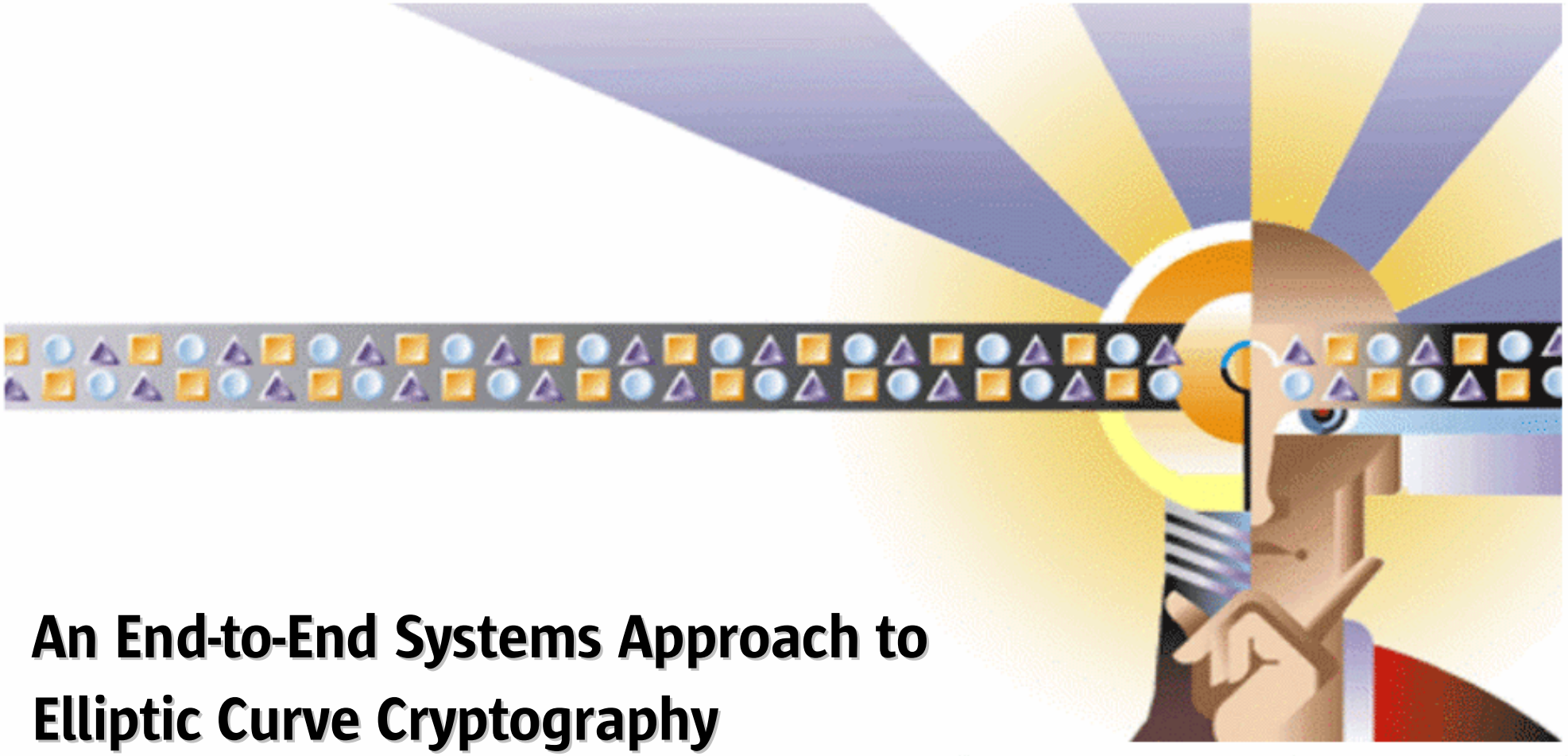


An End-to-End Systems Approach to Elliptic Curve Cryptography

Nils Gura,
Sheueling Chang Shantz, Hans Eberle,
Vipul Gupta

Sun Microsystems Laboratories

Sun microsystems
We make the net work.

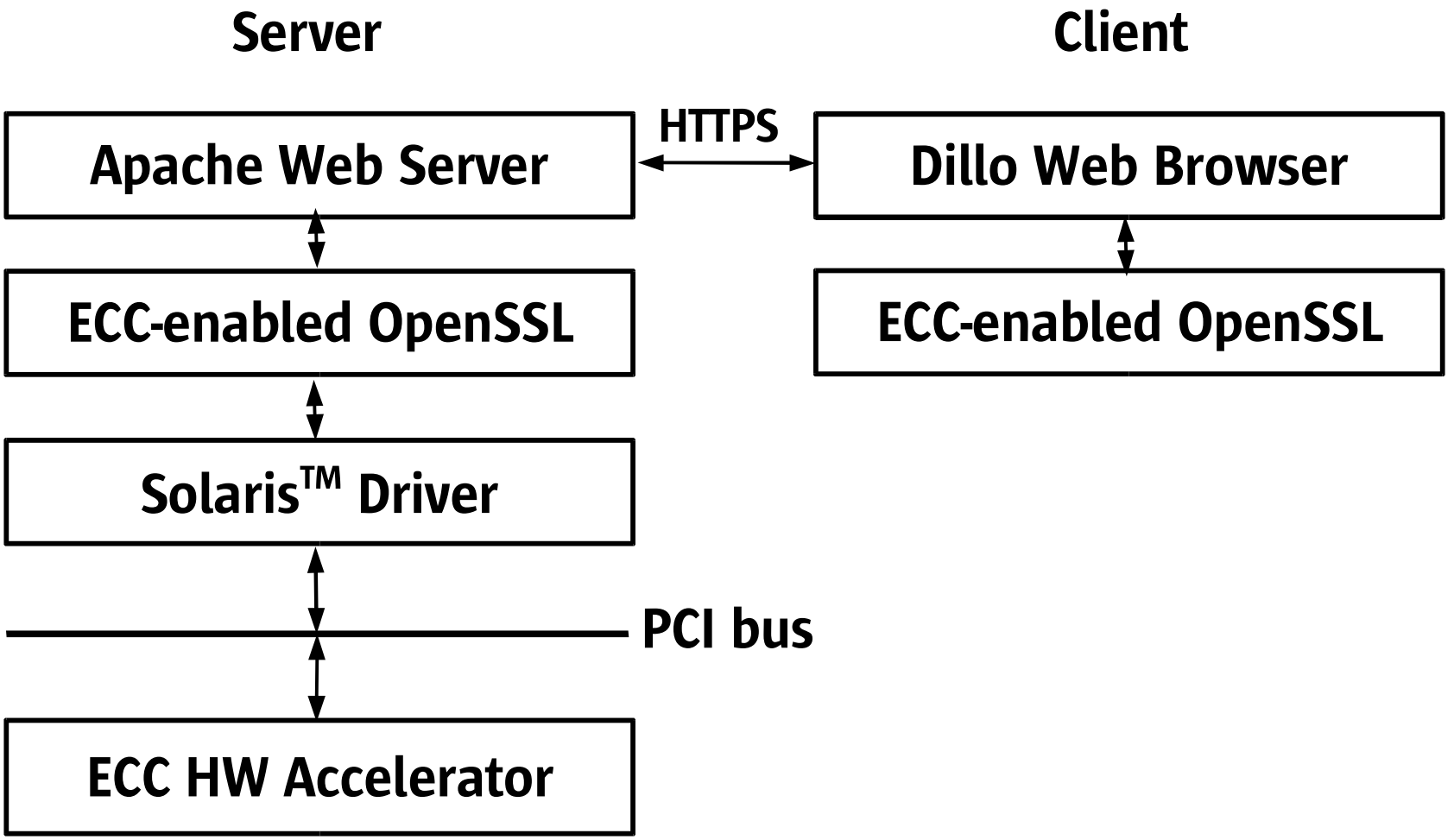
Outline

- ECC for commercial applications
- Integration of ECC into TLS/SSL
- SSL performance with ECC compared to RSA
- Hardware acceleration of ECC
- Conclusions

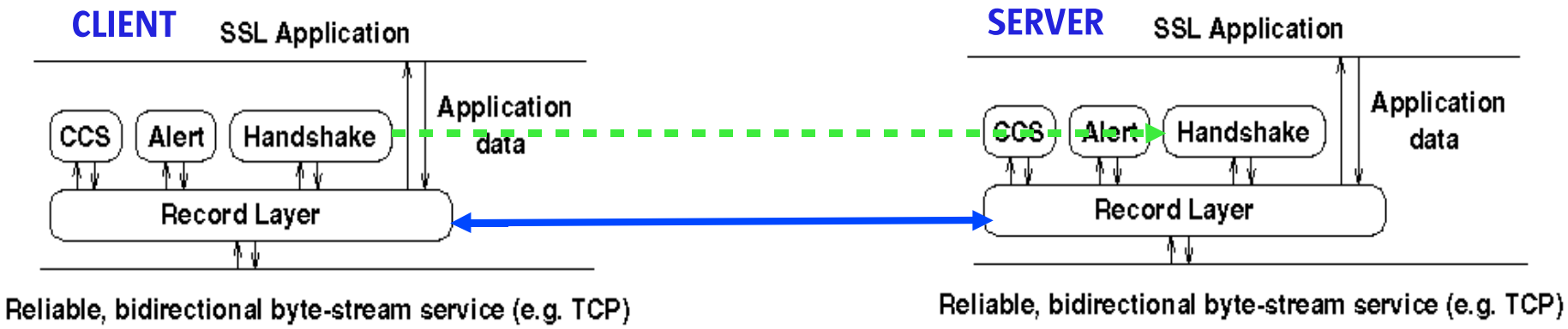
ECC for Commercial Applications

- Confidence in the security of ECC
- Integrated into secure protocols (SSL, IPsec)
- Conformance with standards, e.g. IEEE P1363, ANSI X9.62, X9.63
- Small, cost-efficient, low-power implementations on client devices
- High-performance server-side implementations, >5000 ops/s for ECC-163
- Ability to process a range of curves on the server side

System Overview

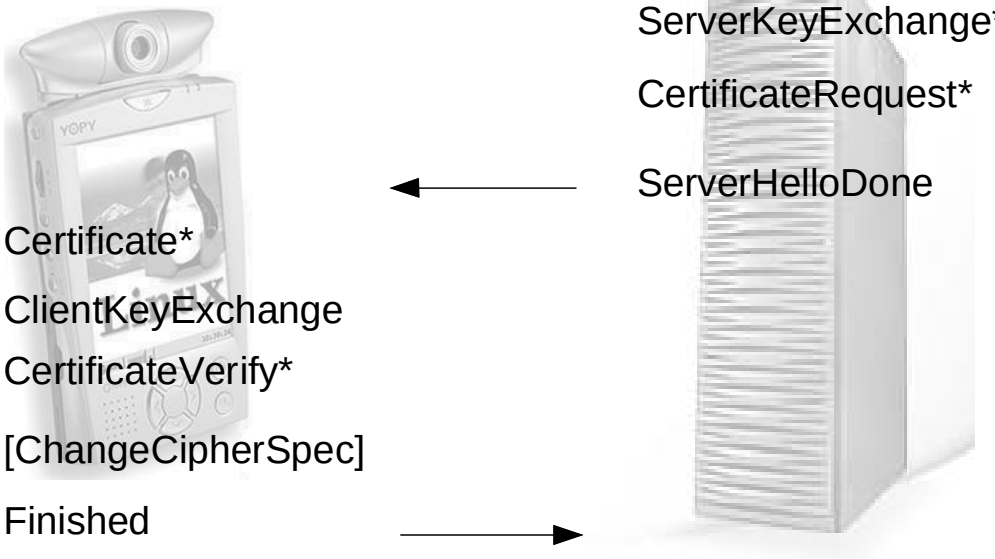


SSL/TLS



- Prevalent protocol securing today's web transactions
- Handshake protocol uses public-key cryptography to establish shared secret
- Record layer uses shared secret to perform MAC and symmetric-key encryption of bulk data
- Client and server negotiate ciphersuite (key exchange, MAC, and symmetric key algorithm), e.g. TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

ECDH-ECDSA Handshake



Client		Server	
ClientHello	→		Initial proposal
		ServerHello	Ciphersuite negotiated
		Certificate*	Server sends ECDH public key k_sP certified with ECDSA signature by certificate authority
		ServerKeyExchange*	
		CertificateRequest*	
	←	ServerHelloDone	
Certificate*			Client verifies server key, computes ECDH secret k_ck_sP, sends its ECDH public key k_cP
ClientKeyExchange			
CertificateVerify*			
[ChangeCipherSpec]			
Finished	→		Server computes ECDH secret k_sk_cP
		[ChangeCipherSpec]	
	←	Finished	Ready for bulk encryption and client authentication
Application Data	↔	Application Data	

SSL Handshake Operations

	RSA	ECDH-ECDSA
Client	RSA_{verify} + $RSA_{encrypt}$	$ECDSA_{verify}$ + $ECDH_{op}$
Server	$RSA_{decrypt}$	$ECDH_{op}$

a. Server authentication only

	RSA	ECDH-ECDSA
Client	RSA_{verify} + $RSA_{encrypt}$ + RSA_{sign}	$ECDSA_{verify}$ + $ECDH_{op}$ or
		$ECDSA_{verify}$ + $ECDSA_{sign}$ + $ECDH_{op}$
Server	2*RSA_{verify} + $RSA_{decrypt}$	$ECDSA_{verify}$ + $ECDH_{op}$ or
		2*$ECDSA_{verify}$ + $ECDH_{op}$

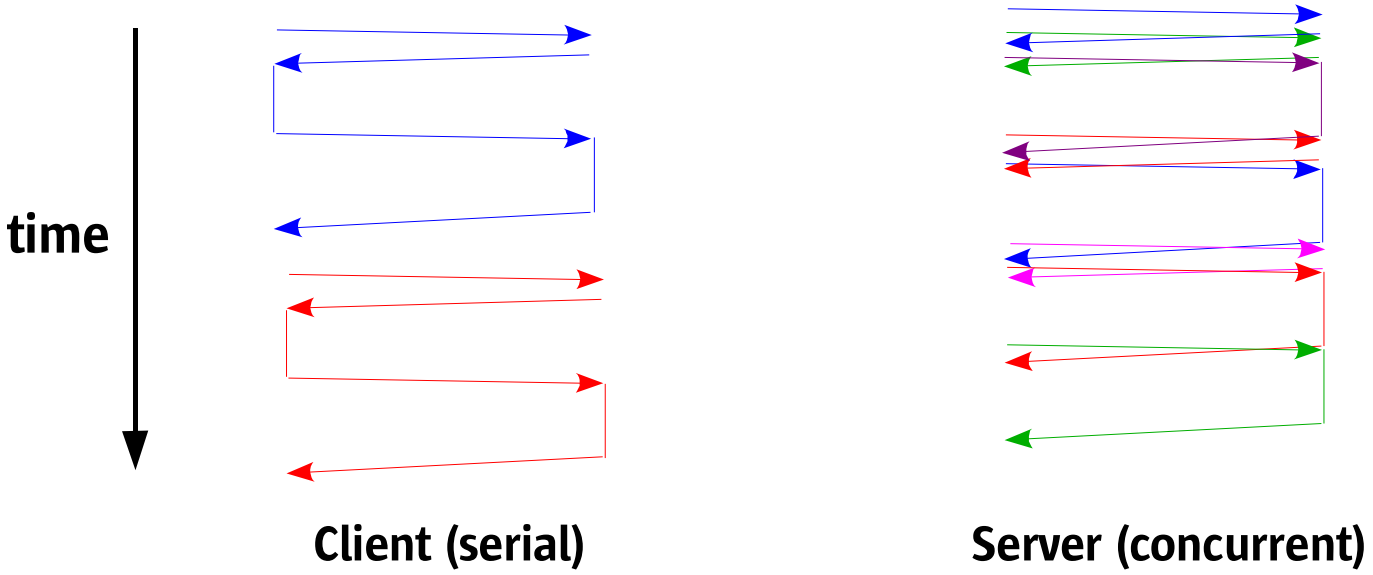
b. Client and server authentication

Measured Crypto Performance

	Key sizes	RSA_{pub}	RSA_{priv}	$ECDSA_{verify}$	$ECDSA_{sign}$	$ECDH_{op}$
Ultra-80	1024/163	1.7	32.1	13	6.8	6.1
Ultra-80	2048/193	6.1	205.5	18.1	9.2	8.7
Yopy	1024/163	10.8	188.7	46.5	24.5	22.9
Yopy	2048/193	39.1	1273.8	76.6	39	37.7

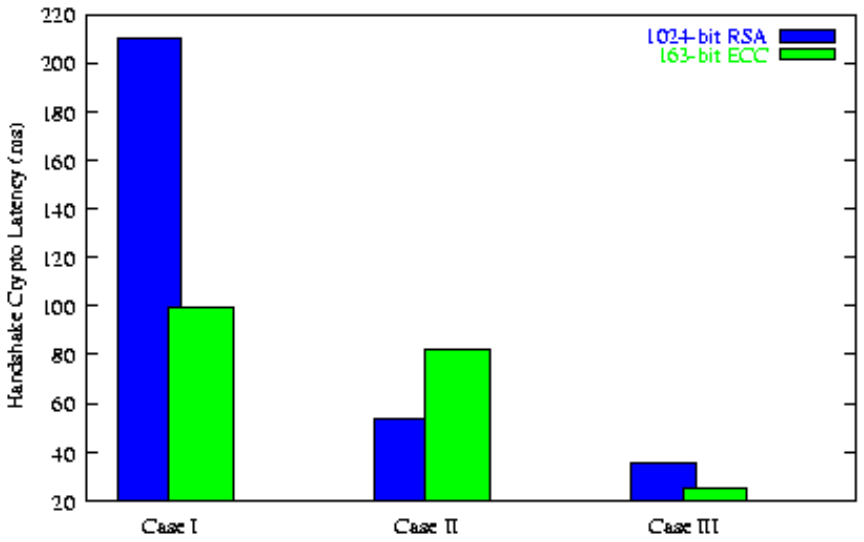
- Yopy: Linux PDA with 200MHz 32-bit StrongARM
- Ultra-80: Sun workstation with 450MHz 64-bit Ultra-SPARC II
- Measured time in ms using OpenSSL “speed” for two different key sizes

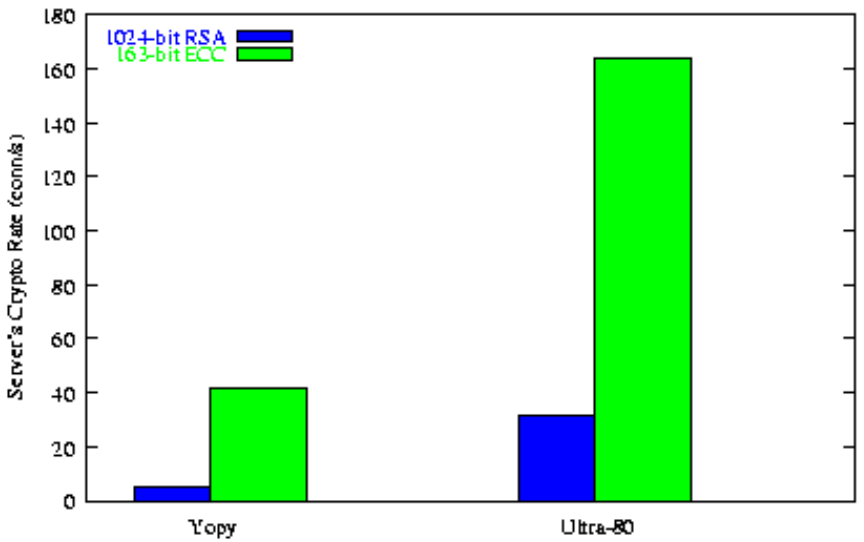
Analyzing SSL-Level Performance



- Different performance criteria on client and server – latency of a connection vs. connection throughput
- First approximations:
  Handshake crypto latency, server's crypto rate

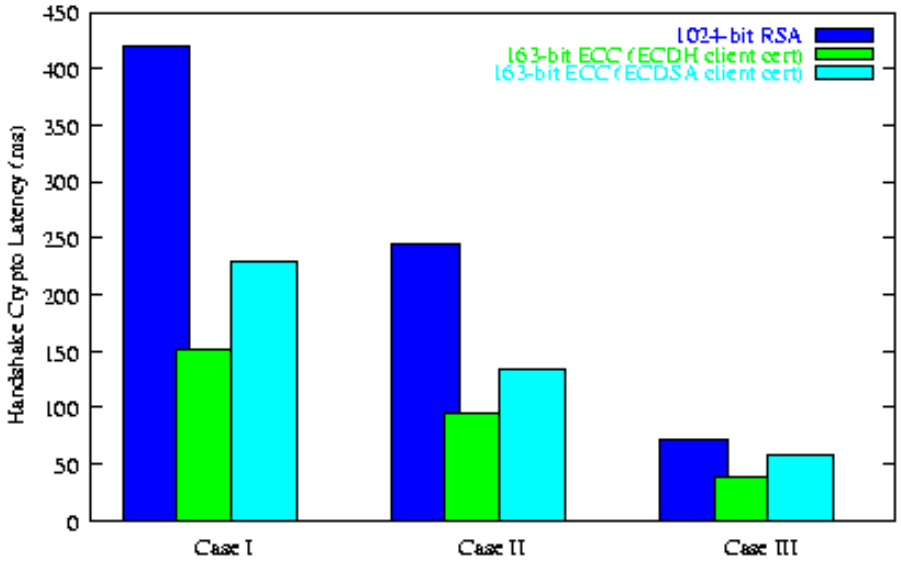
Analytical Performance I
(Without client-side authentication)

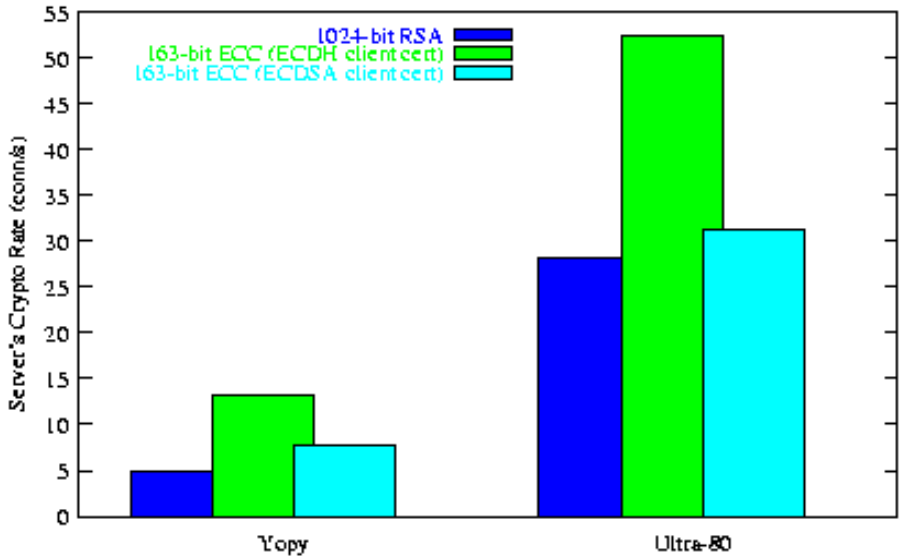




- Three cases: Yopy to Yopy, Yopy to Ultra-80, Ultra-80 to Ultra-80

Analytical Performance II
(With client-side authentication)

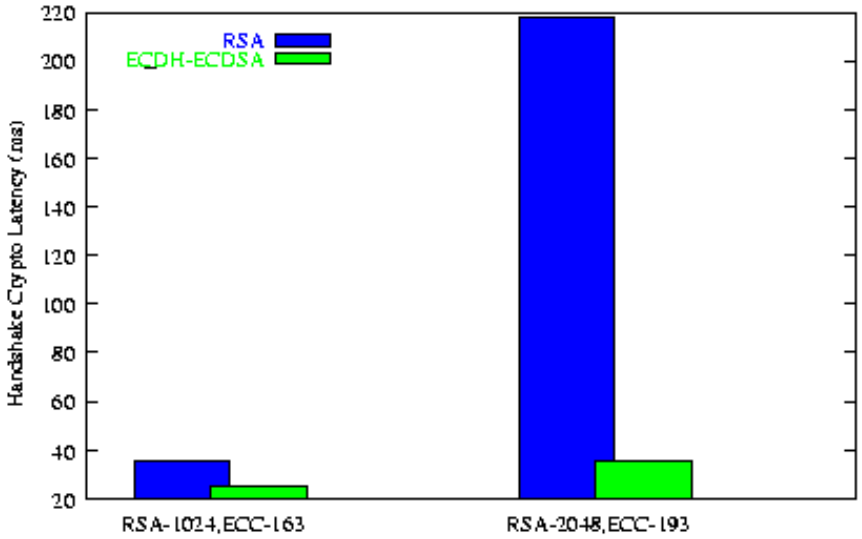


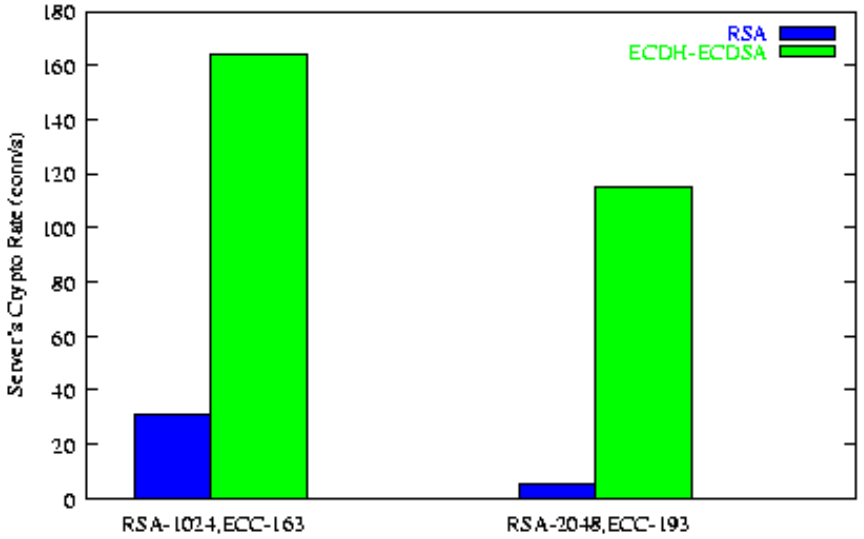


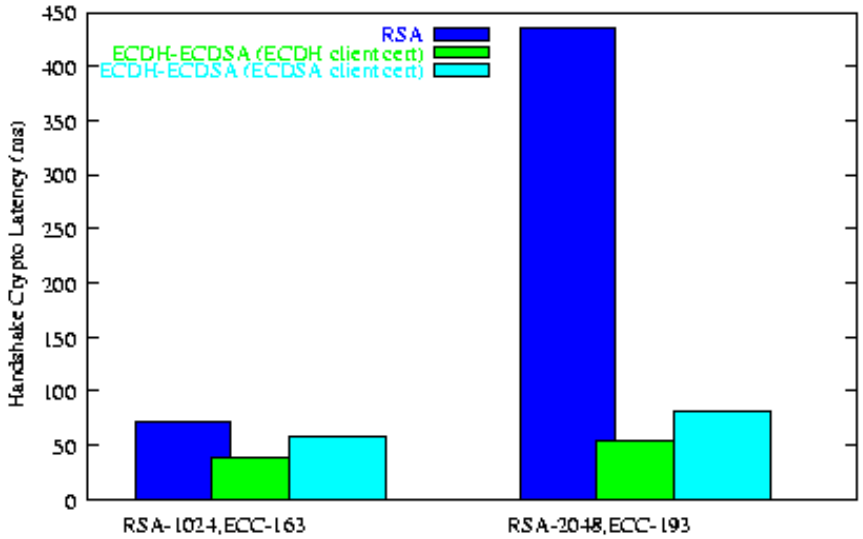
- Same three cases
- Two types of ECC client authentication

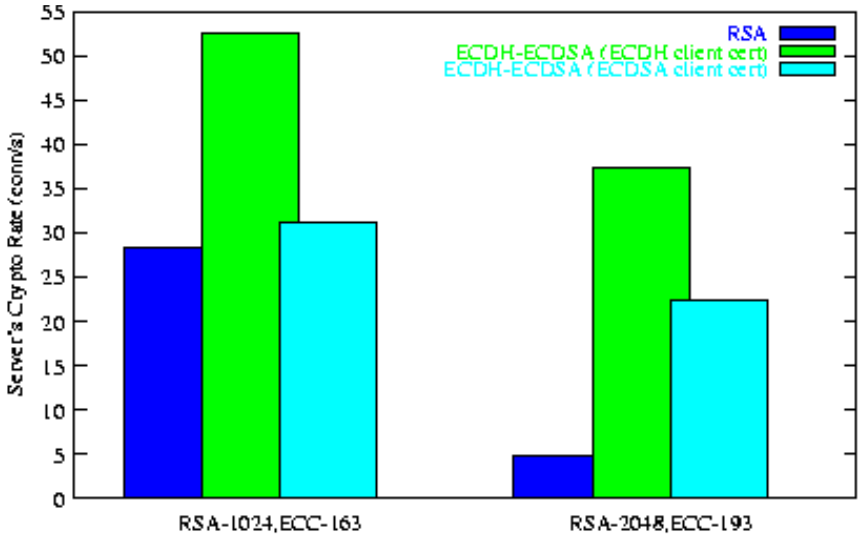
Analytical Performance III

(Impact of key size increase)









Caveats

- Besides public-key cryptographic operations, other CPU operations and network delays impact SSL handshake (actual measurements important)
- Abbreviated handshake does not involve public-key cryptography
- Results change with algorithm code optimizations, processors and hardware acceleration

ECC Hardware Acceleration

- Aggregation of client connections
- Significantly reduces CPU load of secure web servers
- Need to support a variety of curves
- Focused on server-side acceleration of point multiplication over GF(2^m)

Related Work

- Orlando/Paar 2000
  - digit-serial processing
  - highest reported performance
  - designed for specific curves
  - high reconfiguration overhead
- Goodman/Chandrakasan 2001
  - bit-serial processing
  - low power
  - designed for generic curves
  - low reconfiguration overhead

Typical Assumptions

- Squarings are cheap
- Register size == key size
- Reduction can be hardwired
- Different curves can be handled with reconfigurable hardware

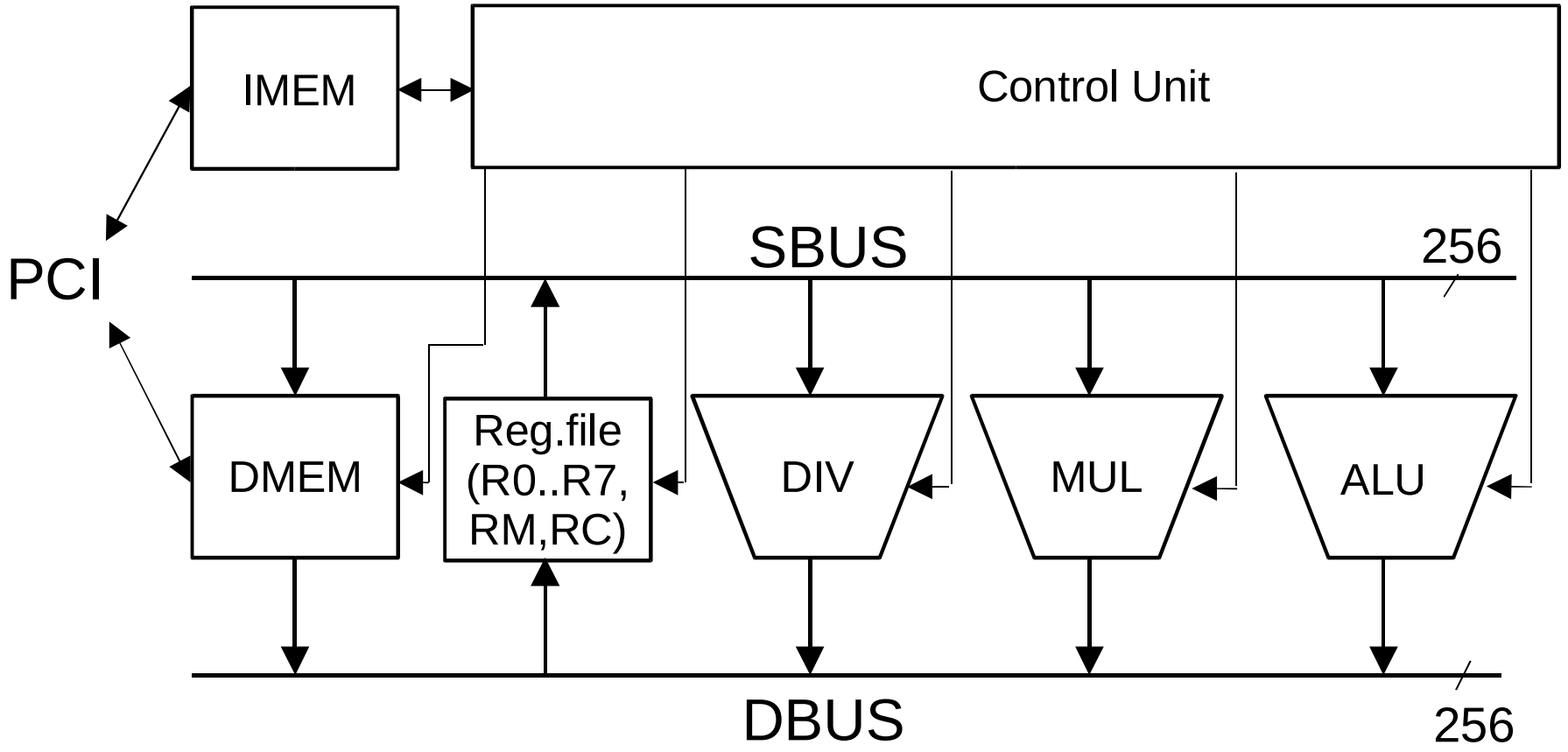
Challenges

- Multiple named curves
  - listed in standards
  - known irreducible polynomials
- Generic curves
  - not known at implementation time
  - infrequently used
- Various key sizes
- High performance
- System integration

Accelerator Characteristics

- Finite field arithmetic for GF(2^m), m≤255
- Arbitrary irreducible polynomials
- Microprogrammable architecture
- Overlapped and parallel instruction execution
- 66 MHz clock
- 66 MHz/64-bit PCI interface

Accelerator Architecture

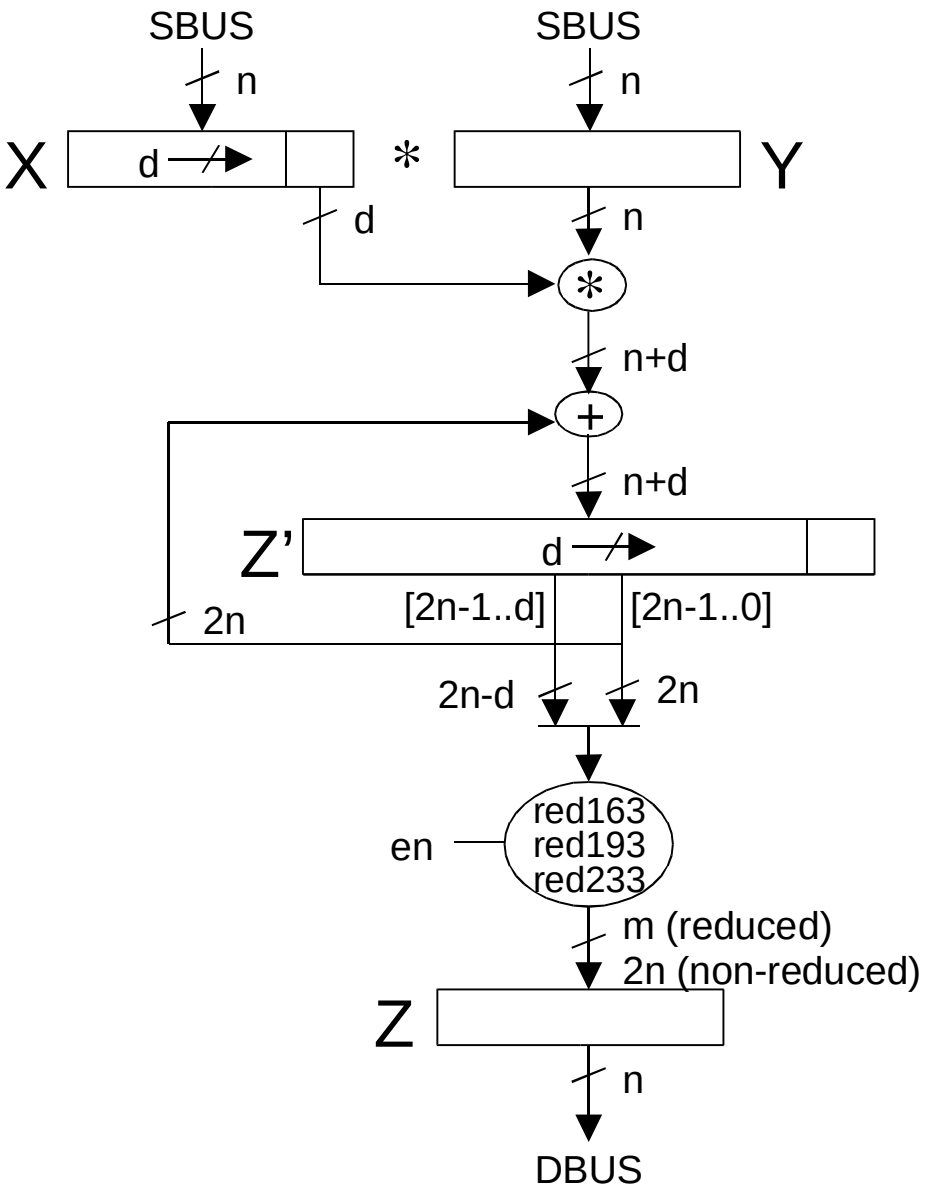


Instruction Set

Instruction		Name	Cycles
Memory Instructions			
LD	DMEM,RD	Load	3
ST	RS, DMEM	Store	3
Arithmetic Instructions			
DIV	RS0,RS1,RD	Divide	≤2m+4
MUL	RS0,RS1,RD	Multiply	8 (7)
MULNR	RS0,RS1,RD	Multiply w/o Reduction	8
ADD	RS0,RS1,RD	Add	3
SQR	RS,RD	Square	3
SL	RS,RD	Shift Left	3
Control Instructions			
BMZ	ADDR	Branch if MSB zero	2
BEQ	ADDR	Branch if equal	4
JMP	ADDR	Jump	2
END		End	

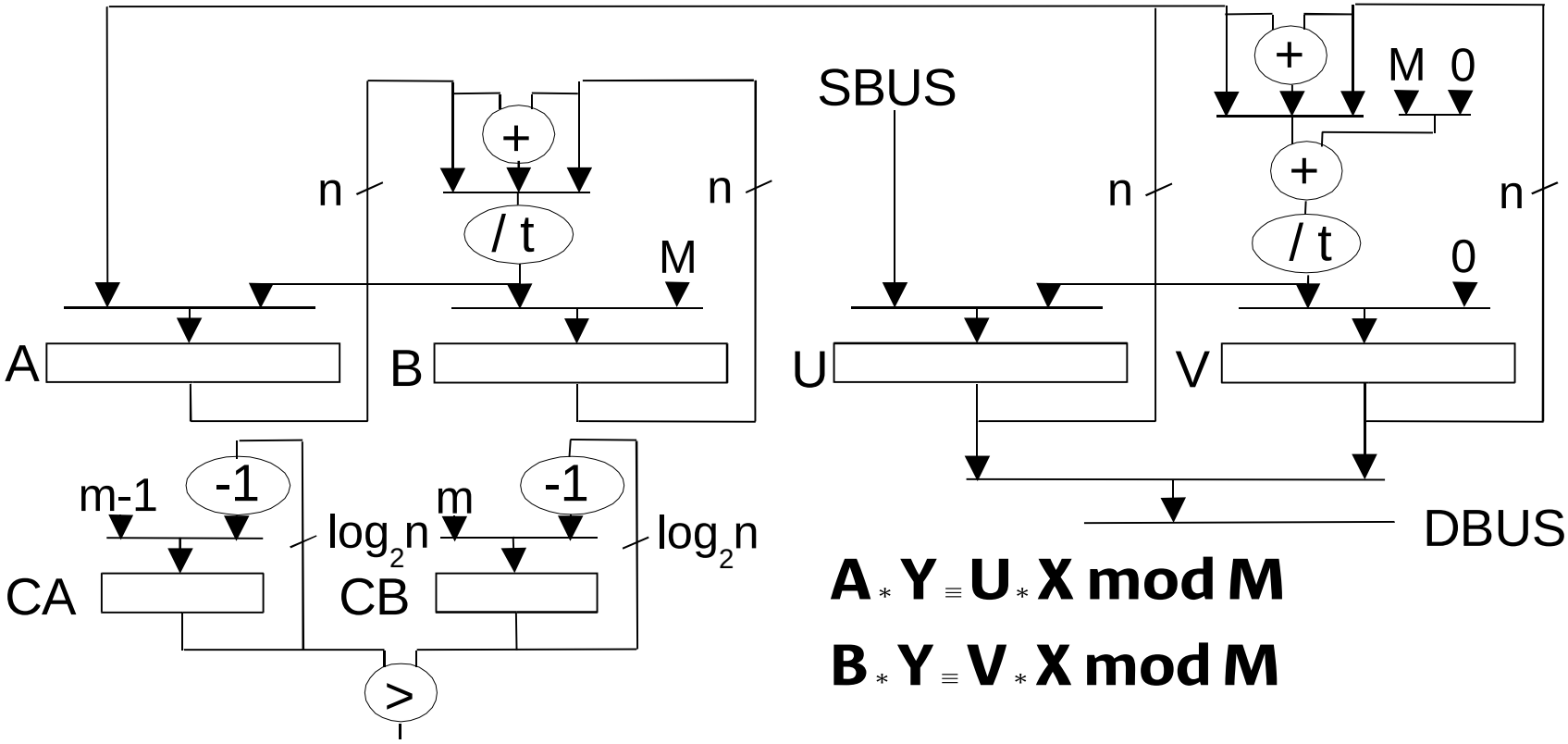
Multiplier

- Register sizes
  X,Y,Z: n=256
  Z': 2n=512
- Digit size d=64
- $\lceil m/d \rceil + 1$ cycles
- Hardwired reduction for $GF(2^{163})$, $GF(2^{193})$, $GF(2^{233})$



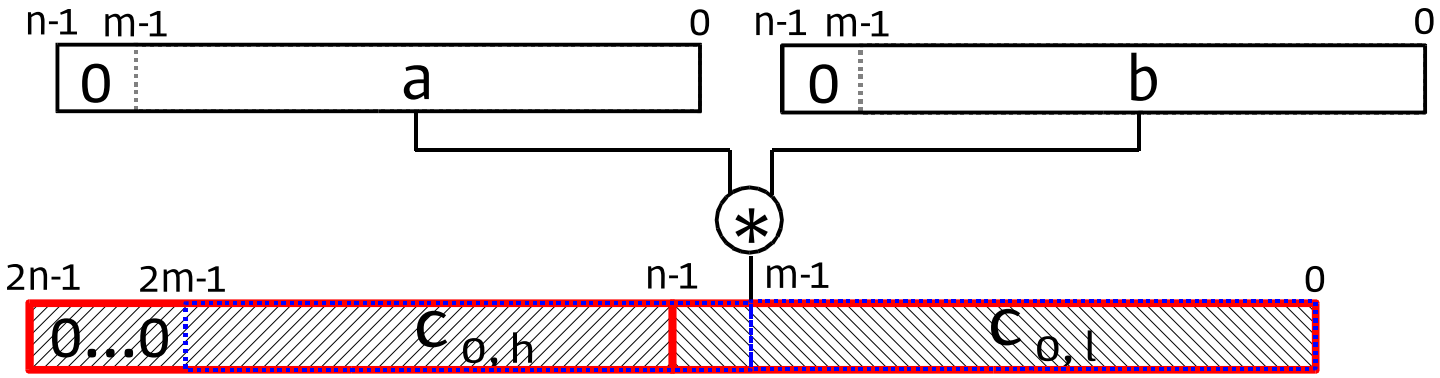
Divider

- Computes Y/X mod M for arbitrary irreducible polynomials M
- Faster than soft-coded inversion algorithms



Generic Curves

- Register size > key size

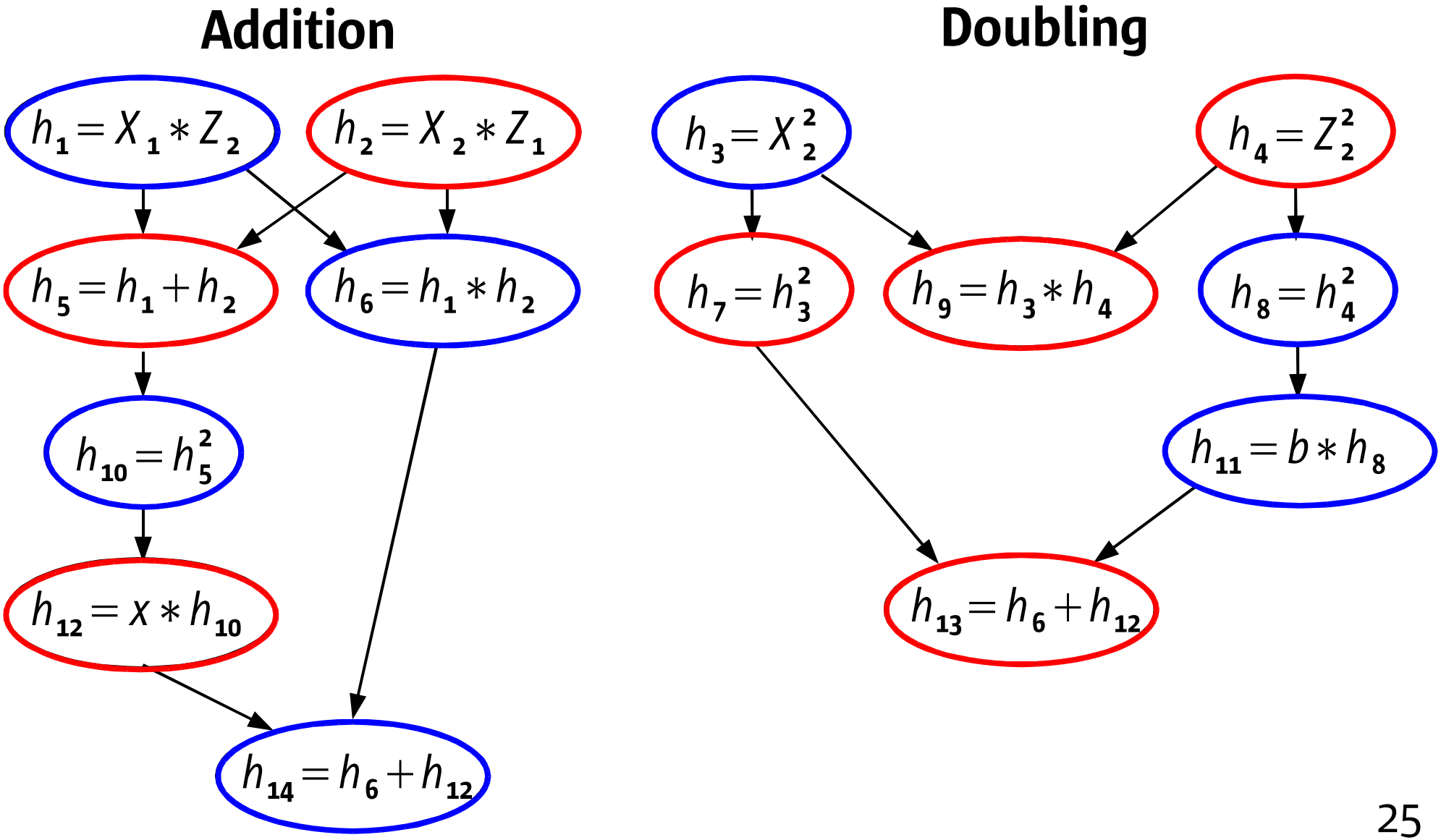


- Single multiplication requires 3 MULNR and 1 ADD instruction including reduction
- Squarings as expensive as multiplications
- Soft-coded inversion algorithms become expensive

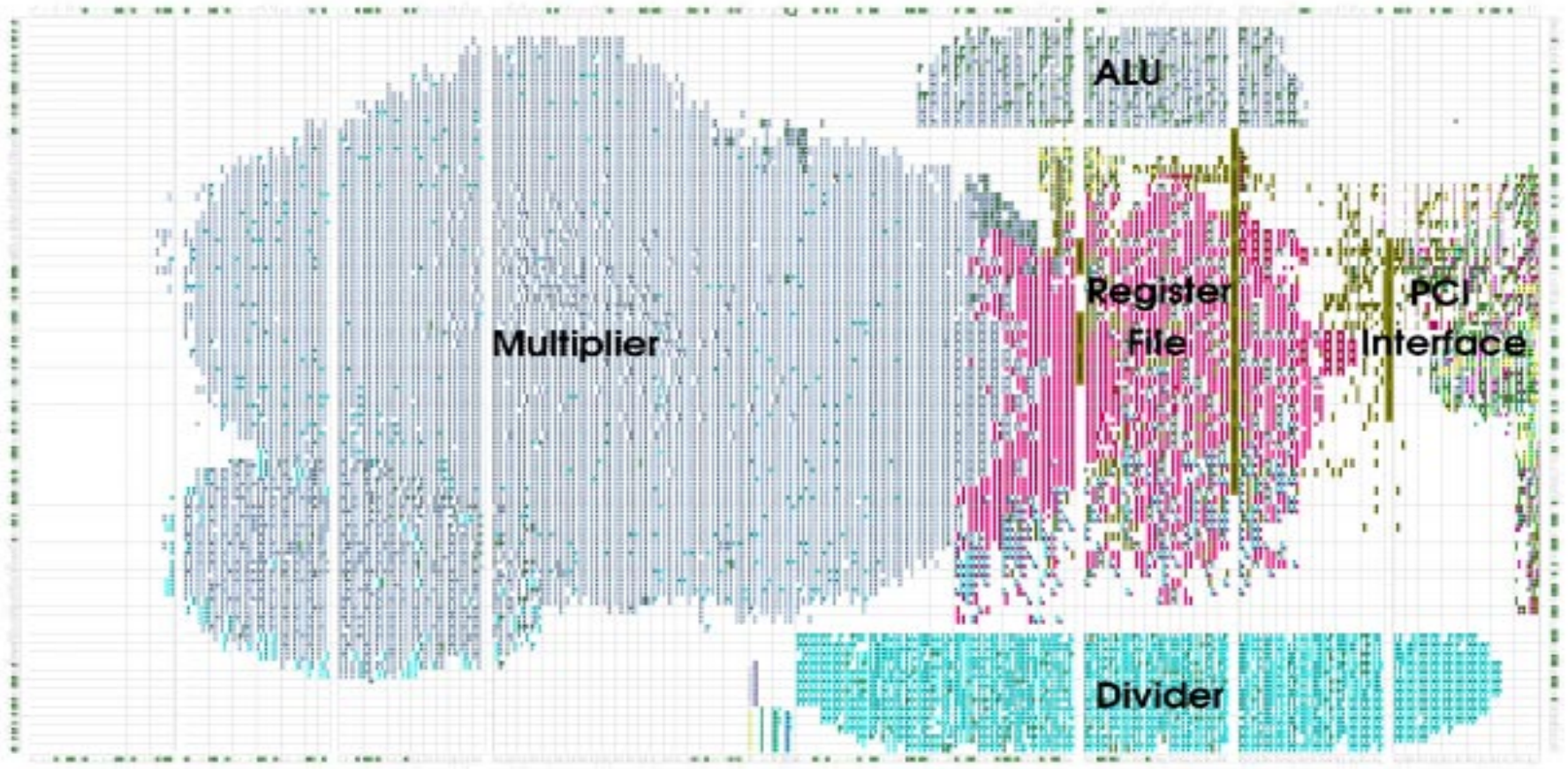
Montgomery Point Multiplication

- Projective coordinates:
  ~ 6m multiplications ~ 5m squarings
  ~ 3m additions 1 division
- Simple control flow for hardware implementations
- Low memory requirements
- Paired point additions and doublings
- Provides protection against timing and power analysis attacks
- Concurrent operations

Montgomery Point Multiplication



Accelerator Floorplan



Technology: Xilinx XCV2000E FPGA
Size: 20068 LUTs, 6321 FFs
Clock: 66 MHz

Performance

ops/s	Hardware	Software	Speedup
Named Curves			
GF(2^{163})	6987	322	21.7
GF(2^{233})	4438	223	19.9
Generic Curves			
GF(2^{163})	1075	322	3.3
GF(2^{233})	757	223	3.4
ECDH			
GF(2^{163})	3813	304	12.5
ECDSA (sign)			
GF(2^{163})	1576	292	5.4
ECDSA (verify)			
GF(2^{163})	1224	151	8.1

Conclusions

- Built end-to-end secure system based on ECC-enabled OpenSSL
- Performance evaluation and optimization on the system level
- Hardware acceleration for named and generic curves without reconfiguration
- Reduction is the costliest operation for generic curves
- High mul/div ratio favors projective coordinate representation

Web Resources

- Sun Labs Website
  - http://www.research.sun.com
  - http://www.experimentalstuff.com
- Next Generation Crypto Project
  - http://research.sun.com/projects/crypto
- OpenSSL
  - http://www.openssl.org