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

<table>
<thead>
<tr>
<th>Class</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public-key</strong></td>
<td>Key exchange, user authentication, digital signature</td>
</tr>
<tr>
<td><strong>Symmetric-key</strong></td>
<td>Confidentiality</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Integrity</td>
</tr>
</tbody>
</table>
Algorithm Set

<table>
<thead>
<tr>
<th>Class</th>
<th>Typical key size</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public-key</td>
<td>1024 - 2048 bits (non-elliptic curve)</td>
<td><strong>Diffie-Hellman</strong>, ElGamal, DSA</td>
</tr>
<tr>
<td></td>
<td><strong>163 - 233 bits (elliptic curve)</strong></td>
<td></td>
</tr>
<tr>
<td>Symmetric-key</td>
<td>128 - 256 bits</td>
<td>DES, AES</td>
</tr>
<tr>
<td>Hash</td>
<td>N/A</td>
<td>SHA, MD5</td>
</tr>
</tbody>
</table>

- **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 \( E \).

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose ( a ).</td>
<td>Choose ( b ).</td>
</tr>
<tr>
<td>( P \times a )</td>
<td>( P \times b )</td>
</tr>
<tr>
<td>( (P \times b) \times a )</td>
<td>( (P \times a) \times b )</td>
</tr>
</tbody>
</table>

Based on elliptic-curve discrete logarithm problem
Point Multiplication

• $P \times a$ 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

<table>
<thead>
<tr>
<th>P = (x, y)</th>
<th>P = (x₁, y₁), Q = (x₂, y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P × 2 = (xᵢ, yᵢ)</td>
<td>P + Q = (x₃, y₃)</td>
</tr>
<tr>
<td>[ \theta = x + \frac{y}{x} ]</td>
<td>[ \theta = \frac{y₂ + y₁}{x₂ + x₁} ]</td>
</tr>
<tr>
<td>[ xᵢ = \theta^2 + \theta + a ]</td>
<td>[ x₃ = \theta^2 + \theta + x₁ + x₂ + a ]</td>
</tr>
<tr>
<td>[ yᵢ = x^2 + (\theta + 1)xᵢ ]</td>
<td>[ y₃ = \theta(x₁ + x₃) + x₃ + y₁ ]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Point doubling</th>
<th>Point addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 addition, 2 multiplication, 2 squaring, 1 inversion</td>
<td>9 addition, 2 multiplication, 1 squaring, 1 inversion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addition</th>
<th>Multiplication</th>
<th>Squaring</th>
<th>Inversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>Shift-and-add</td>
<td>Self-multiplication</td>
<td>Extended Euclidean</td>
</tr>
<tr>
<td></td>
<td>Table-lookup</td>
<td>Table-lookup</td>
<td>Almost Inverse</td>
</tr>
<tr>
<td></td>
<td>Polynomial multiplier</td>
<td>New instructions</td>
<td></td>
</tr>
</tbody>
</table>

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

- DH-163: 14.5
- ElGamal-163: 16.9
- DSA-163: 15.3
- DH-233: 12.9
- ElGamal-233: 16.3
- DSA-233: 17.9
1a. EC-DHKE basic (1.0)  
1b. EC-DHKE optimized (14.5)  
2a. EC-ElGamal basic (1.0)  
2b. EC-ElGamal optimized (16.9)  
3a. EC-DSA basic (1.0)  
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×)

2. **Memory** instructions are as frequent as **compute** instructions.
   - **Reasons**: 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)