Workload Characterization of Elliptic Curve Cryptography and other Network Security Algorithms for Constrained Environments

#### A. Murat Fiskiran and Ruby B. Lee

#### Princeton Architecture Laboratory for Multimedia and Security

Princeton University





# Need for Security?

- Wireless devices: PDAs, multimedia cell phones, tablet PCs ...
  - Public channel = need for cryptography
  - Limited processing power, memory, power

Class	Function	
Public-key	Key exchange, user authentication, digital signature	
Symmetric-key	Confidentiality	
Hash	Integrity	

# Algorithm Set

Class	Typical key size	Examples	
Dublic kov	1024 – 2048 bits (non-elliptic curve)	Diffie-Hellman,	
Рибііс-кеу	163 – 233 bits (elliptic curve)	ElGamal, DSA	
Symmetric-key	128 – 256 bits	DES, AES	
Hash N/A		SHA, MD5	

• Diffie-Hellman is representative of other elliptic-curve algorithms.

#### Diffie-Hellman on Elliptic Curves

• E is an elliptic curve, P = (x,y) is a point on E.



Based on elliptic-curve discrete logarithm problem

## **Point Multiplication**

- P × a is point multiplication. The result is **another point** on the elliptic curve.
- Computed by a double-and-add chain
  - No easy way to compute any arbitrary multiple of P.
- Example: if a = 13, then:

 $P \times 13 = [(2 \times P + P) \times 2 \times 2] + P$ 

## Point Doubling and Addition

P = (x, y)	$P = (x_1, y_1), Q = (x_2, y_2)$	
$P \times 2 = (x_f, y_f)$	$P + Q = (x_3, y_3)$	
$\boldsymbol{q} = x + \frac{y}{x}$ $x_f = \boldsymbol{q}^2 + \boldsymbol{q} + a$ $y_f = x^2 + (\boldsymbol{q} + 1)x_f$	$\boldsymbol{q} = \frac{y_2 + y_1}{x_2 + x_1}$ $x_3 = \boldsymbol{q}^2 + \boldsymbol{q} + x_1 + x_2 + a$ $y_3 = \boldsymbol{q}(x_1 + x_3) + x_3 + y_1$	
4 addition, 2 multiplication, 2 squaring, 1 inversion	9 addition, 2 multiplication, 1 squaring, 1 inversion	

# **Binary Fields**

- Coordinates of P=(x,y) come from a field.
- Fastest implementations are on binary fields.

– Field elements = binary polynomials

• Example:

$$P = (x+1, x^2+1) = (0011, 0101)_2$$

### ECC and Polynomial Operations

Point doubling	Point addition	
4 addition, 2 multiplication,	9 addition, 2 multiplication,	
2 squaring, 1 inversion	1 squaring, 1 inversion	

Addition	Multiplication	Squaring	Inversion
XOR	Shift-and-add Table-lookup Polynomial multiplier	Self-multiplication Table-lookup New instructions	Extended Euclidean Almost Inverse

Orange: basic

Green: optimized

# Methodology

- Algorithms coded and optimized in assembly
- 64-bit basic RISC architecture
- Simulated using two algorithm sets: basic and optimized
- 163-bit and 233-bit keys
- Diffie-Hellman, ElGamal, DSA
- AES and SHA (for completeness)

#### Speedup From Optimized Algorithms



## Instruction Distribution



- 1a. EC-DHKE basic (1.0)
- 2a. EC-ElGamal basic (1.0)
- 3a. EC-DSA basic (1.0)

- 1b. EC-DHKE optimized (14.5)
- 2b. EC-ElGamal optimized (16.9)
- 3b. EC-DSA optimized (15.3)

### Pathlength Increase: From 163-bit to 233-bit keys



## Observations

- 1. Algorithmic enhancements provide 15× speedup.
  - Mainly by reducing arithmetic operations (up to  $30 \times$ )
- 2. Memory instructions are as frequent as compute instructions.
  - <u>Reasons</u>: Function call overhead, long data types
  - Further speedups possible based on memory optimizations (?)
- 3. Longer keys result in disproportionately large slowdowns.
  Complexity of ECC operations
- 4. DH and ElGamal have similar distributions.
  - DSA is different; includes SHA as hash algorithm.
- 5. Optimized algorithms use little extra memory (<1kB).
- 6. A separate multiplier is not needed.

# Summary

- 1. Selection of algorithms suitable for constrained environments:
  - Elliptic-curve versions of DH, ElGamal, and DSA; AES and SHA
- 2. Description of operations needed; focus on elliptic-curve and polynomial operations
- 3. Instruction frequencies
- 4. Sufficiency of a simple RISC processor

## Future Work

- Expand algorithm set
  - Include block ciphers, other signature and hash algorithms
- Expand arithmetic operations to
  - Integers (prime fields)
  - Different representation of polynomials (different bases)
  - Different coordinate systems (e.g. projective)