# 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, <br> digital signature |
| Symmetric-key | Confidentiality |
| Hash | Integrity |

## Algorithm Set

| Class | Typical key size | Examples |
| :---: | :---: | :---: |
| Public-key | $1024-2048$ bits <br> (non-elliptic curve) | }{} |
|  | $163-233$ bits <br> (elliptic curve) |  |
|  |  |  |$|$|  |  |  |
| :---: | :---: | :---: |
| 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 $E$.

| Alice |  | Bob |
| :---: | :---: | :---: |
| Choose a. |  | Choose b. |
| $\mathrm{P} \times \mathrm{a}$ | $\mathrm{P} \times \mathrm{b}$ |  |
| $(\mathrm{P} \times \mathrm{b}) \times \mathrm{a} \longrightarrow$ | $(\mathrm{P} \times \mathrm{a}) \times \mathrm{b}$ |  |

Based on elliptic-curve discrete logarithm problem

## Point Multiplication

- $\mathrm{P} \times \mathrm{a}$ 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

$$
\begin{array}{c|c}
\mathrm{P}=(x, y) & \mathrm{P}=\left(x_{1}, y_{1}\right), \mathrm{Q}=\left(x_{2}, y_{2}\right) \\
\mathrm{P} \times 2=\left(x_{f}, y_{f}\right) & \mathrm{P}+\mathrm{Q}=\left(x_{3}, y_{3}\right) \\
\hline \theta=x+\frac{y}{x} & \theta=\frac{y_{2}+y_{1}}{x_{2}+x_{1}} \\
x_{f}=\theta^{2}+\theta+a & x_{3}=\theta^{2}+\theta+x_{1}+x_{2}+a \\
y_{f}=x^{2}+(\theta+1) x_{f} & y_{3}=\theta\left(x_{1}+x_{3}\right)+x_{3}+y_{1}
\end{array}
$$

4 addition, 2 multiplication, 2 squaring, 1 inversion

9 addition, 2 multiplication, 1 squaring, 1 inversion

## Binary Fields

- Coordinates of $\mathrm{P}=(\mathrm{x}, \mathrm{y})$ come from a field.
- Fastest implementations are on binary fields.
- Field elements = binary polynomials
- Example:

$$
\mathrm{P}=\left(x+1, x^{2}+1\right)=(0011,0101)_{2}
$$

## ECC and Polynomial Operations

## Point doubling <br> Point addition

4 addition, 2 multiplication, 2 squaring, 1 inversion

9 addition, 2 multiplication, 1 squaring, 1 inversion

| Addition | Multiplication | Squaring | I nversion |
| :---: | :---: | :---: | :---: |
| XOR | Shift-and-addl | Self-multiplication |  |
|  | Table-lookup | Table-lookup | Extended Euclidean |
|  | Polynomial multiplier | New instructions |  |

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 $15 \times$ speedup.

- Mainly by reducing arithmetic operations (up to $30 x$ )

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 ( $<1 \mathrm{kB}$ ).
6. A separate multiplier is not needed.

## Summary

1. Selection of algorithms suitable for constrained environments:

- Elliptic-curve versions of DH, EIGamal, 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)

