Kryptographie mit elliptischen Kurven

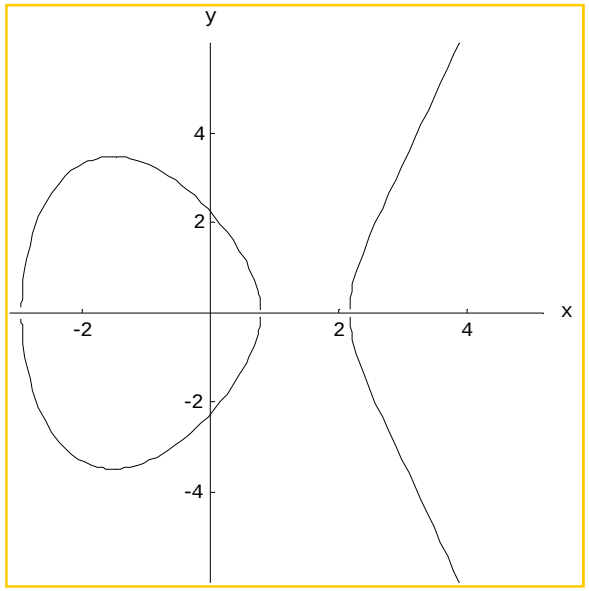
Markus Hermann, Dejan Lazich

Micronas GmbH

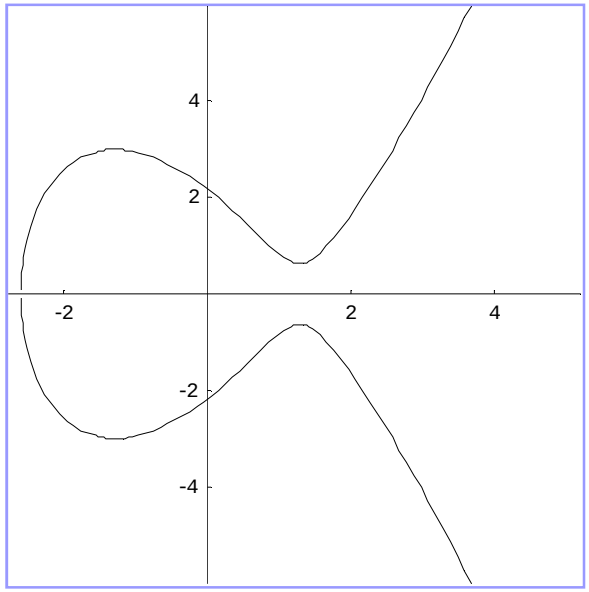
Freiburg, 2006

Elliptische Kurven (EK) über $\mathbb{R}$

- EK sind Teilmengen der Ebene $\mathbb{R}$x $\mathbb{R}$, definiert durch a und b in Gleichungen der Form $y^2 = x^3 + ax + b$
- Es gibt zwei Formen: „Küste mit Insel“ und „Kleiderbügel“
- Beispiele:



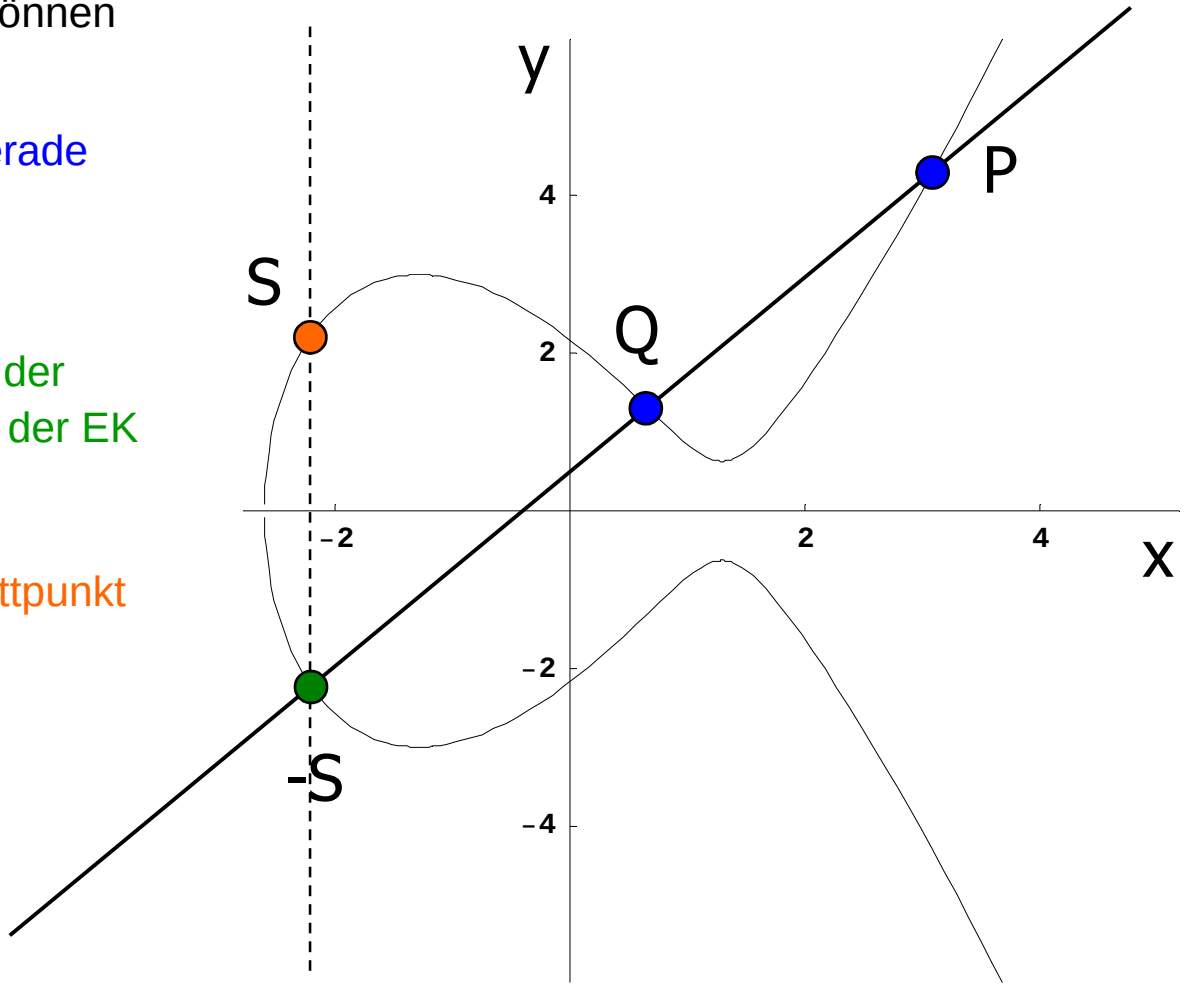
$y^2 = x^3 - 7x + 5$



$y^2 = x^3 - 5x + 4.7$

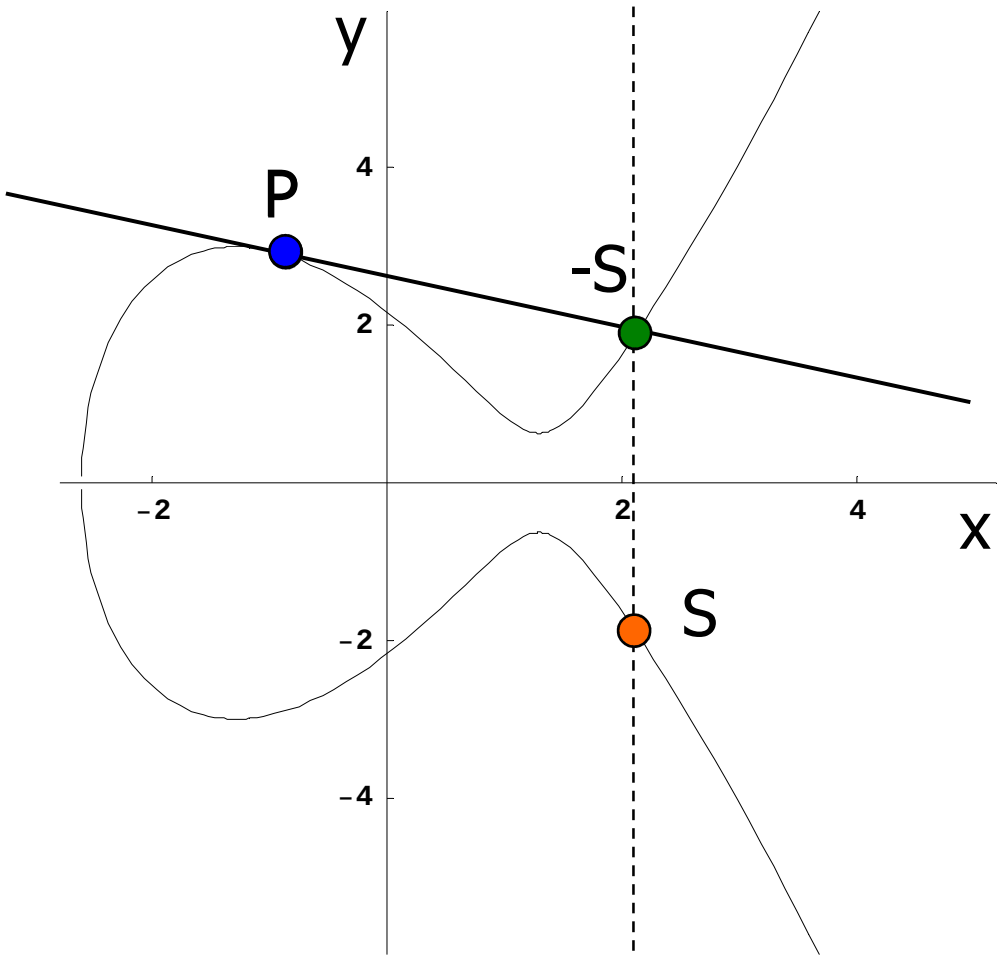
Beispiel zur Punktaddition P ⊕ Q=S

- Zwei Kurvenpunkte P, Q können „addiert“ werden:
  - Ziehe Verbindungsgerade zwischen P und Q
  - Ermittle Schnittpunkt der Verbindungsgerade mit der EK
  - Spiegle diesen Schnittpunkt an der x-Achse



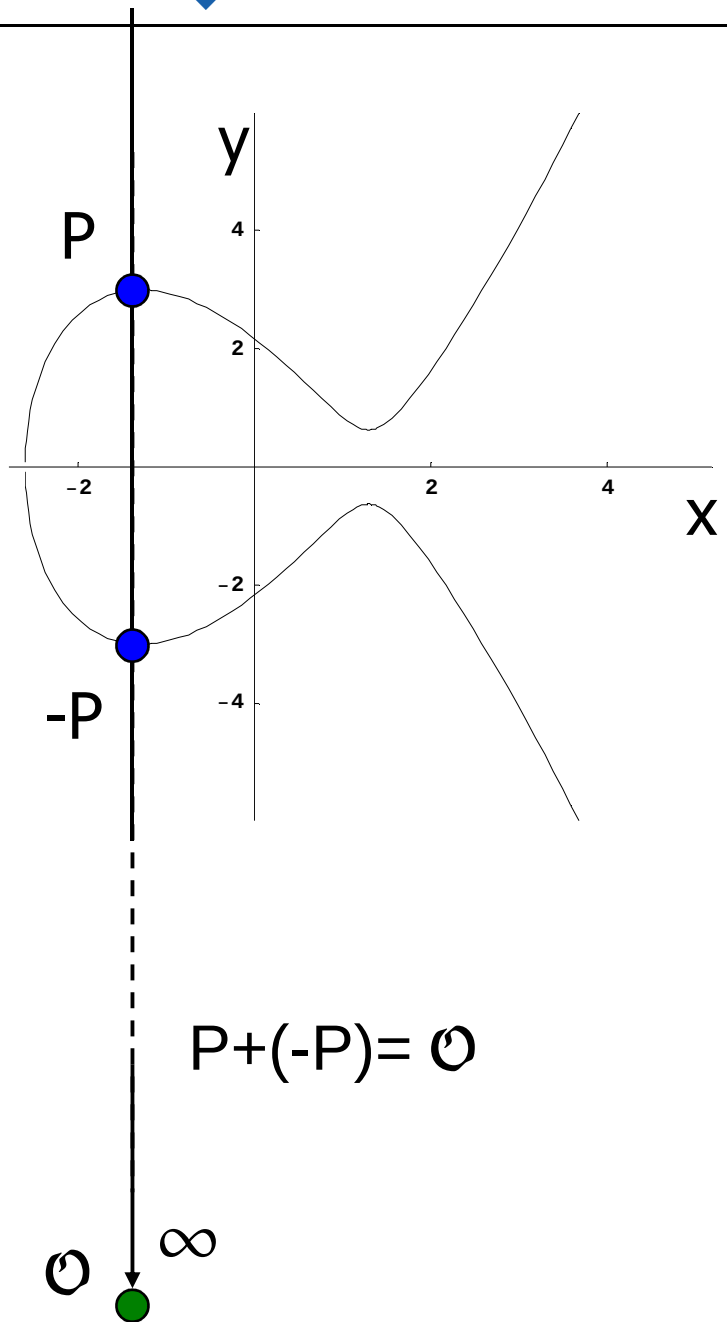
Beispiel zur Punktverdopplung ② P=S

- Sonderfall P + Q mit P = Q („Verdopplung“):
  - Zeichne Tangente der EK im Punkt P
  - Ermittle Schnittpunkt der Tangente mit der EK
  - Spiegle diesen Schnittpunkt an der x-Achse



Der abstrakte Punkt $\mathcal{O}$

- Sonderfall P + Q mit Q = -P:
  - Ziehe Verbindungsgerade zwischen P und –P
  - Sie ist senkrecht zur x-Achse, hat keinen dritten Schnittpunkt mit der EK
  - Definiere: Die Verbindungsgerade schneidet die EK „im Unendlichen“, im abstrakten Punkt $\mathcal{O}$



Gruppenstruktur

- Man kann die Punkte einer EK:
  - Addieren: $P \oplus Q$
  - Subtrahieren: $P \ominus Q$
  - Verdoppeln: $2P$
  - Und: $2P \oplus P = 3P$

    $2P \oplus 2P = 4P$

    $2P \oplus 2P \oplus P = 5P$

Hier kann Punktaddition und -Verdopplung beliebig kombiniert werden (Additionsketten, Double-and-Add, usw.)

=> Mit ganzen Zahlen multiplizieren: nP

- Das bedeutet: Wir können die Punkte einer EK „ganz normal“ zueinander addieren, subtrahieren und mit n multiplizieren.

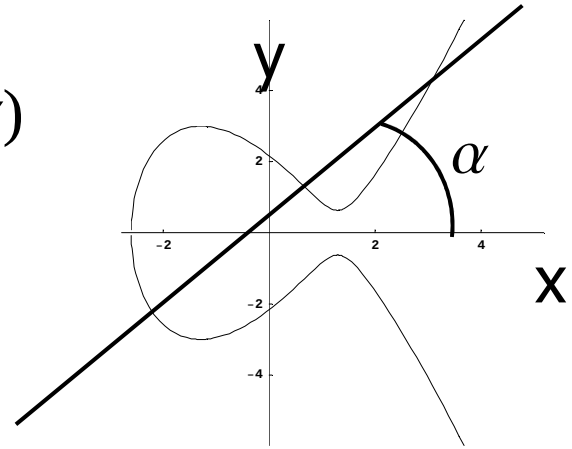
MICRONAS

Formeln für die Punktarithmetik

- Punktaddition:

$$P=(x_P,y_P),\ Q=(x_Q,y_Q)\Rightarrow P \oplus Q=S=(x_S,y_S)$$

$$\left.\begin{aligned} x_S &= s^2-x_P-x_Q \\ y_S &= s(x_P-x_S)-y_P \end{aligned}\right\} \quad s=\frac{y_Q-y_P}{x_Q-x_P}=\tan(\alpha)$$



- Punktverdopplung:

$$P=(x_P,y_P)\Rightarrow \quad ② P=S=(x_S,y_S)$$

$$\left.\begin{aligned} x_S &= \sigma^2-2x_P \\ y_S &= \sigma(x_P-x_S)-y_P \end{aligned}\right\} \quad \sigma=\frac{3x_P^3+a}{2y_P}$$

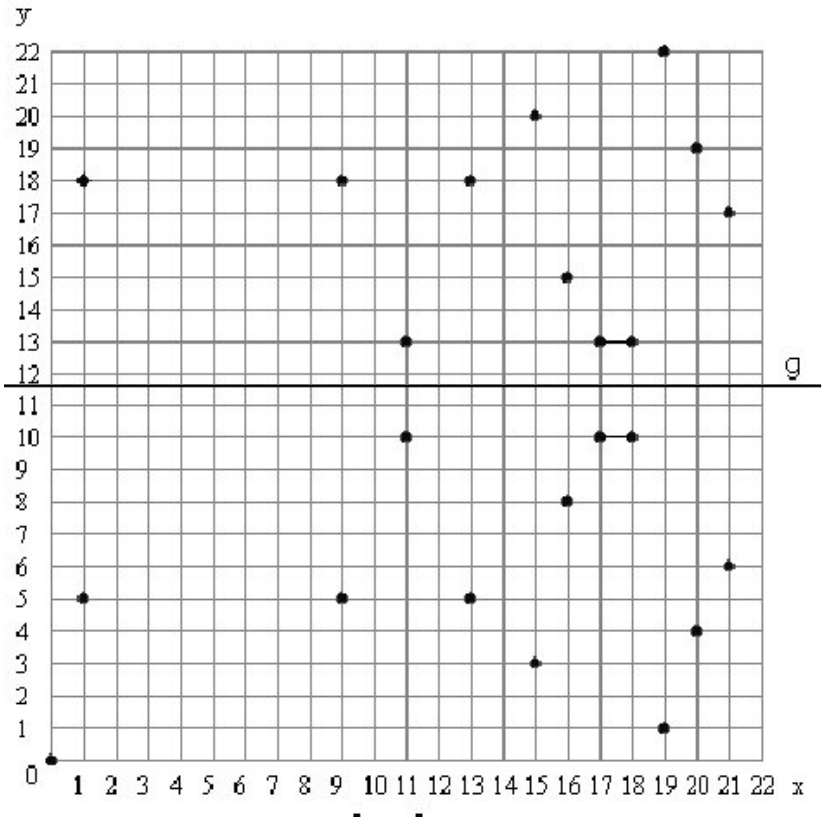
MICRONAS

Elliptische Kurven über endlichen Körpern ($\mathbb{F}_p$, $\mathbb{F}_{2^n}$)

- Anschaulich sind das keine Kurven mehr, sondern Punktmengen. Bsp.: Java-Applet

- Formeln können jedoch trotzdem (im Fall $\mathbb{F}_{2^n}$ mit kleinen Anpassungen) verwendet werden!

- Formeln werden dann im jeweiligen Grundkörper berechnet => Punktarithmetik wird auf Arithmetik in endlichen Körpern zurückgeführt.



Anwendung in der Kryptographie

- Als Grundlage für Public-Key-Kryptographie benötigt man sog. Einwegfunktionen:
  - Berechnung des Funktionswertes einfach („leichte Funktionen“)
  - Berechnung des ursprünglichen Arguments aus dem Funktionswert „unmöglich“ („schwere Funktionen“)
- Zwei Einwegfunktionen für Public-Key-Verfahren:

(1) Multiplikation v. Primzahlen -> Primfaktorzerlegung (RSA)

$x \cdot y = ?$ -> $? \cdot ? = z$

(2) Diskrete Exponentation -> diskreter Logarithmus (DH, EG, DSA)

$x^e = ? \bmod p$ -> $x^? = z \bmod p$

$(?^e = z \bmod p)$

Diskrete Logarithmen

- Das diskrete Logarithmus Problem (DLP):

  Gegeben: Zahlen x und y mit $y = \underbrace{x \cdot x \cdot \cdots \cdot x}_{\text{k-mal}} = x^k \bmod p$.

  Frage: Wie oft wurde x modulo p mit sich selbst multipliziert, um y zu erhalten?

  Gesucht: k 

- Das DLP auf elliptischen Kurven (ECDLP):

  Gegeben: EK-Punkte P und Q mit $Q = \underbrace{P \oplus P \oplus \cdots \oplus P}_{\text{k-mal}} = k\,P$ auf der EK.

  Frage: Wie oft wurde der Punkt P zu sich selbst addiert, um den Punkt Q zu erhalten?

  Gesucht: k 

MICRONAS

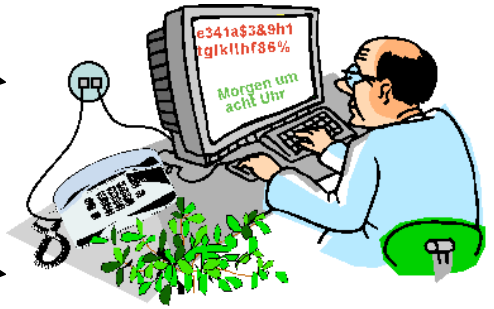
Diffie-Hellman in Gruppen

Alice

Generator g, Primzahl p

$g^a \bmod p$

$g^b \bmod p$



Bob

: a

: b

$X=(g^b)^a=g^{ab} \bmod p$

Kennt g, p, g^a, g^b aber nicht X!

$X=(g^a)^b=g^{ab} \bmod p$

MICRONAS

Diffie-Hellman auf elliptischen Kurven

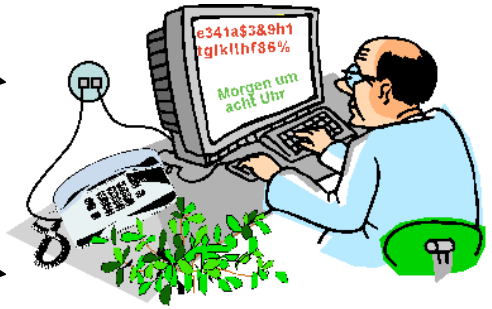
Alice

: u

Punkt P, EK

ⓤP

ⓥP



Bob

: v

X = ⓤ(ⓥ P)

=ⓤⓥP

Kennt P, EK,

ⓤP, ⓥP

aber nicht X!

X = ⓥ(ⓤ P)

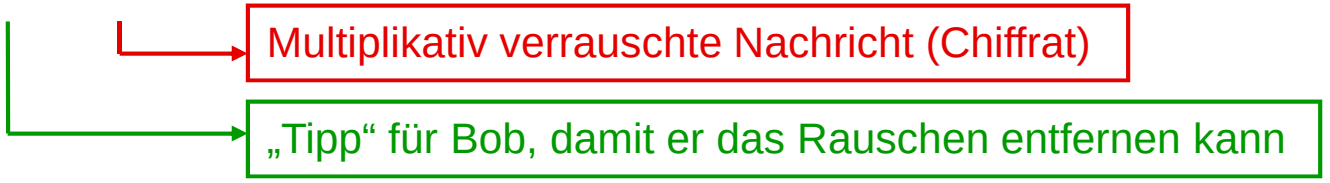
=ⓤⓥP

MICRONAS

ElGamal-Algorithmus in Gruppen

- Bob würfelt Zufallszahl a (geheimer Schlüssel) und veröffentlicht g^a (öffentlicher Schlüssel)

- Alice wählt Zufallszahl k und chiffriert die Nachricht m zu (g^k, mg^{ak})



- Bob kennt a und kann durch den „Tipp“ g^k das Rauschen g^{ak} entfernen, um die Nachricht m zu erhalten.

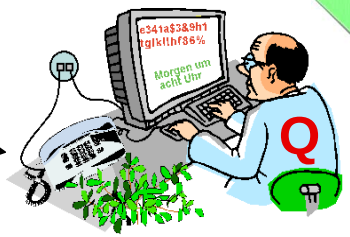
- Zahlreiche Varianten standardisiert

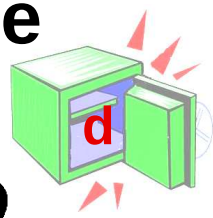
MICRONAS

Elliptic Curve Integrated Encryption Scheme (ECIES)

Punkt P, EK, ENC, DEC, MAC, KDF

Alice





Bob

Öff. Schlüssel Q, Kofaktor h

: u

$R = u \cdot P$

$Z = h \cdot u \cdot Q = (x, y)$

$(k_1, k_2) = KDF(x, R)$

$C = ENC_{k_1}(m)$

$t = MAC_{k_2}(C)$

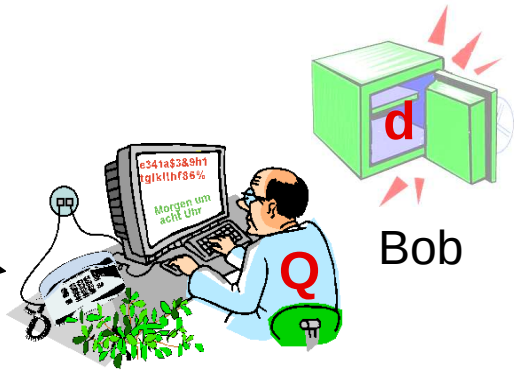
C, t, R

MICRONAS

ECIES (2)

Alice

Punkt P, EK, ENC, DEC, MAC, KDF



Bob

C, t, R

Geh. Schlüssel d, Kofaktor h

$Z = h \cdot d \cdot R = (x, y)$

$(k_1, k_2) = KDF(x, R)$

$t' = MAC_{k_2}(C)$

$t == t'$?

$m = DEC_{k_1}(C)$

ENC, DEC: Symmetrische Chiffre, bspw. 3DES, AES, ...

MAC: Message Authentication Code, bspw. HMAC (im Prinzip Hashfkt., bspw. SHA-1)

KDF: Key Derivation Function (im Prinzip Hashfkt., bspw. SHA-1)

MICRONAS

Digital Signature Algorithm (DSA) mit elliptischen Kurven (EC-DSA) (1)

- Signieren einer Nachricht m

Bekannt ist ein Punkt P auf der EK, seine Ordnung n (n prim, $n>2^{191}$) sowie der geheime Schlüssel d (eine Zahl).

- Wähle Zufallszahl $k \in [0,n-1]$
- Berechne $(x,y):= k \cdot P$ (k-fache Addition eines Kurvenpunktes)
- Berechne $r:=x \bmod n$
- Berechne $e:=SHA\text{-}1(m)$
- Berechne $s:=k^{-1}(e+dr) \bmod n$
- Signatur ist das Tupel (r, s)

Nur x-Koordinate wird weiterverwendet

m, (r, s) →

m, (r, s) →

MICRONAS

DSA mit elliptischen Kurven (EC-DSA) (2)

- Verifizieren einer Signatur (r, s)

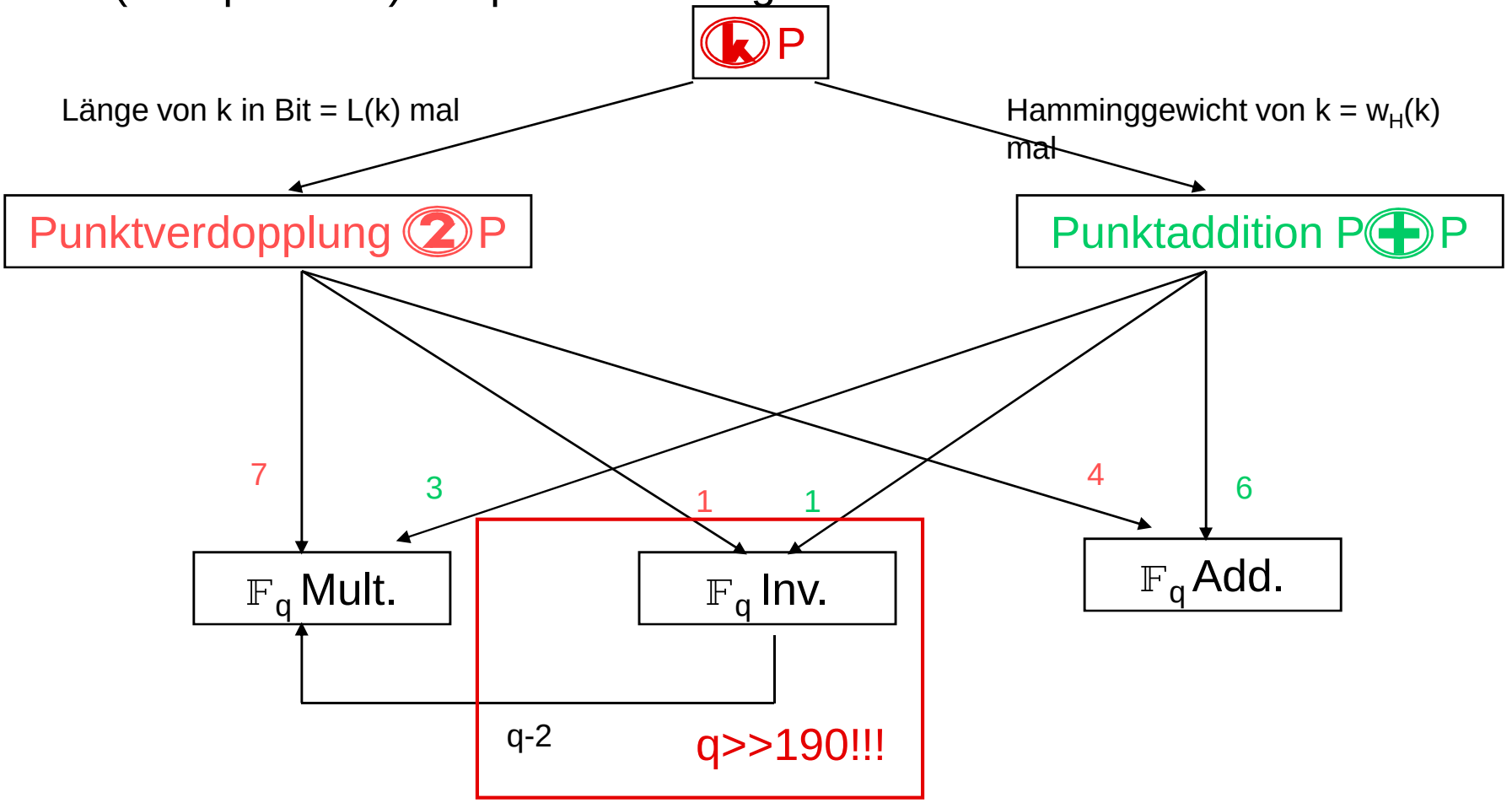
  Bekannt ist ein Punkt P auf der EK, seine Ordnung n (n prim, $n>2^{191}$) sowie der öffentliche Schlüssel Q (ein Punkt auf der EK).

  - Berechne e:=SHA-1(m)
  - Berechne $w:=s^{-1} \bmod n$
  - Berechne $u:=ew \bmod n$
  - Berechne $v:=rw \bmod n$
  - Berechne (x,y):= uP$\oplus$$v$Q (u- bzw. v-fache Addition von P bzw. Q)
  - Berechne $t:=x \bmod n$
  - Wenn t==r, dann Signatur gültig

Nur x-Koordinate wird weiterverwendet

DSA mit elliptischen Kurven (EC-DSA) (3)

- Welchen Aufwand hat eine der anfallenden mehrfachen Additionen (Multiplikation)? Implementierung mit Double-and-Add:

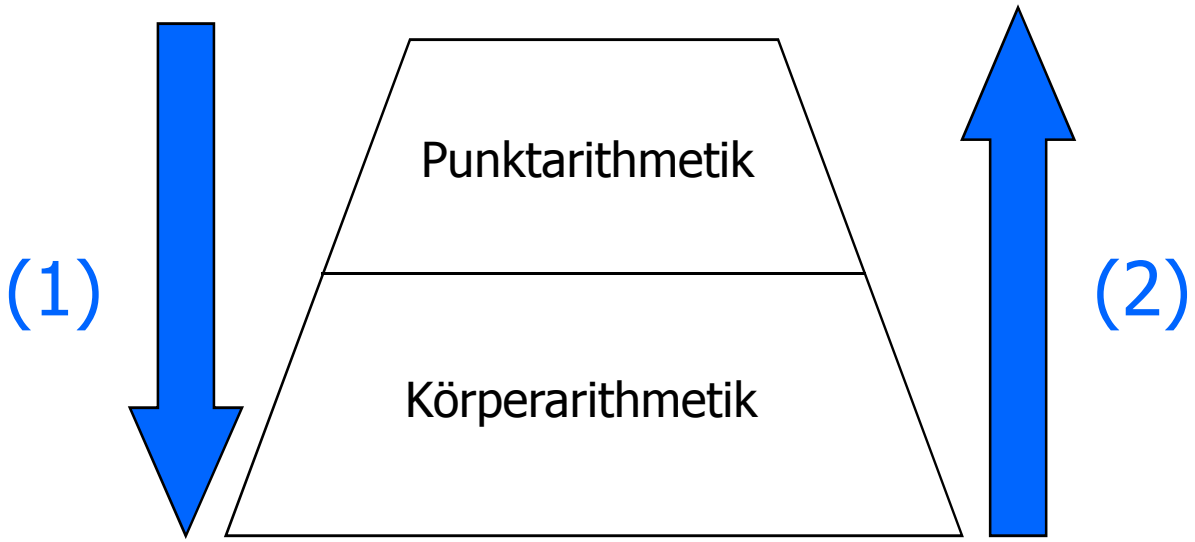


Wie erhält man schnelle Punktarithmetik?

- Zwei Ansätze:

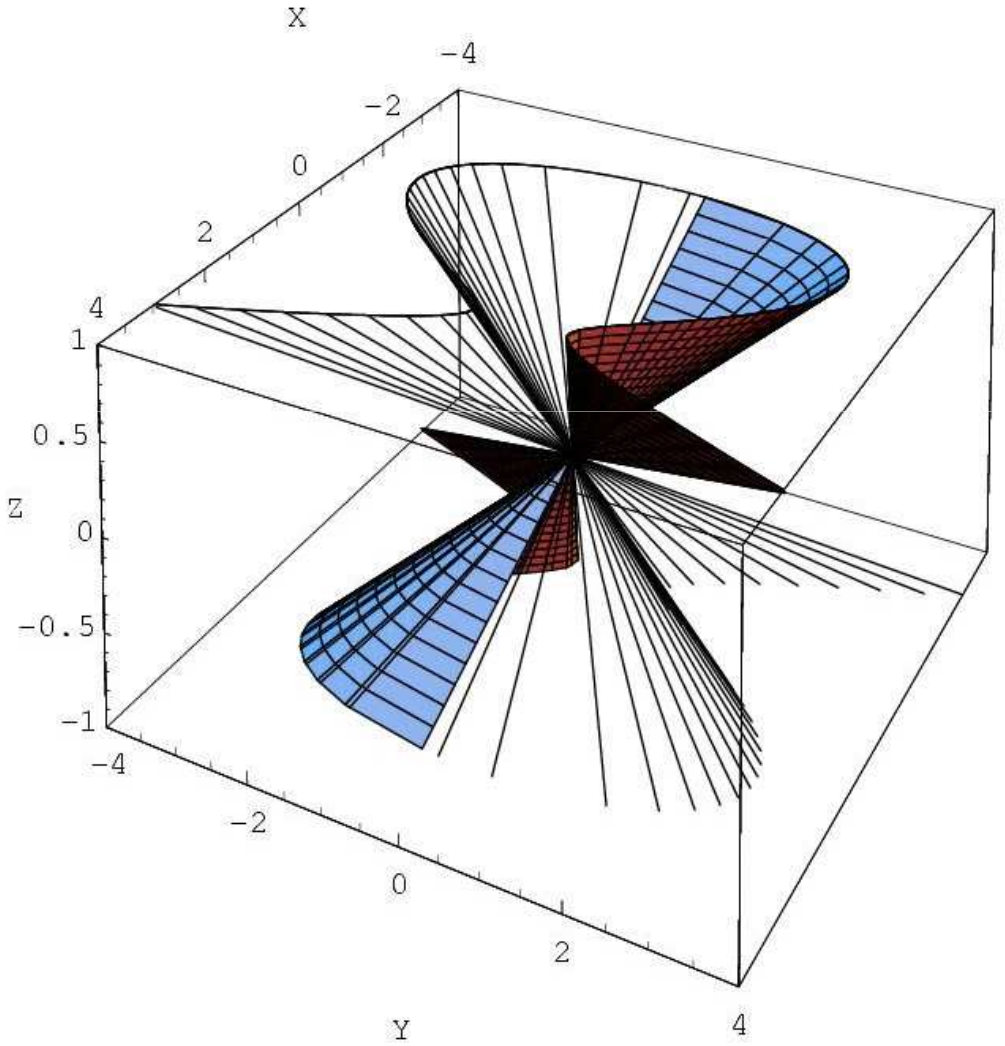
(1) Verändere die Darstellung der Punkte, so dass nur wenige Körperoperationen, nach Möglichkeit keine Invertierungen nötig sind!

(2) Implementiere möglichst schnelle Körperoperationen!



Ansatz (1): Projektive Standarddarstellung

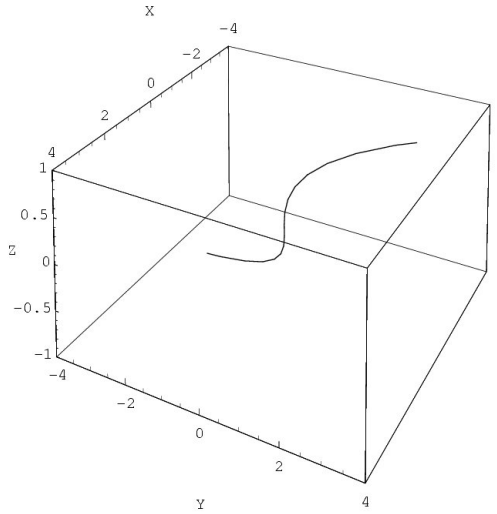
- Jeder Kurvenpunkt wird eindeutig durch eine Ursprungsgerade (einen Strahl) dargestellt
- Zur Addition von zwei Kurvenpunkten können beliebige Punkte auf deren Ursprungsgeraden verwendet werden
- Dadurch können Invertierungen gegen andere Operationen eingetauscht werden.
- Aber: Transformation in die projektive Darstellung und zurück erforderlich!

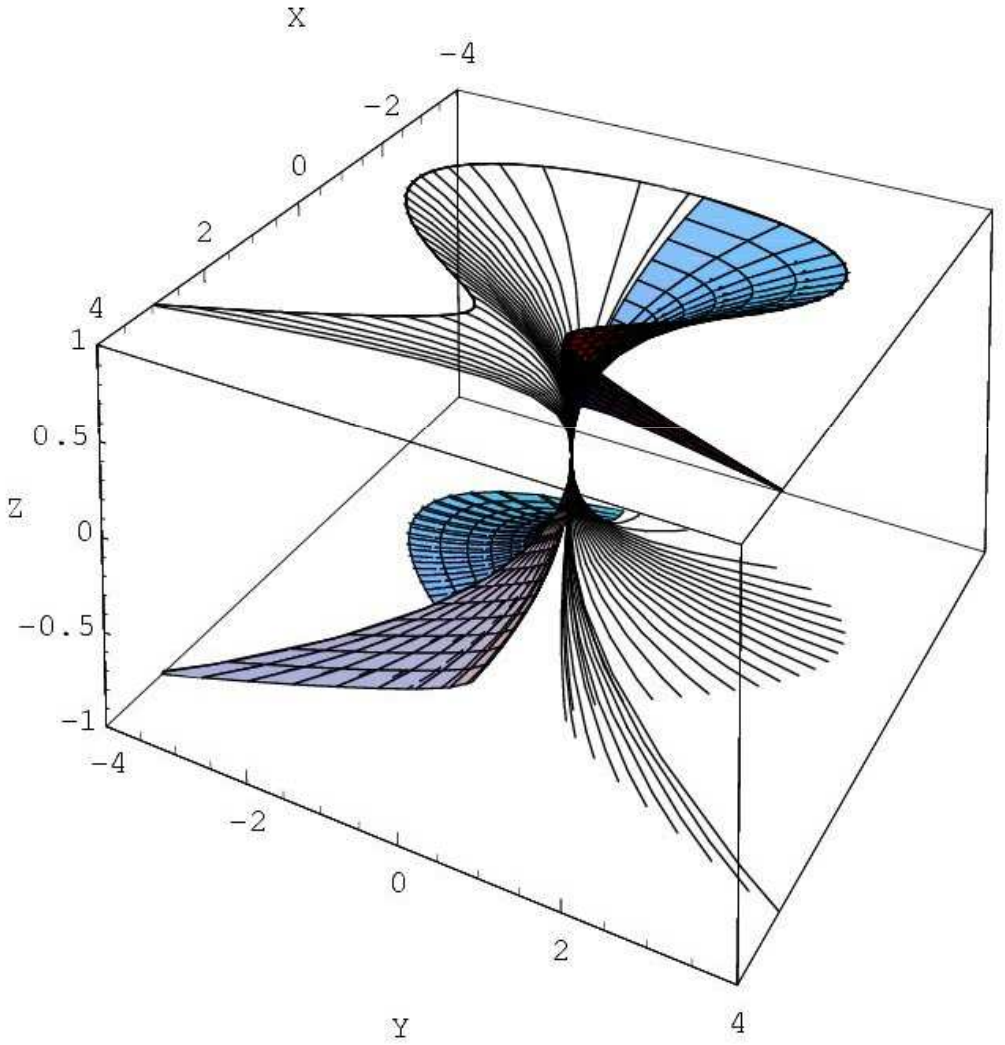


Ansatz (1): Projektive Jacobi-Darstellung

- Alternative zur projektiven Standarddarstellung

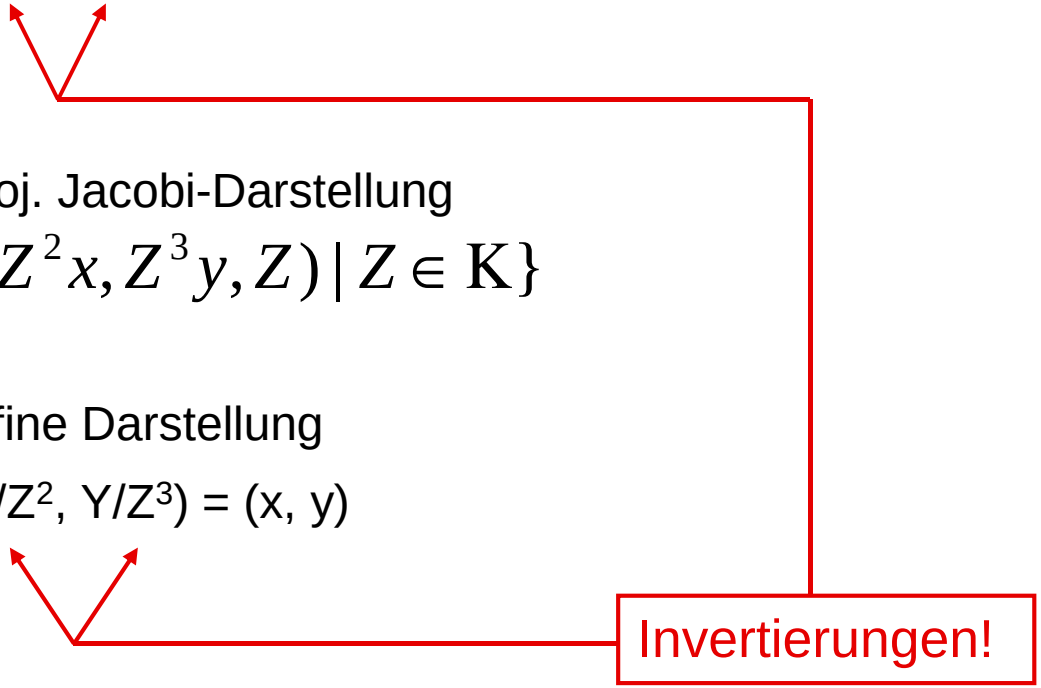
- Jeder Kurvenpunkt wird durch eine kubische Funktion (einen gekrümmten Strahl) dargestellt





Ansatz (1): Transformationen der Darstellung

- Affine Darstellung (x, y)
  -> Proj. Standarddarstellung
  -> $\{(Zx, Zy, Z) \mid Z \in \mathrm{K}\}$
- Proj. Standarddarst. (X, Y, Z)
  -> Affine Darstellung
  -> (X/Z, Y/Z) = (x, y)
- Affine Darstellung (x, y)
  -> Proj. Jacobi-Darstellung
  -> $\{(Z^2x, Z^3y, Z) \mid Z \in \mathrm{K}\}$
- Proj. Jacobi-Darst. (X, Y, Z)
  -> Affine Darstellung
  -> $(X/Z^2, Y/Z^3) = (x, y)$



Ansatz (1): Projektiver Additionsalgorithmus

Algorithmus 22 Punktaddition auf $E(\mathbb{F}_p)$ in projektiver Standarddarstellung

1: **procedure** EC-Fp-Proj-Std-Add($P_{\text{Proj}} = (X_P : Y_P : Z_P), Q_{\text{Proj}} = (X_Q : Y_Q : Z_Q)$)
2: **assert** $\pm P_{\text{Proj}} \neq Q_{\text{Proj}}$
3: $t_1 \leftarrow X_P Z_Q$ ▷ 1 aM
4: $t_2 \leftarrow Y_P Z_Q$ ▷ 1 aM
5: $t_3 \leftarrow Y_Q Z_P - t_2$ ▷ 1 aM, 1 AS
6: $t_4 \leftarrow X_Q Z_P - t_1$ ▷ 1 aM, 1 AS
7: $t_5 \leftarrow t_4^2$ ▷ 1 aM
8: $t_6 \leftarrow t_4 t_5$ ▷ 1 aM
9: $t_7 \leftarrow t_5 t_1$ ▷ 1 aM
10: $t_8 \leftarrow Z_P Z_Q$ ▷ 1 aM
11: $Z_S \leftarrow t_6 t_8$ ▷ 1 aM
12: $t_9 \leftarrow t_3^2 t_8 - t_6 - 2t_7$ ▷ 2 aM, 1 kM, 2 AS
13: $X_S \leftarrow t_4 t_9$ ▷ 1 aM
14: $Y_S \leftarrow t_3(t_7 - t_9) - t_6 t_2$ ▷ 2 aM, 2 AS
15: **return** $S_{\text{Proj}} = (X_S : Y_S : Z_S)$
16: **end procedure**

Ansatz (1): Benötigte Körperoperationen

- Punktaddition

Darstellung	Mult.	Inv.	Add./Sub.	
Affine Darstellung	3	1	6	
Proj. Standarddarstellung	15	0	6	Zzgl. Transformation und Rücktransformation!
Proj. Jacobi-Darstellung	18	0	7	

- Punktverdopplung

Darstellung	Mult.	Inv.	Add./Sub.	
Affine Darstellung	7	1	4	
Proj. Standarddarstellung	18	0	4	Zzgl. Transformation und Rücktransformation!
Proj. Jacobi-Darstellung	16	0	4	
Proj. Jacobi-Darst. ($a \neq -3$)	14	0	5	

Erinnerung: 1 Inv. kostet >> 191 Mult.!!!

Ansatz (1): Wann sind proj. Darst. sinnvoll?

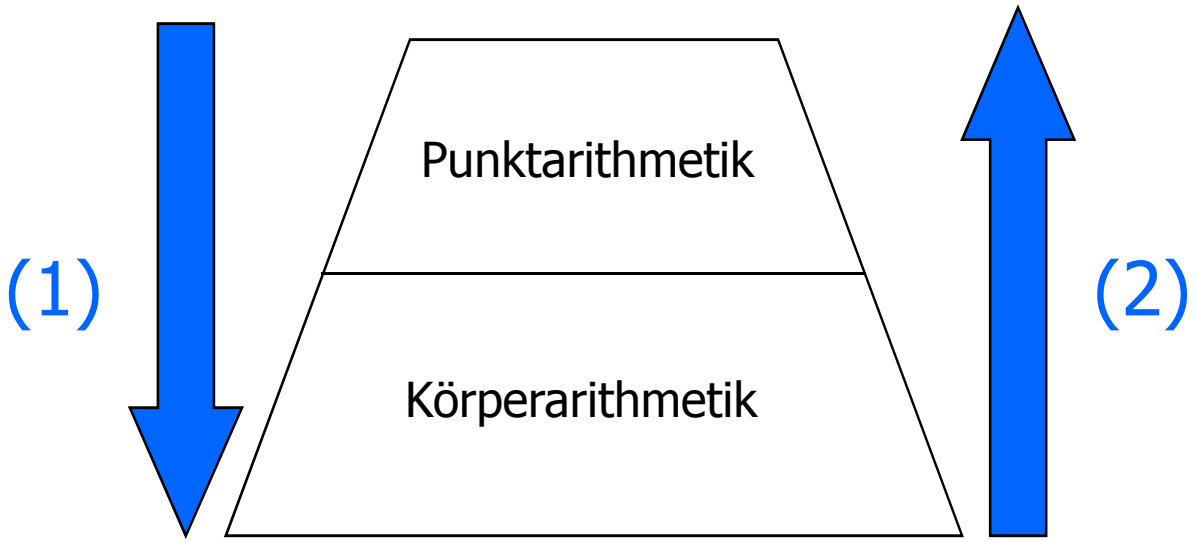
- Für eine einzelne Addition/Verdopplung sind die Transformationen zu teuer, außerdem mehr Multiplikationen notwendig.
- Fallen mehrere Operationen an (bspw. Multiplikation nP), dann gehe wie folgt vor:
  - Transformation in projektive Darstellung
    (keine Invertierung)
  - Führe alle Rechnungen in projektiver Darstellung durch
    (keine Invertierung)
  - Transformiere das Ergebnis zurück in affine Darstellung
    (zwei Invertierungen)

Wie erhält man schnelle Punktarithmetik?

- Zwei Ansätze:

(1) Verändere die Darstellung der Punkte, so dass nur wenige Körperoperationen, nach Möglichkeit keine Invertierungen nötig sind!

(2) Implementiere möglichst schnelle Körperoperationen!



MICRONAS

Ansatz (2): Schnelle Körperarithmetik

- Ziele:
  - Skalierbarkeit (nicht an einen bestimmten Körper gebunden)
  - Einheitlichkeit (soll für $\mathbb{F}_p$, als auch für $\mathbb{F}_{2^n}$ nutzbar sein)
  - Konform mit (möglichst allen) Standards
- Möglichkeiten:
  - Montgomery-Arithmetik (wortweise, eher für CPU geeignet)
  - Rückgekoppelte Schieberegister (bitweise, für Hardware-Lösungen (ASIC) geeignet)

MICRONAS

Standards

Standard	Jahr	Titel
ANSI X9.62	1999	The elliptic curve digital signature algorithm (ECDSA)
ANSI X9.63	2001	Key agreement and key transport
FIPS 186-2	2000	Digital Signature Standard (DSS) (ECDSA, ECDH)
IEEE 1363-2000	2000	Standard specifications for public-key cryptography (ECDSA, Projektive Jacobi-Darstellung)
IEEE 1363a	(draft)	Amendment 1: Additional techniques
ISO/IEC 15946-1	2002	Techniques based on elliptic curves-Part 1: General
ISO/IEC 15946-2	2002	Part 2: Digital Signatures (ECDSA)
ISO/IEC 15946-3	2002	Part 3: Key establishment
ISO/IEC 15946-4	(draft)	Part 4: Digital signatures giving message recovery
SEC 1	2000	Elliptic curve cryptography (ECDSA, ECDH)
SEC 2	2000	Recommended elliptic curve domain parameters
PKCS13	1998	RSA Laboratories

Zusammenfassung

- Auf elliptischen Kurven kann man „normal“ addieren und mit ganzen Zahlen multiplizieren (Gruppenstruktur).
- Es existieren Protokolle für Schlüsselaustausch, Verschlüsselung und Signatur mit elliptischen Kurven, sie wurden von verschiedenen Gremien standardisiert.
- Die Invertierung im Grundkörper ist die bezüglich der Laufzeit teuerste Operation, die zur Punktarithmetik notwendig ist.
- Man kann durch projektive Darstellungen Invertierungen einsparen.
- Offene Frage: Wie kann ein universelles Kryptomodul implementiert werden (für RSA, elliptische Kurven, Primkörper, erw. Körper, ...)?