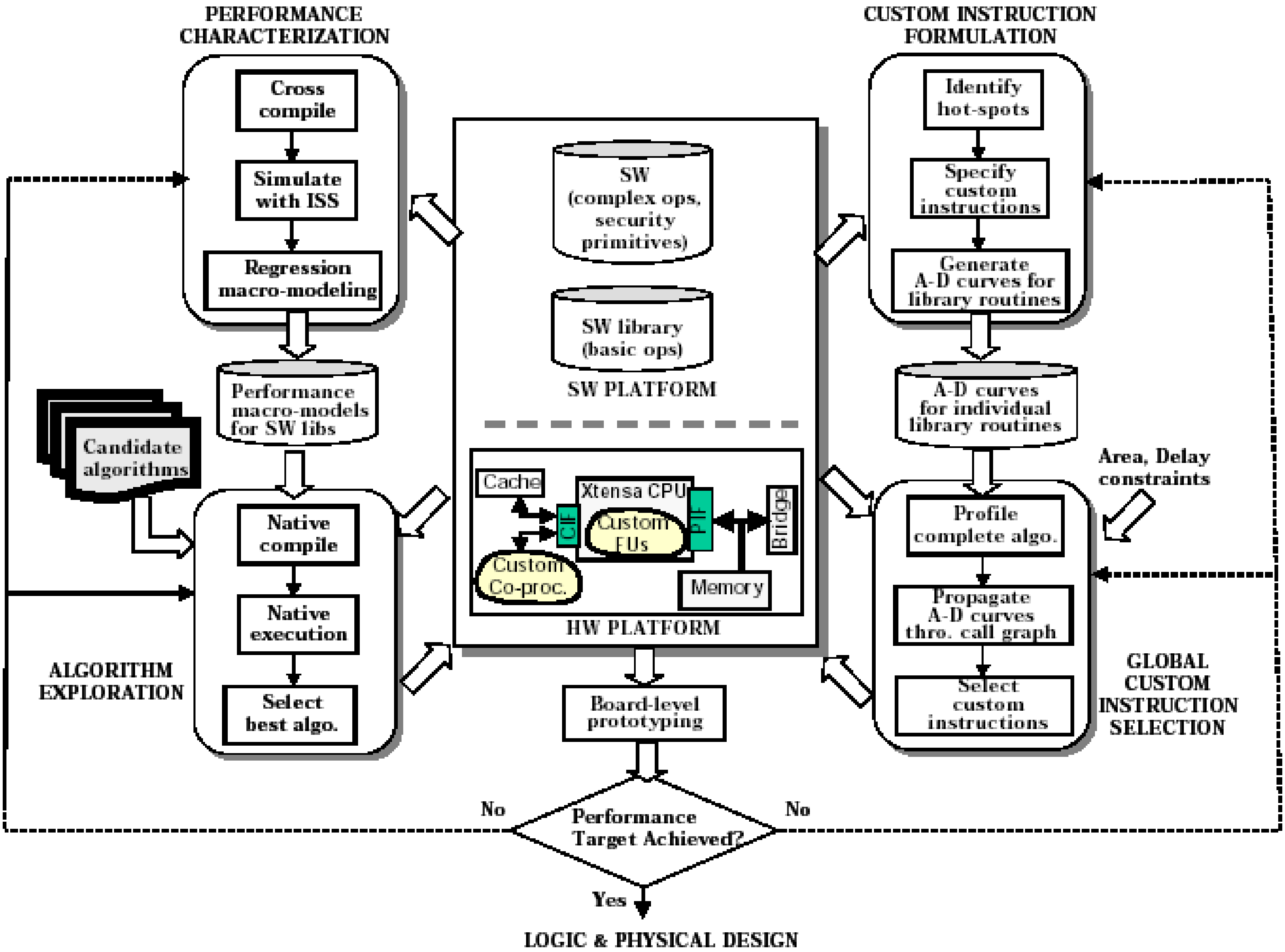
HW accelerators

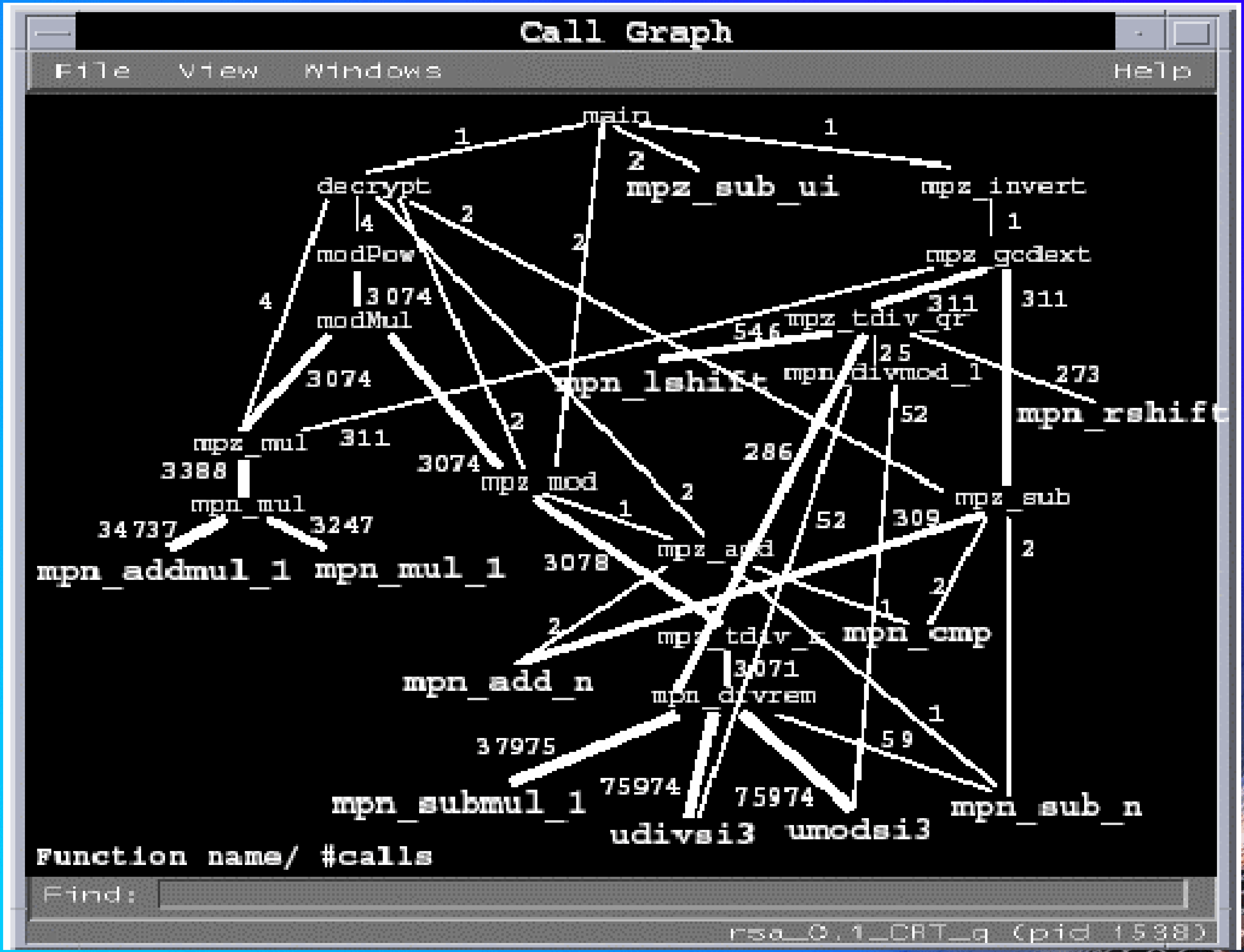
Bridge

SOC Bus

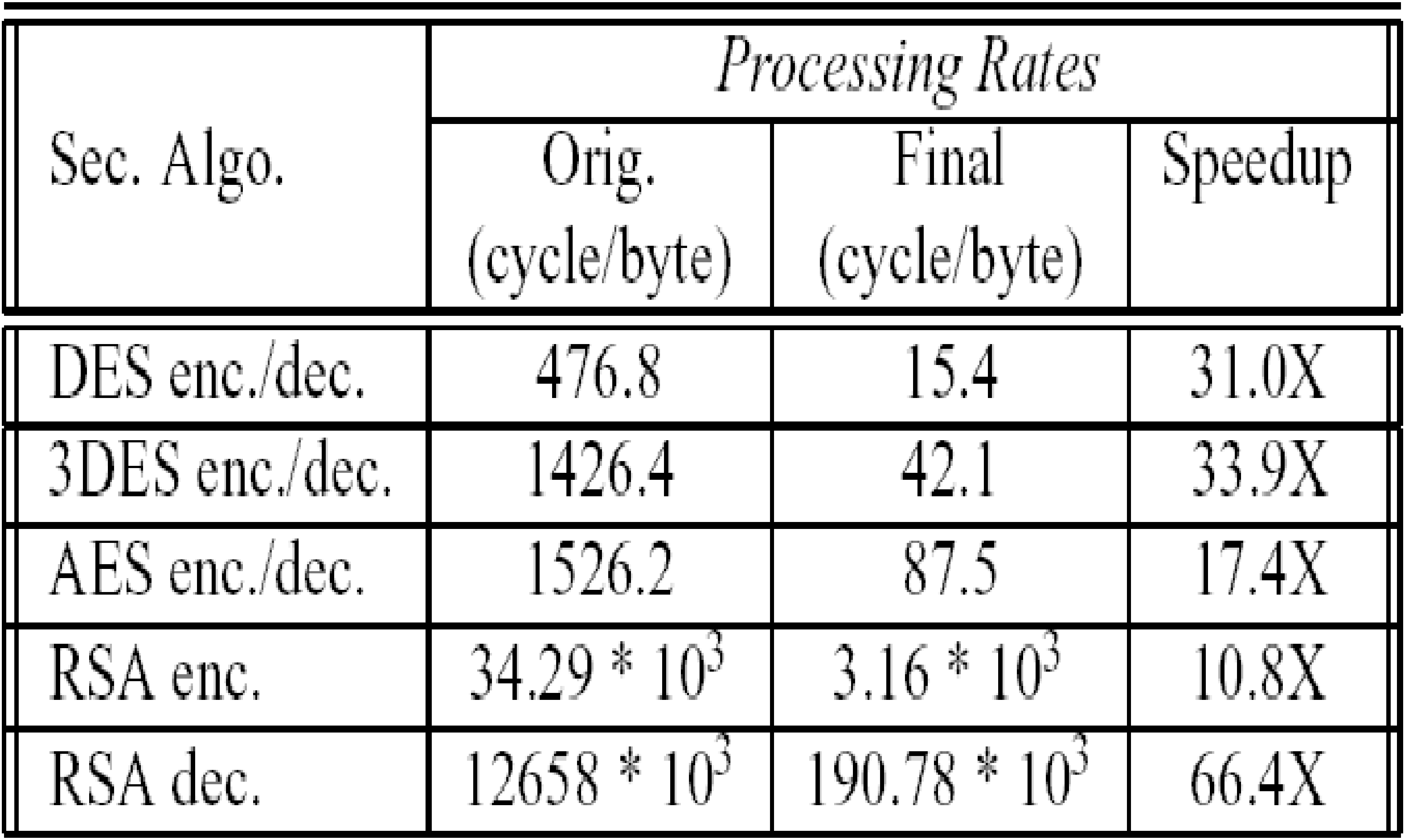
System design methodologies

- Performance characterization of software libraries.
- Algorithm exploration.
- Formulation of candidate custom instructions to accelerate individual library routines.
- Global custom instruction selection to generate the required performance for each security algorithm.

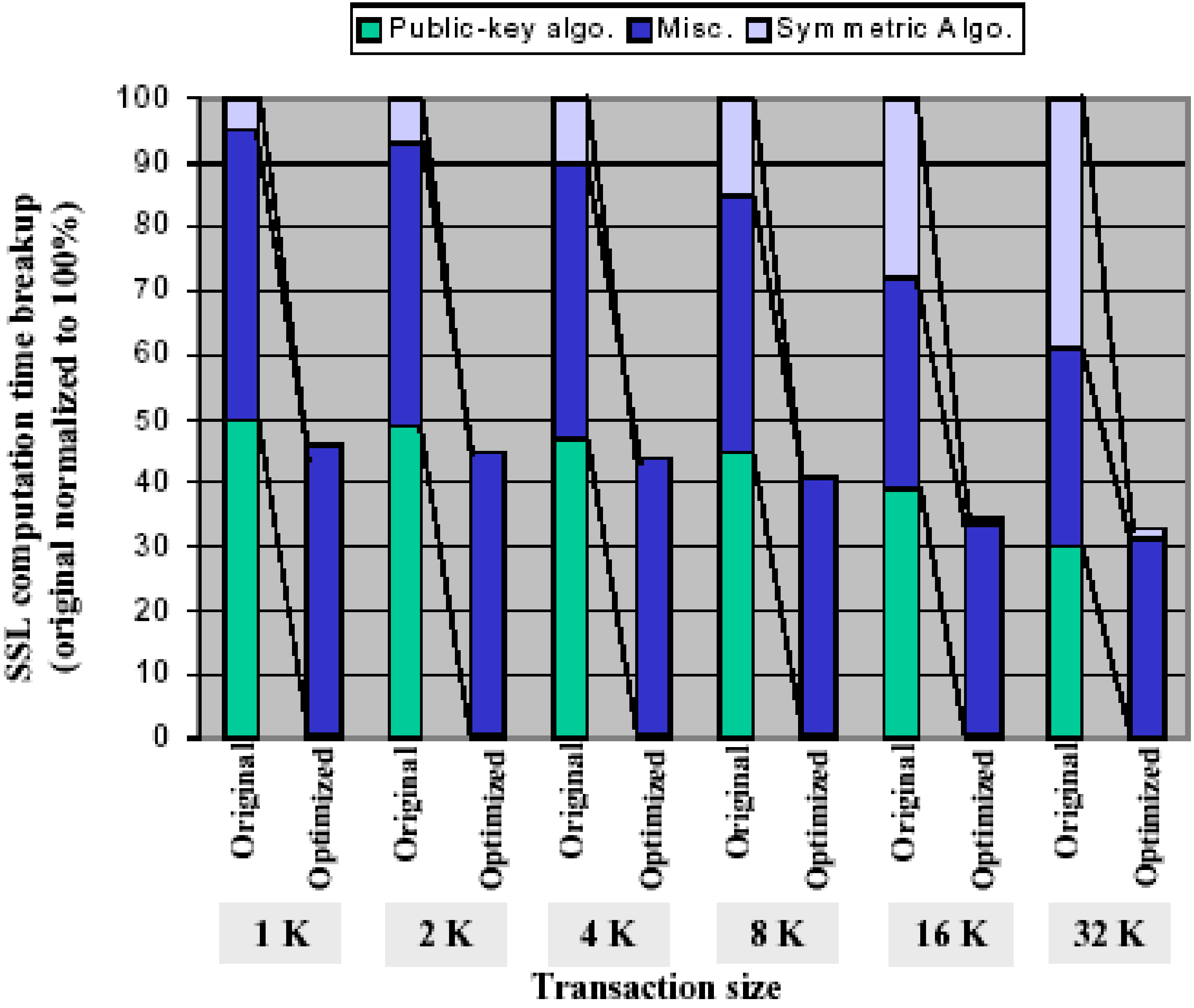

PERFORMANCE CHARACTERIZATION
Cross compile
Simulate with ISS
Regression macro-modeling
Performance macro-models for SW libs
Candidate algorithms
Native compile
Native execution
Select best algo.
ALGORITHM EXPLORATION
SW (complex ops, security primitives)
SW library (basic ops)
SW PLATFORM
Cache
CIF
Xtensa CPU
Custom FUs
PIF
Bridge
Custom Co-proc.
Memory
HW PLATFORM
Board-level prototyping
Performance Target Achieved?
No
No
Yes
LOGIC & PHYSICAL DESIGN
CUSTOM INSTRUCTION FORMULATION
Identify hot-spots
Specify custom instructions
Generate A-D curves for library routines
A-D curves for individual library routines
Area, Delay constraints
Profile complete algo.
Propagate A-D curves thro. call graph
Select custom instructions
GLOBAL CUSTOM INSTRUCTION SELECTION


Call Graph
File
View
Windows
Help
main
1
1
2
decrypt
mpz_sub_ui
mpz_invert
4
2
1
modPow
2
mpz_gcdext
3074
4
modMul
311
311
546
mpz_tdiv_qr
25
3074
mpn_lshift
mpn_divmod_1
273
mpz_mul
311
52
mpn_rshift
2
3388
3074
286
mpz_mod
mpz_sub
mpn_mul
1
2
34737
3247
52
309
mpz_add
2
mpn_addmul_1
mpn_mul_1
3078
2
1
mpn_cmp
2
mpz_tdiv_r
mpn_add_n
3071
mpn_divrem
1
37975
59
75974
75974
mpn_submul_1
mpn_sub_n
udivsi3
umodsi3
Function name/ #calls
Find:
rsa_0.1_CRT_q (pid 1538)

Performance



Sec. Algo.	*Processing Rates*		
	Orig. (cycle/byte)	Final (cycle/byte)	Speedup
DES enc./dec.	476.8	15.4	31.0X
3DES enc./dec.	1426.4	42.1	33.9X
AES enc./dec.	1526.2	87.5	17.4X
RSA enc.	$34.29 * 10^3$	$3.16 * 10^3$	10.8X
RSA dec.	$12658 * 10^3$	$190.78 * 10^3$	66.4X


Public-key algo.
Misc.
Symmetric Algo.
SSL computation time breakup (original normalized to 100%)
100
90
80
70
60
50
40
30
20
10
0
Original
Optimized
1 K
2 K
4 K
8 K
16 K
32 K
Transaction size

Conclusion

- There are several challenges unique to wireless devices and their environment, which need to be addressed.
- A new system architectures and system design methods will be required to address many of these challenges.