Freie Universität Berlin

RSA vs. ECC

A non-expert view
by
Ralph-Hardo Schulz

- The Rivest-Shamir-Adleman-system (RSA) and the systems of

- Elliptic-curve-cryptography (ECC)

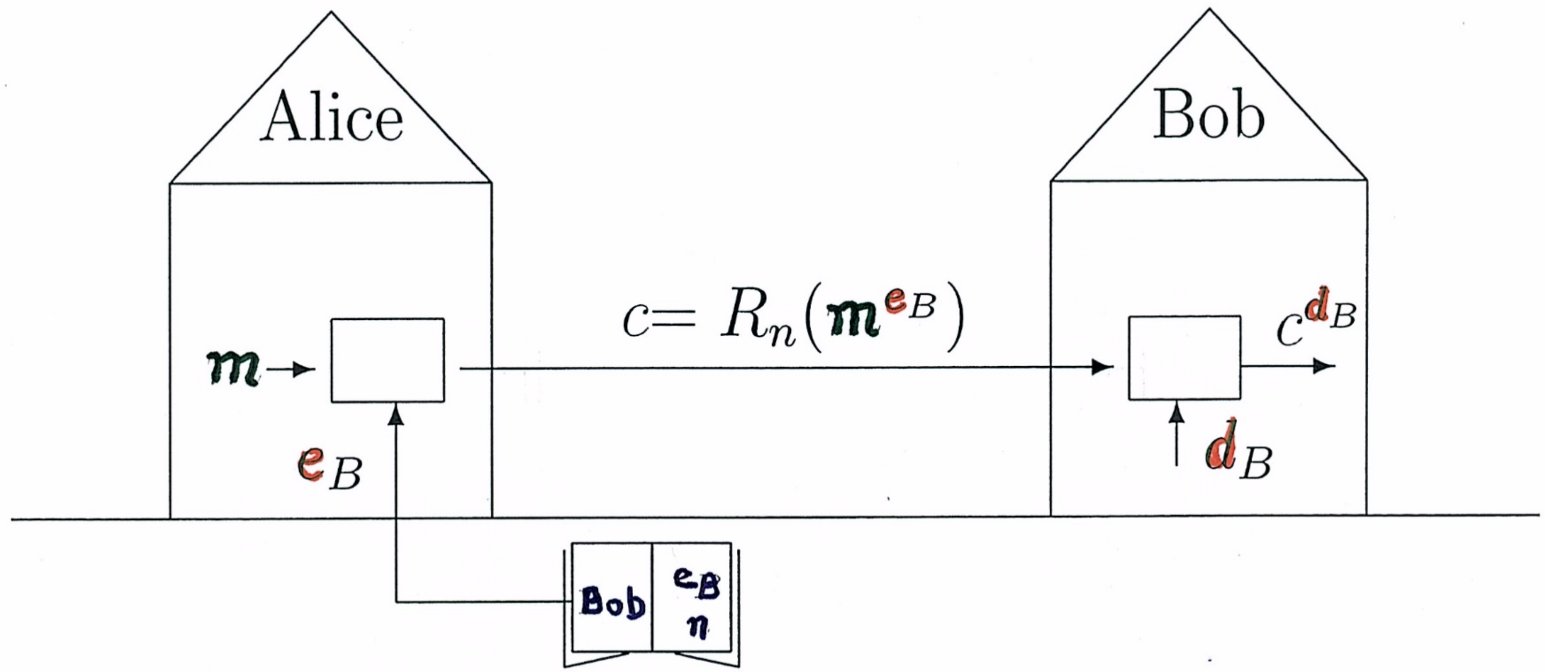
both are public key cryptosystems.

RSA

RSA

In the RSA-System, each participant, e.g. Bob, has as
private key a number d_B and as
public key a pair (e_B, n) where $n=pq$ is a pseudo-prime (i.e. a product of two large primes) and $e_B d_B \equiv 1 \pmod{(p-1)(q-1)}$.

In python: Rsa= lambda m: m**e_B%n



The security of the system depends as well on the possbility of factorising n. For such an attack, there exist many algorithms, e.g.

-the algorithm of Fermat ,

-the quadratic sieve (QS) (Carl Pomerance)

-the number field sieve (NFS)

A direct attack uses the

- Continued fraction method (CFRAC) (M.J.Wiener)

which gives d from e/n if $d<1/3 \cdot n^{1/4}$

The Quadratic Sieve and other sieves:
Find a,b with

$$a^2 \equiv b^2 \pmod{n} \quad \text{and} \quad a \not\equiv \pm b \pmod{n}$$

Then we have:
n divides (a-b)(a+b), but not (a-b) and not (a+b).
Therefore: gcd(a±b,n) are non-trivial divisors of n.

„Ron was wrong, Whit is right“

was the provocative title of a paper of Arjen K. Lenstra, Thorsten Kleinjung et al. who, 2009, had collected several millions of RSA-keys. They could break over 12.000 keys.

„Ron was wrong, Whit is right“
was the provocative title of a paper of Arjen K. Lenstra, Thorsten Kleinjung et al. who, 2009, had collected several Millions of RSA-keys. They could break over 12.000 keys.

Here Ron means Ron Rivest and RSA and
Whit stands for Whit Diffie and Martin Hellman (DSA and ECC).
The main mistake made in key creation was the
Repeated use of primes in several pseudoprimes
such that one could break them by determining the gcd.

The (later so called) number RSA-129
(with 129 decimal digits, 476 binary digits)
which was presented by Martin Gardner 1976 (and believed by Ron Rivest to resist quadrillion years) was factorized 1994 by 600 participants with 10^{17} operations using a version of the quadratic sieve.

RSA-129 =
1143816257578888676692357799761466120102182967212423625625618429357069352457338978305971235639587050589890
75147599290026879543541

The (later so called) number RSA-129
(with 129 decimal digits, 476 binary digits)
which was presented by Martin Gardner 1976 (and believed by Ron Rivest to resist quadrillion years) was factorized 1994 by 600 participants with 10^{17} operations using a version of the quadratic sieve.

RSA-129 =
114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541 =
3490529510847650949147849619903898133417764638493387843990820577*32769132993266709549961988190834461413177642967992942539798288533

To test the security of 'semiprimes', the
RSA-Factoring-Challenge,
a competition, was put forward by the RSA Laboratories;
it ended 2007 when Jens Franke (Bonn) et.al. had factorised

RSA-576 (2003; with 576 binary digits, 174 decimal digits),

RSA-640 (2005; with 193 decimal digits) and, together with Thorsten Kleinjung, a

1039-Bit long Mersenne number (which was not part of the challenge).

Predicted Approximate costs for breaking the actually used RSA-1024 and RSA-2048:

Continued fraction: 2^{120}, 2^{170} operations

Quadratic sieve: 2^{100}, 2^{150}

Numberfield sieve: 2^{80}, 2^{112}

→ Big decrease

(according to Tanja Lange and Daniel J.Bernstein: ECCHacks on YouTube)

The European Union Agency for Network and Information Security (ENISA) recommends

for RSA for the length of n

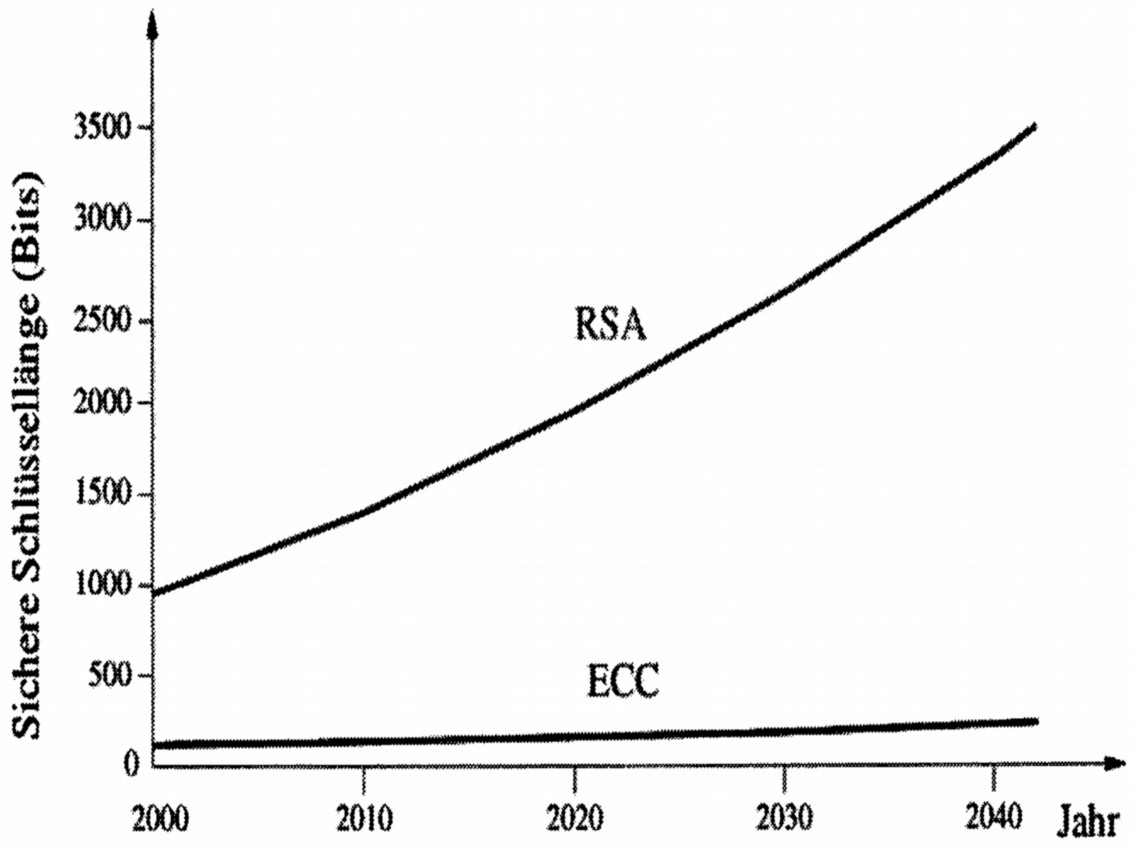
3072 Bits for medium term,

15.360 Bits for long term security,

for ECC for the greatest prime divisor of the group order

160 Bit for medium term and

512 Bit for long term security.



Forecast for the length of secure keys of RSA and ECC

by A.Lenstra and E.Verheul (see CrypTool-Scipt)

Attacks to the LOG-problem: e.g.

Babystep-Giantstep-algorithm for determining $a = \log_g(A)$ (i.e. $A = g^a$).

Let $m = o(g)$ and w with $w-1 < \sqrt{m} \le w$; then $a = w \cdot j + r$ and $A = g^{wj} g^r$

$$A \cdot g^{-r} = (g^w)^j.$$

Attacks to the LOG-problem: e.g.

Babystep-Giantstep-algorithm for determining $a = \log_g(A)$ (i.e. $A = g^a$).

Let $m = o(g)$ and w with $w-1 < \sqrt{m} \leq w$; then $a = w \cdot j + r$ and $A = g^{wj} g^r$

$$A \cdot g^{-r} = (g^w)^j.$$

Attacks to the LOG-problem: e.g.

Babystep-Giantstep-algorithm for determining $a = \log_g(A)$ (i.e. $A = g^a$).

Let $m = o(g)$ and w with $w-1 < \sqrt{m} \leq w$; then $a = w \cdot j + r$ and $A = g^{wj} g^r$

$$A \cdot g^{-r} = (g^w)^j.$$

Compare:

Babystep list $\{A \cdot (g^{-1})^r \mid r = 0, \ldots, w-1\}$

with the

Giantstep list $\{1, g^w, (g^w)^2, \ldots, (g^w)^{w-1}\}$

(which is not dependend from a).

MAN IN THE MIDDLE

Index-Calculus-algorithm to find $\log_g b$

Try to represent g^z in $G=\langle g\rangle$ for random z with a factor-basis $S=\{a_1,..,a_t\}$, i.e. $g^z=a_1^{s_1}\cdots a_t^{s_t}$

giving $z \equiv s_1\log_g a_1+...+ s_t\log_g a_t \pmod{n}$

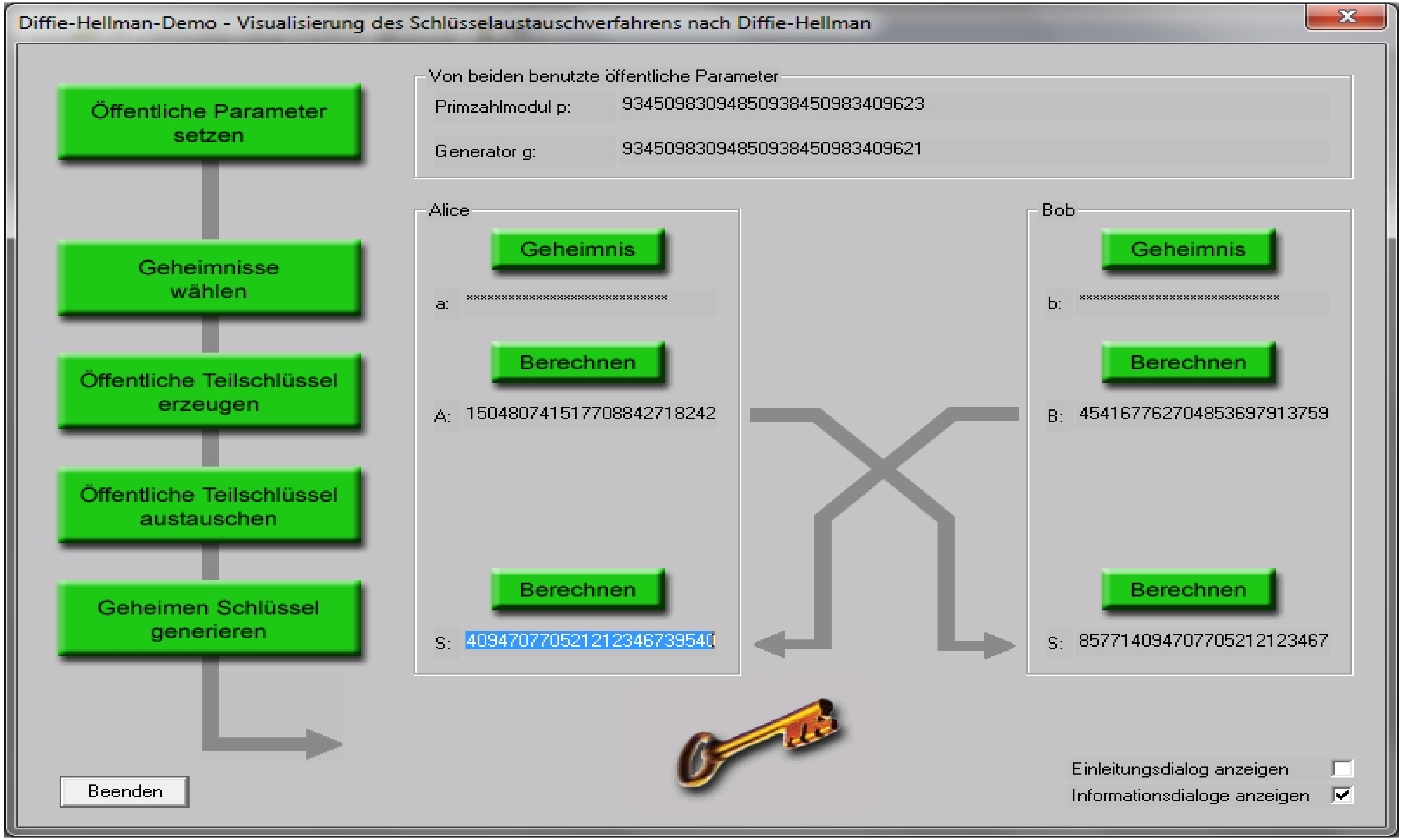
Repeat to get $\log_g a_i$ as solutions of a system of linear equations.

Try to find s with $g^s b=a_1^{b_1}\cdots a_t^{b_t}$; from that one gets

$\log_g b=b_1 \log_g a_1+..-s \pmod{n}$

Other algorithm: Pohlig-Hellman, Pollard-Rho, number field sieve, function field sieve

Visualisation of the key exchange system by Diffie and Hellmann with CrypTool 1



Generator g

Secret random number:
a of Alice,
b of Bob
Public A and B with
$A=g^a$ und $B=g^b$

Common secret:
$S=g^{ab}=A^b=B^a$

ElGamal crypto-system

Private key a

Public key (g,A) (with $A:=g^a$)

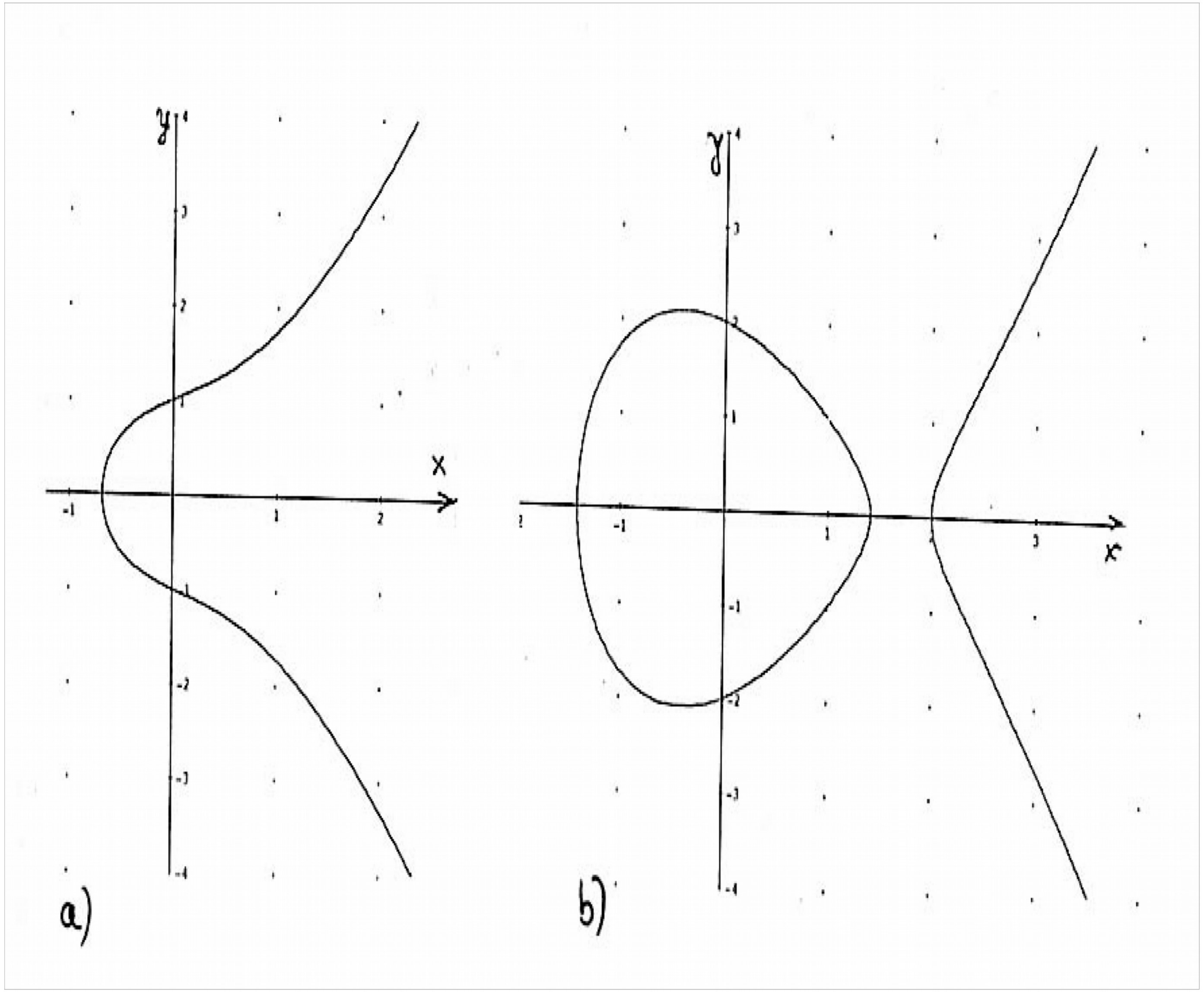
Encoding of plain text m $E(m):=(g^k, m\cdot A^k)$

for a randomly chosen k

to protect a: $A^k=(g^a)^k=(g^k)^a$

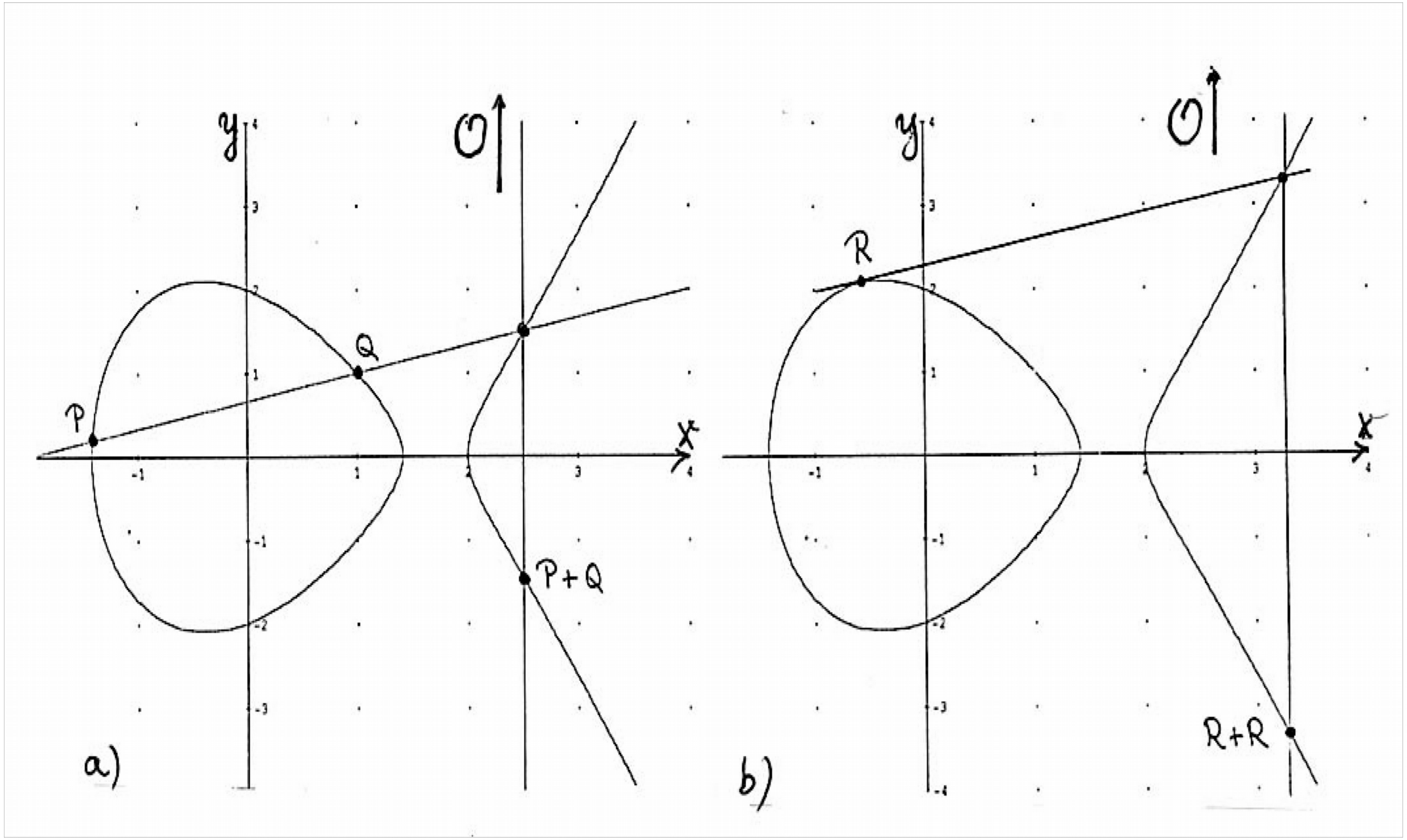
Decoding $D(x,y):=(x^a)^{-1}\cdot y$

Verification $D(g^k, m\cdot A^k)=((g^k)^a)^{-1} mA^k = g^{-ak} g^{ak} m = m.$

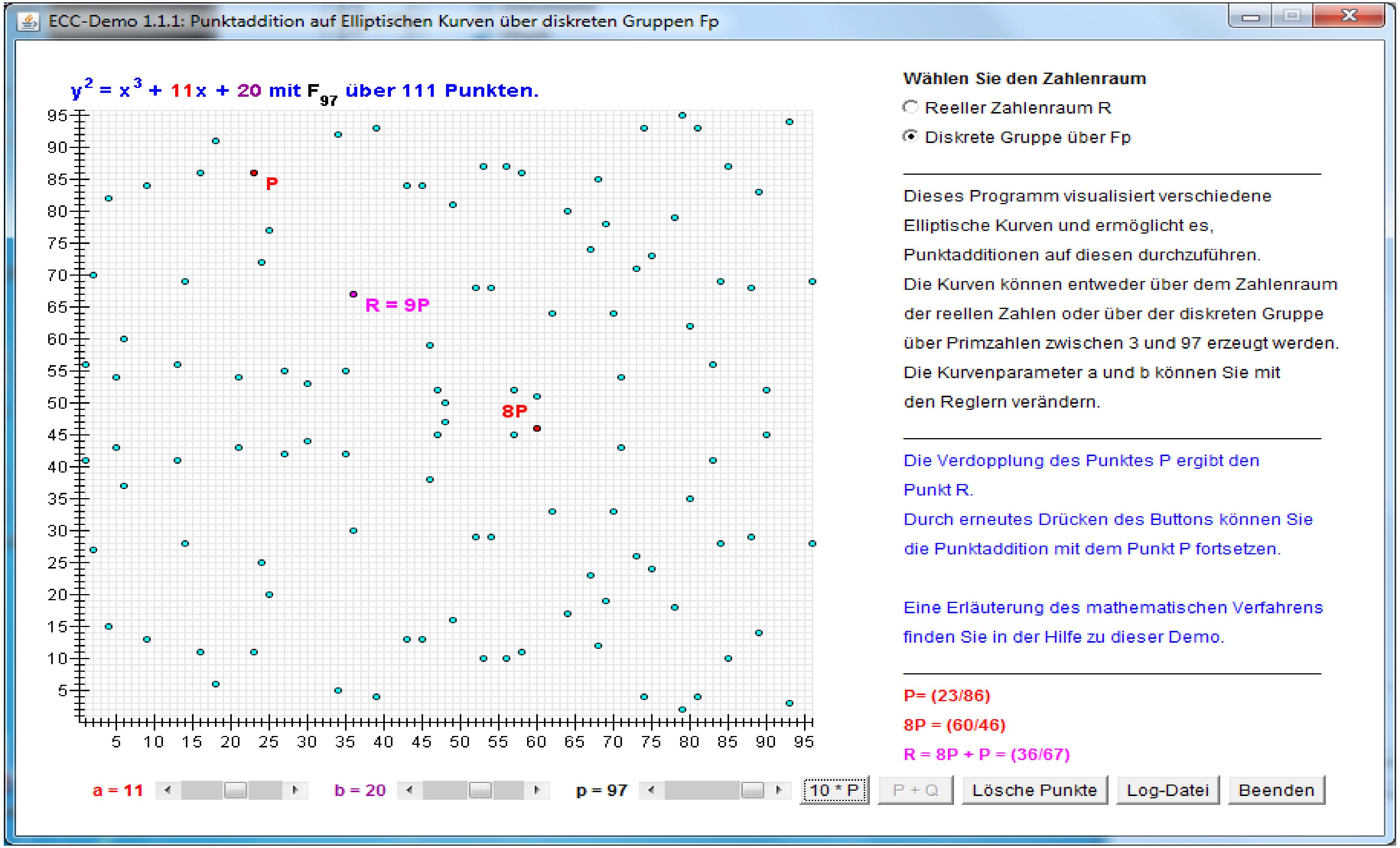


Examples of real elliptic curves with Weierstrass equation

$y^2=x^3+bx+c$



Addition on an elliptic curve



Elliptic curve over Z_{97} (generated with CrypTool by B.Esslinger)

Daniel Bernstein (Chicago and Eindhoven) and
Tanja Lange (Eindhoven) recommend elliptic curves
with other equations
(because of easier implementation and
possible back doors
in the curves recommended by NIST)

http://ecchacks.cr.yp.to/.
https://www.youtube.com/watch?v=l6jTFxQaUJA}
(Video)

E.g.:
It is known that the
„Dual Elliptic Curve Deterministic Random Number Generator“
has a back door:

If the points are randomly chosen, the x-coordinates are not randomly distributed.

Europol chief warns on computer encryption (BBC 29 March 2015)

„Hidden areas of the internet and encrypted communications make it harder to monitor terror suspects“,
warns Europol's Rob Wainwright.

Other types of elliptic curves:

Edwards curves with equation

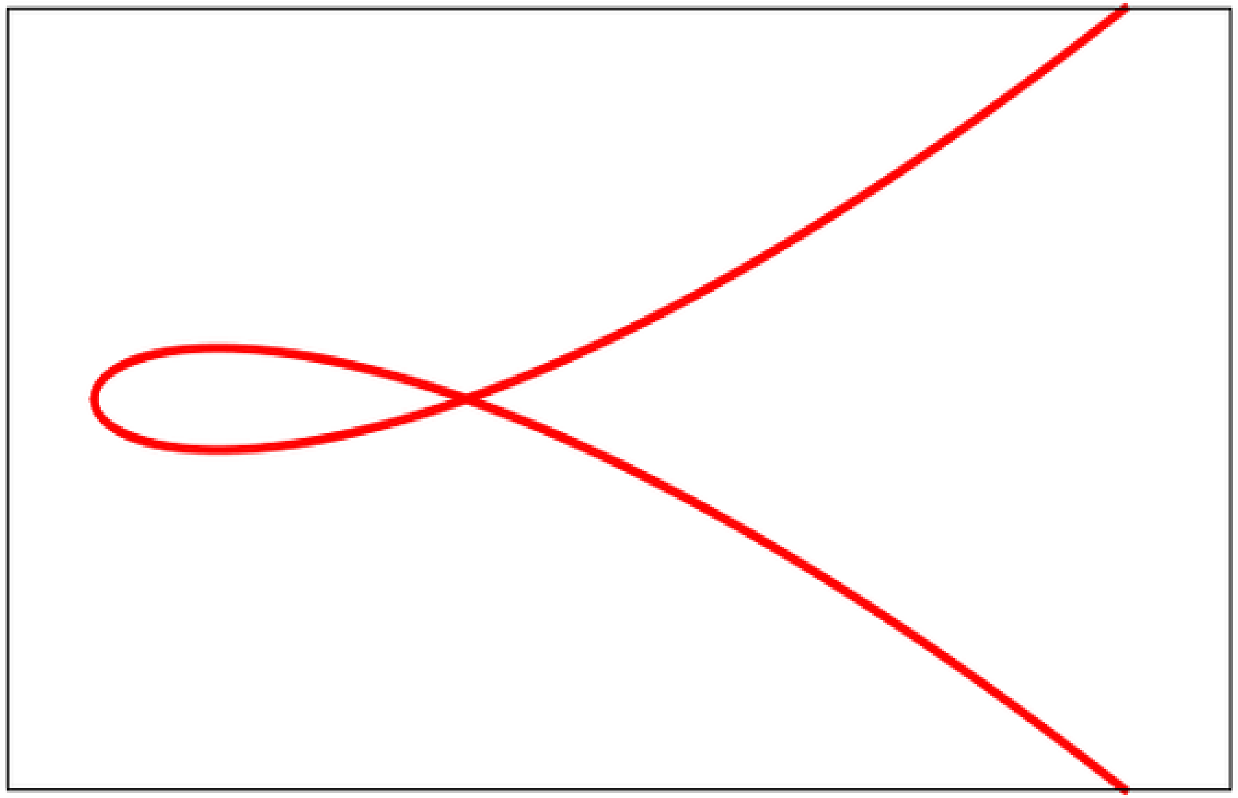
$ax^2+y^2=1+dx^2y^2$ (with a non-square d)

Montgommery curves

$By^2=x^3+Ax^2+x$

with special case Bernsteins elliptic curve25519

(used in OpenSSH, GnuPG)



$$y^2=x^3+486662x^2+x$$

Bernstein's elliptic curve

http://www.heise.de/security/meldung/Konkurrenz-fuer-die-NIST-Bernsteins-Elliptische-Kurven-auf-dem-Weg-zum-Standard-2560881.html

Bernstein's elliptic curve

E: $y^2=x^3+Ax^2+x$

$p=2^{255}-19$

A with A^2-4 not a square mod p, e.g.

$A=486662$

„Curve 25519-Function“: $\mathbb{F}_p$-restricted x-coordinate scalar multiplication on $E(\mathbb{F}_{p^2})$

Post Quantum Computer

Research on lattices, error corrrecting codes, TSP (time stamp protocols), Hash based procedures

Thank you for your attention!