# Kriterien zur optimalen Auswahl von Elliptic Curve Cryptography als Hard- oder Softwarelösung

Mathias Schmalisch · Hagen Ploog · Dirk Timmermann

Universität Rostock



Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock

Dipl.-Ing. Mathias Schmalisch

mathias.schmalisch@uni-rostock.de

# Übersicht

- Motivation
- Elliptische Kurven
- Algorithmen
- Invertierung
- Auswahl
- Vergleich


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Motivation

- ECC benötig große Anzahl an Operationen

- Möglichkeiten für höheren Durchsatz
  - Takterhöhung (linear, begrenzt durch Technologie)
  - Parallelisierung (linear, Fläche wächst linear)
  - Algorithmische Optimierung (> linear)

- Algorithmische Optimierungen sind technologieunabhängig und bringen daher zusätzlichen Durchsatz

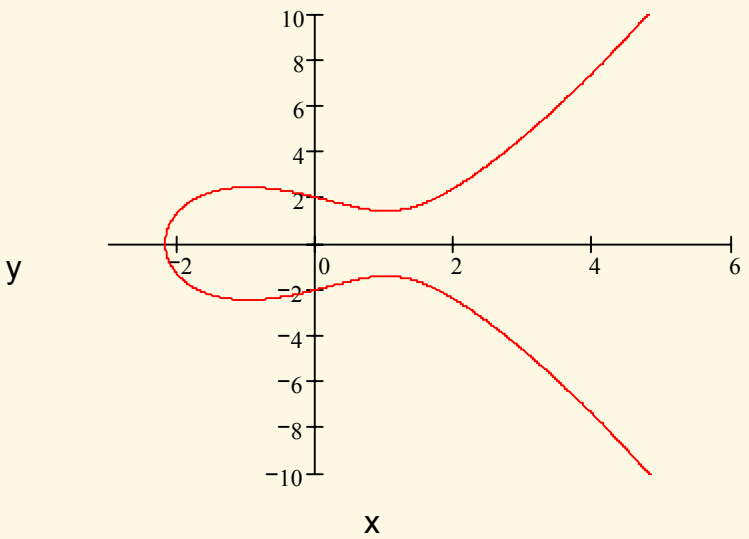

Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Übersicht

- Motivation
- **Elliptische Kurven**
- Algorithmen
- Invertierung
- Auswahl
- Vergleich


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Elliptische Kurven - Einführung




- Weierstrass Normalform:

  E: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

  $x, y, a_i \in GF(q)$

- Elliptische Kurven werden auf endliche Körper GF(q) abgebildet

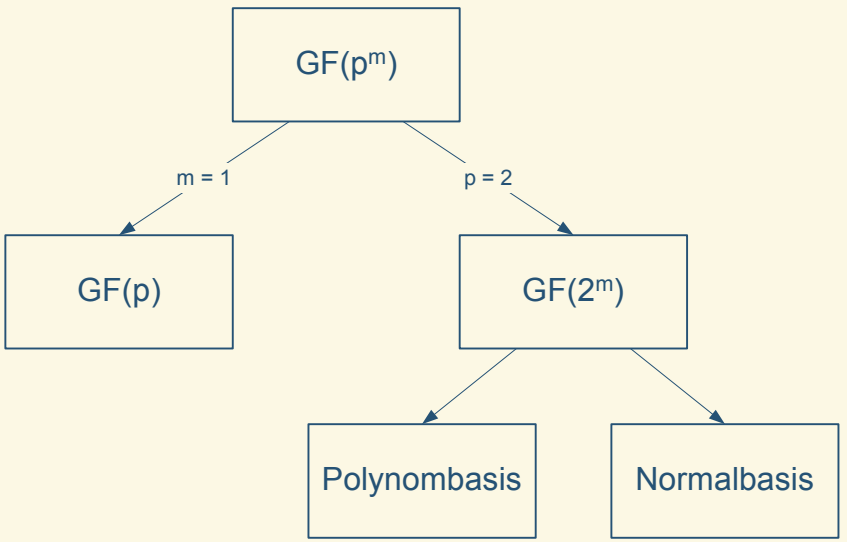
- Elliptische Kurven dürfen nicht singulär sein ($\Delta \neq 0$)


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Endliche Körper GF(q)

- Für endliche Körper sind verschiedene Bezeichnungen bekannt ( $GF(q) = \mathbb{F}_q = \mathbb{Z}_q$ )
- Wobei $q = p^m$ , p ist eine Primzahl
- In der Kryptographie finden die beiden Sonderfälle m = 1 oder p = 2 Anwendung
- Für Hardwarelösungen ist der Sonderfall p = 2 interessant
- $char(GF(2^m)) = 2$

  E: $y^2 + xy = x^3 + ax^2 + b$





Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Punktoperationen in GF($2^m$)

- Punktaddition: $P \neq Q$

  $\lambda = (y_1 + y_2) * (x_1 + x_2)^{-1}$

- Punktverdopplung: $P = Q$

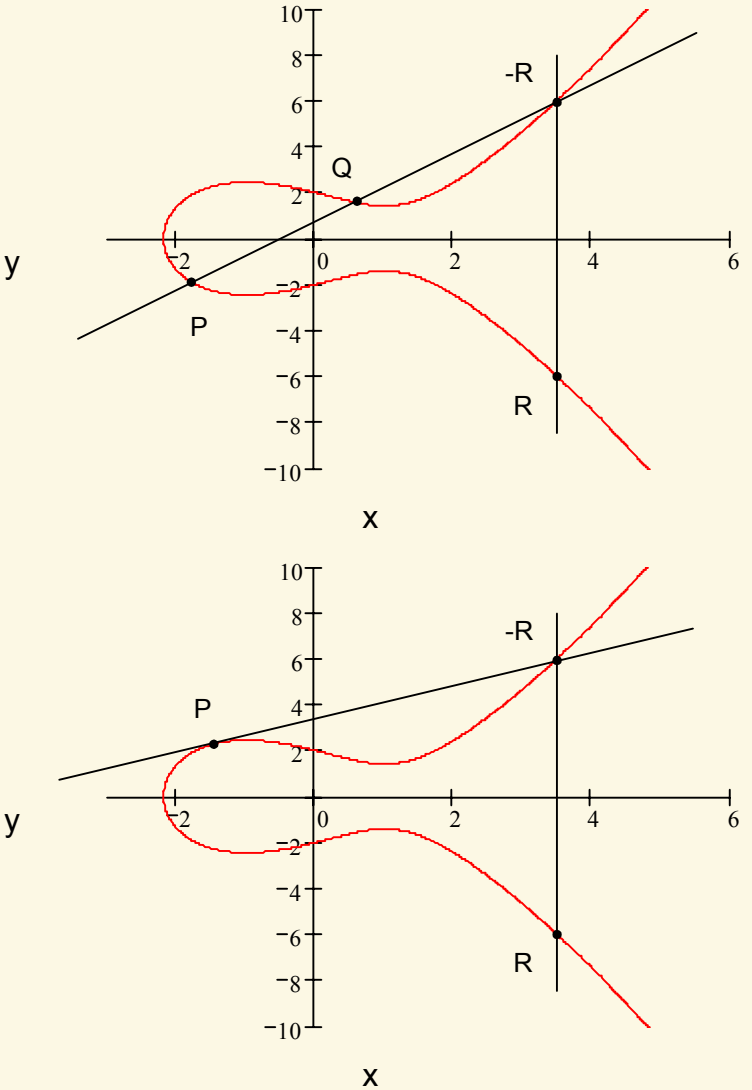
  $\lambda = x_1 + y_1 * (x_1)^{-1}$

  $x_3 = \lambda^2 + \lambda + a + x_1 + x_2 = x_1^2 + b * (x_1)^{-2}$

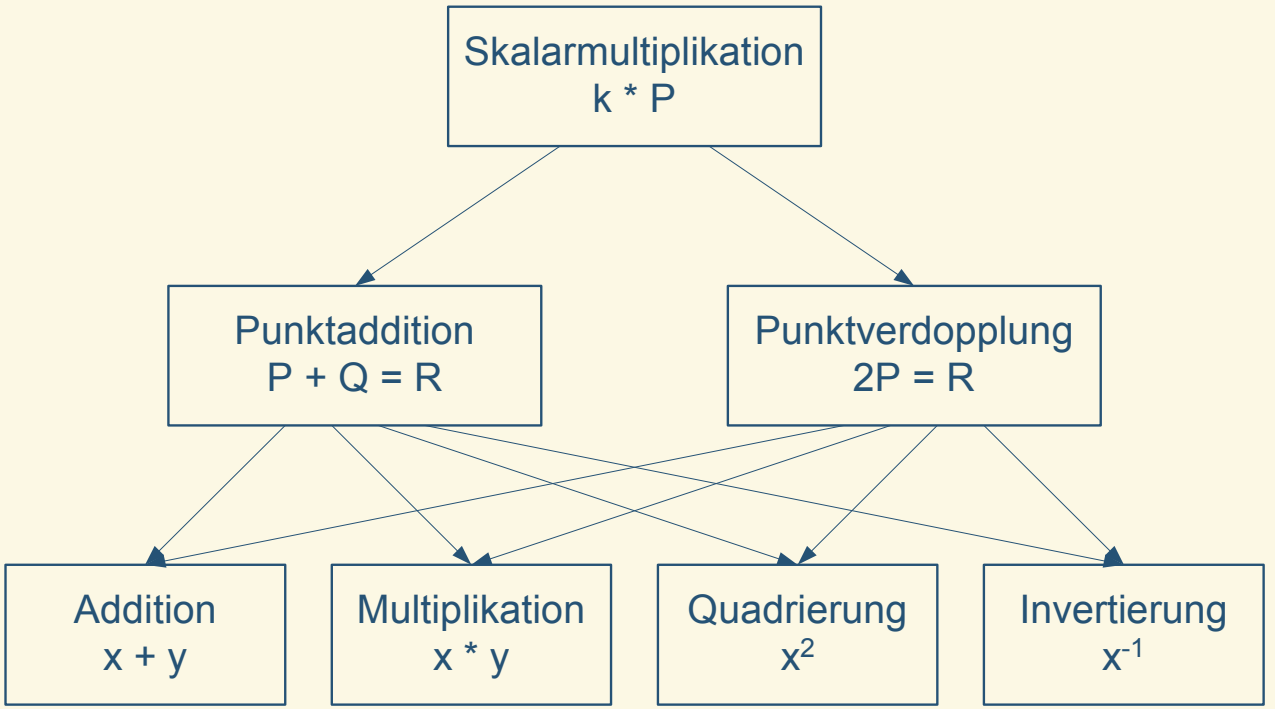
- Berechnung von R

  $x_3 = \lambda^2 + \lambda + a + x_1 + x_2$

  $y_3 = \lambda * x_3 + x_3 + x_1^2$



# Skalarmultiplikation k * P




Skalarmultiplikation

Punktoperationen auf der elliptischen Kurve

Operationen im endlichem Körper

# Beispiel

- Diffie-Hellmann Schlüsselaustausch
- Alice wählt geheime Zahle a und Bob die Zahl b
- Alice und Bob einigen sich auf eine ell. Kurve E und Punkt P
- Alice berechnet PA = a * P
- Bob berechnet PB = b * P
- PB und PA werden ausgetauscht
- Alice berechnet Schlüssel x mit  x = a * PB
- Bob berechnet Schlüssel x mit x = b * PA

$$b * PA = b * (a * P) = b * (P * a) = (b * P) * a = a * PB$$

# Übersicht

- Motivation
- Elliptische Kurven
- **Algorithmen**
- Invertierung
- Auswahl
- Vergleich


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Skalarmultiplikation in GF(q) – Verfahren

Input: Integer $k > 0$ mit m Bitstellen, Punkt P auf der elliptischen Kurve

Output : $Q = k * P$

## Double and Add

1. $Q = P$
2. for i from m-2 downto 0 do
3. $\quad Q = 2 * Q$
4. $\quad$ if $k_i = 1$ then
5. $\quad\quad Q = Q + P$
6. return (Q)

## Montgomery

1. $P_1 = P, P_2 = 2P$
2. for i from m-2 downto 0 do
3. $\quad$ if $k_i = 1$ then
4. $\quad\quad P_1 = P_1 + P_2, P_2 = 2 * P_2$
5. $\quad$ else
6. $\quad\quad P_1 = 2 * P_1, P_2 = P_1 + P_2$
7. return $(Q = P_1)$


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Skalarmultiplikation in GF(q) – Bewertung

## Double and Add

- Vorteile:
  - ▸ Weniger Punktoperationen
- Nachteil:
  - ▸ Anzahl Operationen hängen vom Hamminggewicht von k ab

## Montgomery

- Aus den x-Koordinaten zweier benachbarter Punkte können die y-Koordinaten bestimmt werden

- Vorteil:
  - ▸ nur x-Kooridnaten sind zu berechnen
  - ▸ Anzahl Operationen unabhängig vom Hamminggewicht von k

# Ebenen

## Affine Ebene

- (x, y) Koordinaten
- Umrechnung: Y = y, X = x, Z = 1
- Vorteil:
  - ▸ einfachere Berechnung
- Nachteile:
  - ▸ viele Invertierungen

## Projektive Ebene

- (X, Y, Z) Koordinaten
- Umrechnung: y = Y / Z, x = X / Z
- Vorteil:
  - ▸ Punkte der affinen Ebene bilden in der projektiven Ebene eine Gerade
  - ▸ nur eine Invertierung notwendig
- Nachteil:
  - ▸ kompliziertere Berechnung

# Übersicht

- Motivation
- Elliptische Kurven
- Algorithmen
- **Invertierung**
- Auswahl
- Vergleich


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Invertierung in GF(q)

## Erweiterter Euklidischer Algorithmus

$$ax + by \equiv 1 \bmod b$$

Input: zwei Zahlen a, b mit $b \geq a$,
b – Primzahl

Output: $x = a^{-1}$

1. $x_2 = 1, x_1 = 0, y_2 = 0, y_1 = 1$
2. while $b > 0$ do
3. $q = \lfloor a / b \rfloor, r = a - qb$
4. $x = x_2 - qx_1, y = y_2 - qy_1$
5. $a = b, x_2 = x_1, x_1 = x$
6. $b = r, y_2 = y_1, y_1 = y$
7. return $(x = x_2)$

## Fermat‘sches Theorem

$$a^b \bmod b \equiv a$$

$$a^{b-2} \bmod b \equiv a^{-1}$$

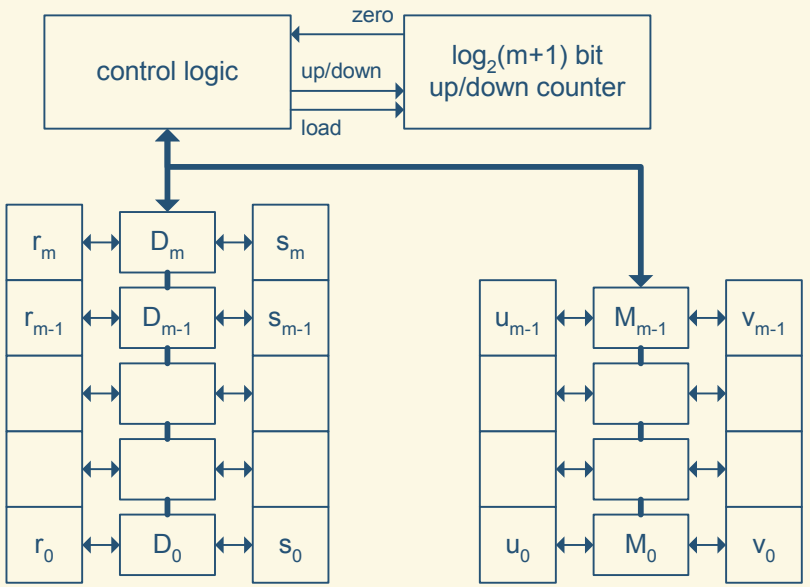
- b ist Primzahl
- ggT(a, b) = 1


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Brunner, Curiger, Hofstetter

- Berechnung nach dem Erweiterten Euklidischen Algorithmus
- Geeignet für endliche Körper GF($2^m$) – Polynombasis

- Kleinste dokumentierte Hardwarerealisierungen
- Berechnet Invertierung in m Takten
- Leicht parallelisierbar



# Itoh, Tsujii

- Berechnung nach dem Fermat‘schen Theorem
- Geeignet für alle endlichen Körper $GF(p^m)$

- Berechnung mit Hilfe von Multiplikationen und Quadrierungen

- Anzahl Multiplikationen:

$$\lfloor \log_2(m-1) \rfloor + H_w(m-1) - 1$$

- Anzahl Quadrierungen:

$$m - 1$$


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Übersicht

- Motivation
- Elliptische Kurven
- Algorithmen
- Invertierung
- **Auswahl**
- Vergleich


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Anzahl Operationen

| Algorithmus | | Affine Ebene | | Projektive Ebene | |
|---|---|---|---|---|---|
| | | D. and A. | Montgomery | D. and A. | Montgomery |
| $P_1 + P_2$ | + | 8 | 3 | 7 | 2 |
| | * | 2 | 1 | 15 | 4 |
| | $x^2$ | 2 | 1 | 7 | 1 |
| | $x^{-1}$ | 1 | 1 | 0 | 0 |
| $2 * P_1$ | + | 4 | 1 | 4 | 1 |
| | * | 3 | 1 | 5 | 2 |
| | $x^2$ | 2 | 1 | 7 | 4 |
| | $x^{-1}$ | 1 | 1 | 0 | 0 |
| k * P | + | 8h + 4m | 4m + 6 | 7h + 4m | 3m + 7 |
| | * | 2h + 3m | 2m + 4 | 15h + 5m + 3 | 6m + 10 |
| | $x^2$ | 2h + 2m | 2m + 2 | 7h + 7m + 1 | 5m + 3 |
| | $x^{-1}$ | h + m | 2m + 1 | 1 | 1 |

m = $\log_2$ k; h = Hamminggewicht von k (∅ = m / 2)


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Auswahl

- Software: Montgomery Algorithmus über projektive Ebene
  - Invertierung mit Algorithmus von Itoh, Tsujii
- Hardware: Montgomery Algorithmus über affine Ebene
  - Invertierung mit Hardwarelösung von Brunner, Curiger, Hofstetter

- Beispiel Hardware (serielle Implementierung) m = 167:
  - Addition: 1 Takt
  - Multiplikation: m Takte
  - Quadrierung: m/2 Takte
  - Invertierung: affine Ebene | projektive Ebene
    m Takte | 15614 Takte

| Algorithmus | Affine Ebene | | Projektive Ebene | |
|---|---|---|---|---|
| | D. and A. | Montgomery | D. and A. | Montgomery |
| Anzahl Takte | 197148 | 141289 | 514908 | 255518 |
| Prozentual | 77 % | 55 % | 202 % | 100 % |


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Übersicht

- Motivation
- Elliptische Kurven
- Algorithmen
- Invertierung
- Auswahl
- **Vergleich**


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Vergleich

- Aktuell schnellste dokumentierte Hardwareimplementierung von Gerardo Orlando und Christof Paar (CHES 2000)
  - Benutzen Montgomery Algorithmus über projektive Ebene
  - Invertierung nach Itoh, Tsujii
  - Teilweise parallelisierte Implementierung
  - Berechnung von k * P in 0.21 ms

- Resultierende Idee für unsere Hardwarerealisierung:
  - Benutzung von Montgomery Algorithmus über affine Ebene
  - Invertierung nach Brunner, Curiger, Hofstetter
  - Geschwindigkeitssteigerung von ca. 38 % zu erwarten


Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@uni-rostock.de

# Vergleich mit RSA Verfahren

- Schlüssellänge wesentlich kürzer als bei RSA Verfahren
- RSA Schlüssellänge steigt schneller an
- bei kurzen Nachrichten (Schlüsselaustausch) viel Overhead