Cryptographic applications of curves over finite fields

Tanja Lange

Codingtheory and Cryptology

Technische Universiteit Eindhoven

tanja@hyperelliptic.org

15.02.2007

Classic Cryptography

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  G

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  GH

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  GHD

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters ($\mathtt{A} \longrightarrow \mathtt{D}$) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  GHDU

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  GHDU FOHRSDWUD,

Caesars' Cipher

- The idea of encrypting by shifting the alphabet by 3 letters (A $\longrightarrow$ D) is often credited to Julius Caesar.

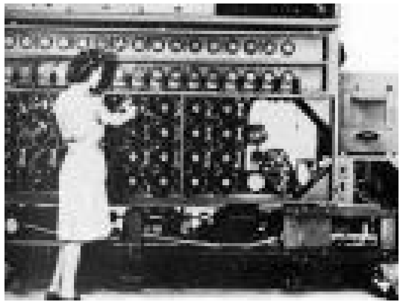
  XYZABCDEFGHIJKLMNOPQRSTUVW
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  DEAR CLEOPATRA,
  GHDU FOHRSDWUD,

- Obvious generalization: shift by chosen number of letters (known to sender and receiver and kept secret)
- Very easy to break from seeing the pattern of words.
- Also without spaces very simple analysis through statistics; all natural languages have non-uniform distribution of letters/symbols.

Enigma

- During the second world war the Germans used Enigma for secret transmission of messages.
- Hard work and ingenuous ideas lead to a complete break of the system. This was facilitated through incorrect usage, known plaintext-ciphertext pairs and available details on the hardware.
- 1938 Polish researchers had managed to break the standard version of Enigma. (The submarine one had an extra rotor.)
- Researchers in Bletchley Park (UK) built Colossus, a code-cracking machine which broke the "unbreakable" improved Enigma within one hour per key.



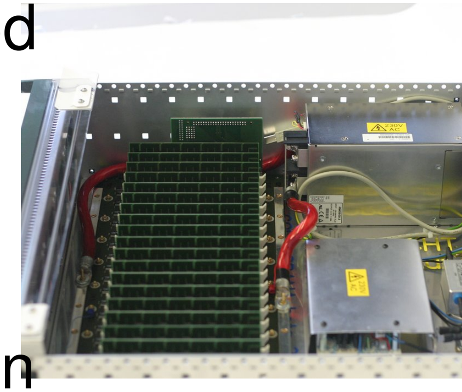
Modern Systems I

- 1973-74 IBM designed DES, the Data Encryption Standard, which was standardized 1977. It encrypts data using a secret key which must be known to the sender and receiver.
- Already 1977 Diffie and Hellman criticized DES because the key length was only 56 bits and that seemed within reach for brute force attacks in the not too far away future.

- 1998 Paul Kocher built the DES cracker a special purpose machine to break DES. The DES cracker was funded by the Free Software Foundation with 250,000 USD. It took 3 days to complete the RSA Laboratory's "DES Challenge II" which came with 10,000 USD cash prize.

Modern Systems II

- 2006 Copacobana was built by Christof Paar (Bochum) and his students as a highly optimized brute force engine. Built for only 10,000 USD it breaks DES in 9 days.



- Triple DES offers a bit more security.
- Current standard AES, the Advanced Encryption Standard, is unbroken and used in normal banking applications etc.
- Problem: key distribution! Every user needs to have a shared secret with every partner he communicates with. Banking cards should be usable also at ATMs of other banks.

Public-key cryptography

- Invented by Diffie and Hellman in 1976.
- Based on the observation that it can be fast to evaluate a function while the inverse operation is computationally hard.
- Current examples of such problems are factorization of integers, computing discrete logarithms in a group of known order, finding shortest vectors in lattices, solving systems of multivariate polynomials.
- Diffie and Hellman suggested a key exchange protocol. In the multiplicative group of a finite field $\mathbb{F}_q$ they take a subgroup of prime order $\ell | q - 1$ and a generator g of this order.
- In general, let G be a finite group and consider the group generated by $P \in G$.

Diffie-Hellman Key exchange

Alice		**Bob**
1. secretly generates $a < \lvert\langle P\rangle\rvert$		1. secretly generates $b < \lvert\langle P\rangle\rvert$
2. computes $Q_1 = [a]P$		2. computes $Q_2 = [b]P$
3. transmits Q_1		3. transmits Q_2
4. computes		4. computes
$[a]Q_2$	$= [ab]P =$	$[b] \cdot Q_1$

Common Key: the group element $k = [(ab)]P \in G$ can be used in symmetric encryption.

Cryptanalysis

- For any group G (finite fields, points on elliptic curve over finite field, ...) there are attacks running in $O(\sqrt{|G|})$. Such attacks are called generic attacks since they apply to every group.
- Try to find groups in which this is the running time of the best known attack.
- There are index calculus attacks for the multiplicative group of finite fields which have subexponential running time.
- Also for curves of genus ≥ 4 the generalizations of the index calculus attack are faster than generic attacks.

Interesting systems

- Elliptic curves over finite fields, hyperelliptic curves of genus 2 or 3 over finite fields.
- Weil's theorem states

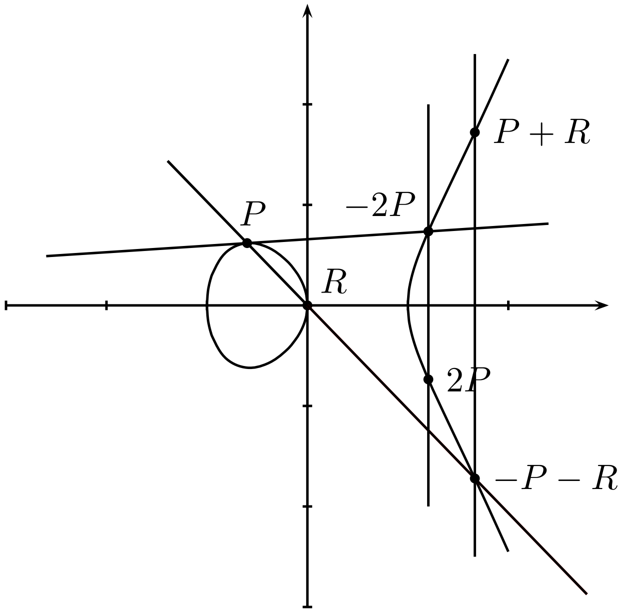
$$(\sqrt{q}-1)^{2g} \leq |\mathcal{J}_C(\mathbb{F}_q)| \leq (\sqrt{q}+1)^{2g},$$

so $|\mathcal{J}_C(\mathbb{F}_q)| = q^g + O\left(g^{(2g-1)/2}\right)$.

- My research is about
  - Fast arithmetic in the Jacobian of curves.
  - Efficient computation of the Tate pairing.
  - Cryptanalysis.
  - Finding secure protocols.

Group Law (q odd)

$$E : y^2 = x^3 + a_4 x + a_6, \; a_i \in \mathbb{F}_q$$



for $(x_1, y_1) \neq (x_2, -y_2)$:

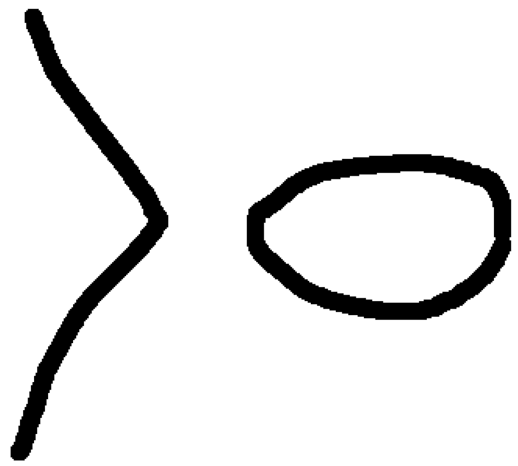
$$\begin{aligned}(x_1, y_1) + (x_2, y_2) &= \\ &= (x_3, y_3) = \\ &= (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1),\end{aligned}$$

where

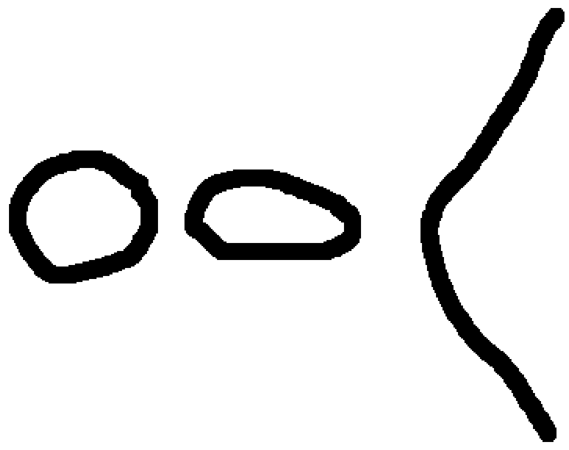
$$\lambda = \begin{cases} (y_2 - y_1)/(x_2 - x_1) & \text{if } x_1 \neq x_2, \\ (3x_1^2 + a_4)/(2y_1) & \text{else.} \end{cases}$$

Addition: 1 I, 2M, 1S
Doubling: 1 I, 2M, 2S.

Elliptic vs. Hyperelliptic



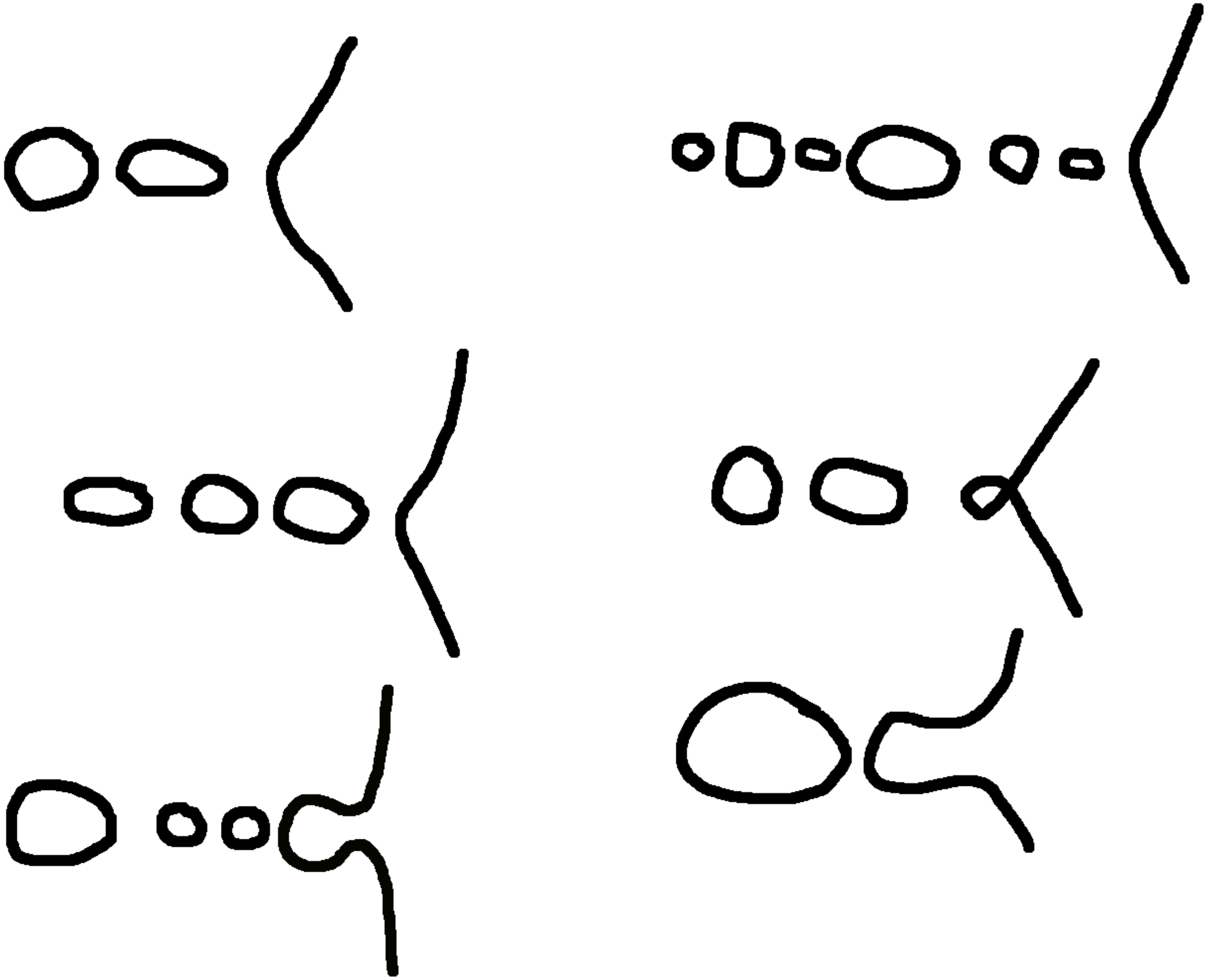
$g = 1$



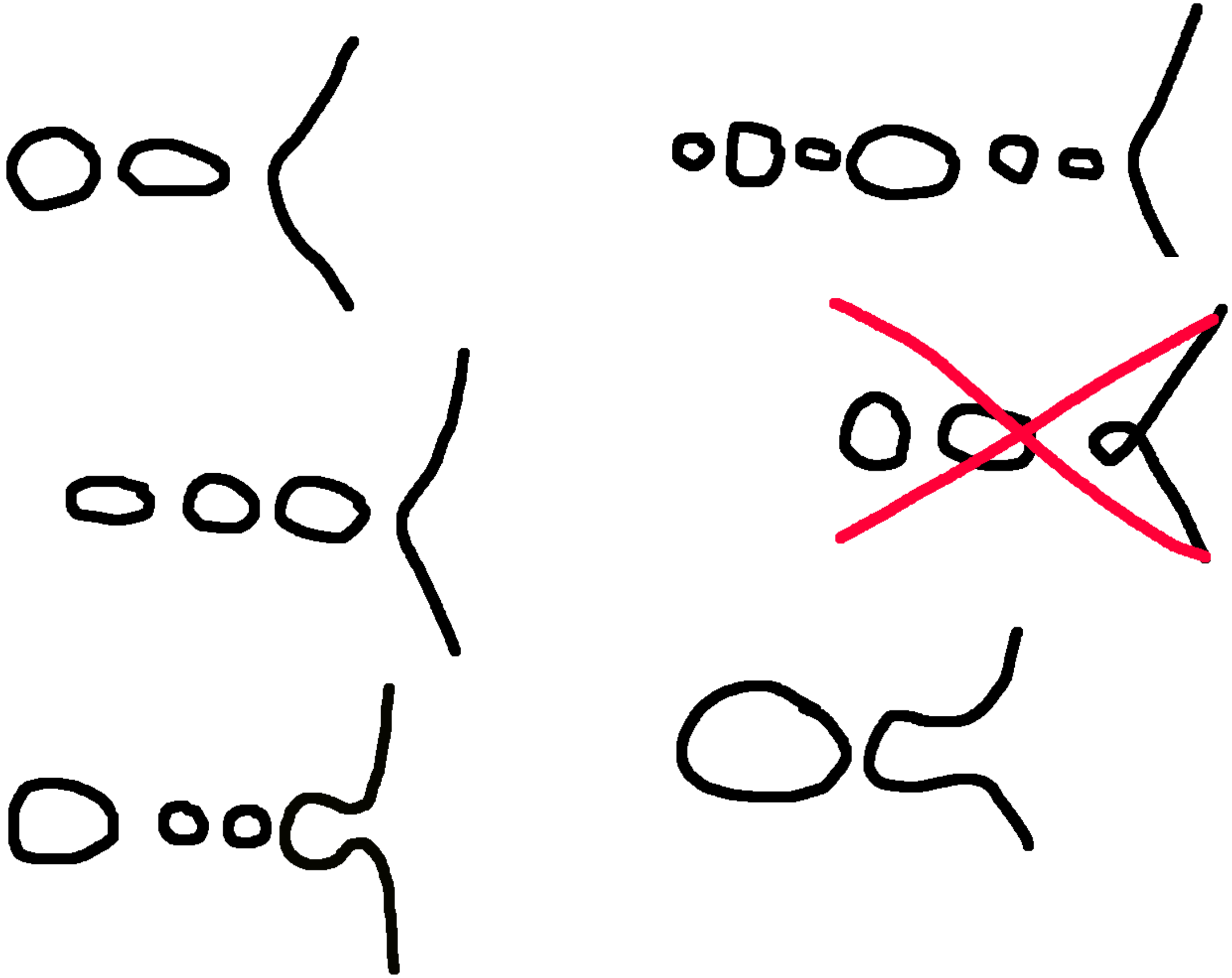
$g = 2$

(… already after some transformations …)

. . . and friends



. . . and friends



. . . and friends

Affine equation of hyperelliptic curve of genus g (with $\mathbb{F}_q$-rational Weierstraß-point at infinity)

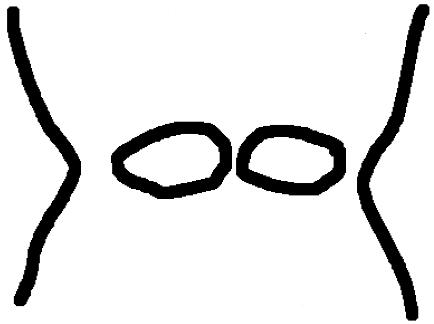
$$C : y^2 + h(x)y = f(x)$$

$h(x), f(x) \in \mathbb{F}_q[x]$, f monic, $\deg f = 2g + 1, \deg h \leq g$
non singular, i. e. in no point $(a, b) \in C/\overline{\mathbb{F}_q}$ over the algebraic closure $\overline{\mathbb{F}_q}$ we have both partial derivatives vanishing, i.e.

$2b + h(a) = 0$ and $h'(a)b - f'(a) = 0$.

Otherwise (a, b) is called a singular point.

(There are also the real quadratic function fields with $\deg(f) = 2g + 2$ . . .



. . . and friends

Affine equation of hyperelliptic curve of genus g (with $\mathbb{F}_q$-rational Weierstraß-point at infinity)

$$C : y^2 + h(x)y = f(x)$$

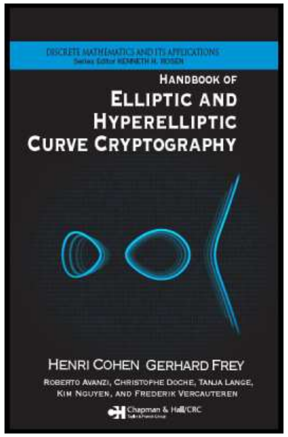
$h(x), f(x) \in \mathbb{F}_q[x]$, f monic, $\deg f = 2g + 1, \deg h \leq g$
non singular, i. e. in no point $(a, b) \in C/\overline{\mathbb{F}_q}$ over the algebraic closure $\overline{\mathbb{F}_q}$ we have both partial derivatives vanishing, i.e.
$2b + h(a) = 0$ and $h'(a)b - f'(a) = 0$.

Otherwise (a, b) is called a singular point.

(There are also the real quadratic function fields with $\deg(f) = 2g + 2$ . . .
. . . the step-brothers . . .)

Divisor class group

For details see THE book.
The divisor class group of degree zero is the factor group of the group of divisors of degree zero Div_C^0 modulo the principal divisors.



$$\mathrm{Pic}_C^0 = \mathrm{Div}_C^0 / \mathrm{Princ}_C.$$

Each divisor class has a unique reduced representative

$$D = \sum_{\substack{i=1 \\ P_i \in C(\overline{\mathbb{F}_q}) \setminus \{P_\infty\}}}^{m} P_i - mP_\infty$$

with $P_i \neq P_\infty$, $P_i \neq -P_j$ for $i \neq j$ and $m \leq g$.

Representation

- Idea: use polynomials to represent divisors, ignore P_∞ – multiplicity dictated by affine part.
- Let D be semi-reduced $D = \sum_{i=1}^m P_i - mP_\infty$ with $P_i = (x_i, y_i)$. Put

  $$u(x) = \prod_{i=1}^{m}(x - x_i) \text{ and define } v \text{ by } v(x_i) = y_i \text{ with multiplicity,}$$

  i.e. if P_i appears n_i times then

  $$\left(\frac{d}{dx}\right)^j \left[v(x)^2 + v(x)h(x) - f(x)\right]_{|x=x_i} = 0, \quad \text{for } 0 \le j \le n_i - 1.$$

- Corresponds to ideal $\langle u(x), y - v(x) \rangle$ in coordinate ring of curve.

Arithmetic on hyperelliptic curves

Composition & Reduction (Cantor/Koblitz)

`IN:` $D_1 = [u_1, v_1], D_2 = [u_2, v_2], C : y^2 + h(x)y = f(x)$
`OUT:` $D = [u, v]$ reduced with $D \sim D_1 + D_2$

1. compute $d_1 = \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2. compute $d = \gcd(d_1, v_1 + v_2 + h) = c_1 d_1 + c_2(v_1 + v_2 + h)$
3. let $s_1 = c_1 e_1, s_2 = c_1 e_2, s_3 = c_2$
4. $u = \frac{u_1 u_2}{d^2} \quad v = \frac{s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \bmod u$

This result $[u, v]$ corresponds to a semireduced divisor.

Arithmetic on hyperelliptic curves

Composition & Reduction (Cantor/Koblitz)

`IN:` $D_1 = [u_1, v_1], D_2 = [u_2, v_2], C : y^2 + h(x)y = f(x)$
`OUT:` $D = [u, v]$ reduced with $D \sim D_1 + D_2$

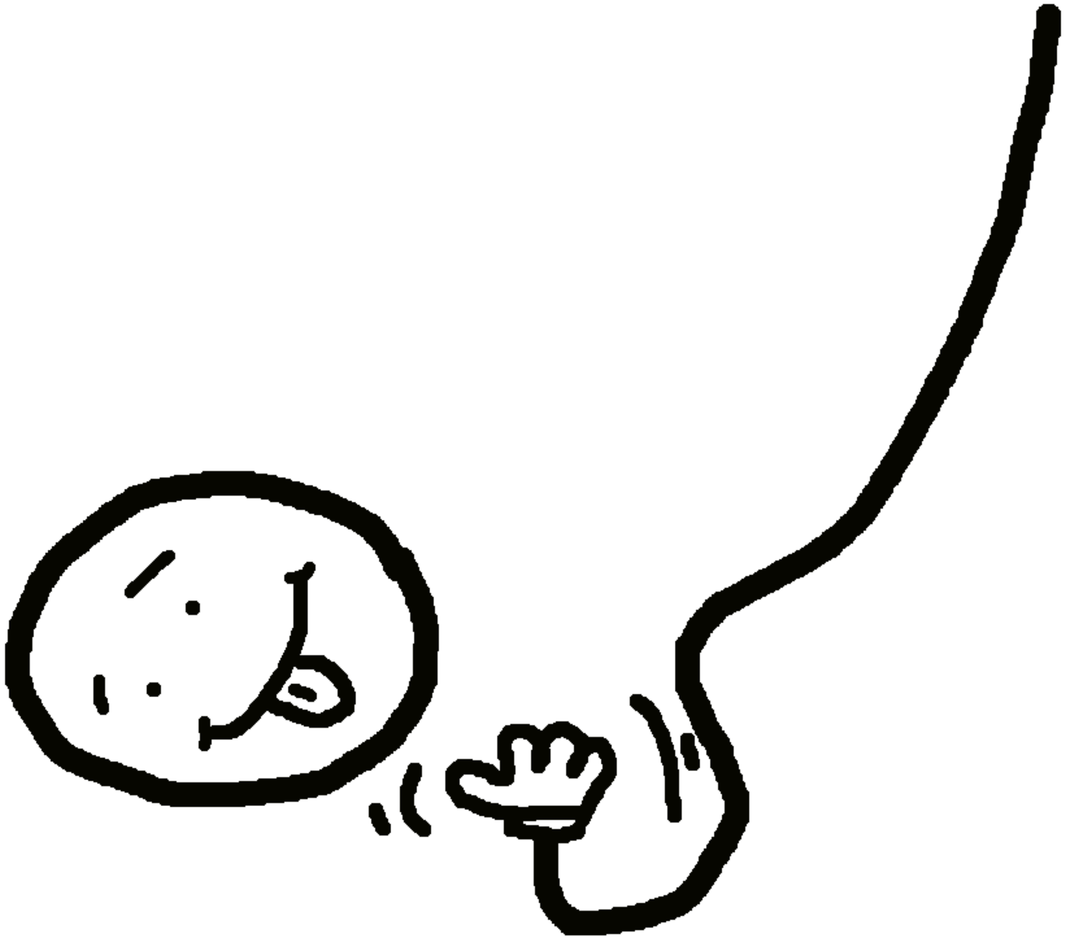
1. compute $d_1 = \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2. compute $d = \gcd(d_1, v_1 + v_2 + h) = c_1 d_1 + c_2(v_1 + v_2 + h)$
3. let $s_1 = c_1 e_1, s_2 = c_1 e_2, s_3 = c_2$
4. $u = \frac{u_1 u_2}{d^2} \quad v = \frac{s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \bmod u$
5. let $u' = \frac{f - vh - v^2}{u} \quad v' = (-h - v) \bmod u'$
6. if $\deg u' > g$ put $u := u', v := v'$ goto step 5
7. make u monic

This result $[u, v]$ corresponds to a reduced divisor.

First implementation results

Nigel P. Smart: On the Performance of Hyperelliptic Cryptosystems. EUROCRYPT 1999: 165-175: Performance of hyperelliptic is really bad!

Spotlight on elliptic curve



More destruction results . . .

Index calculus attacks on HECC

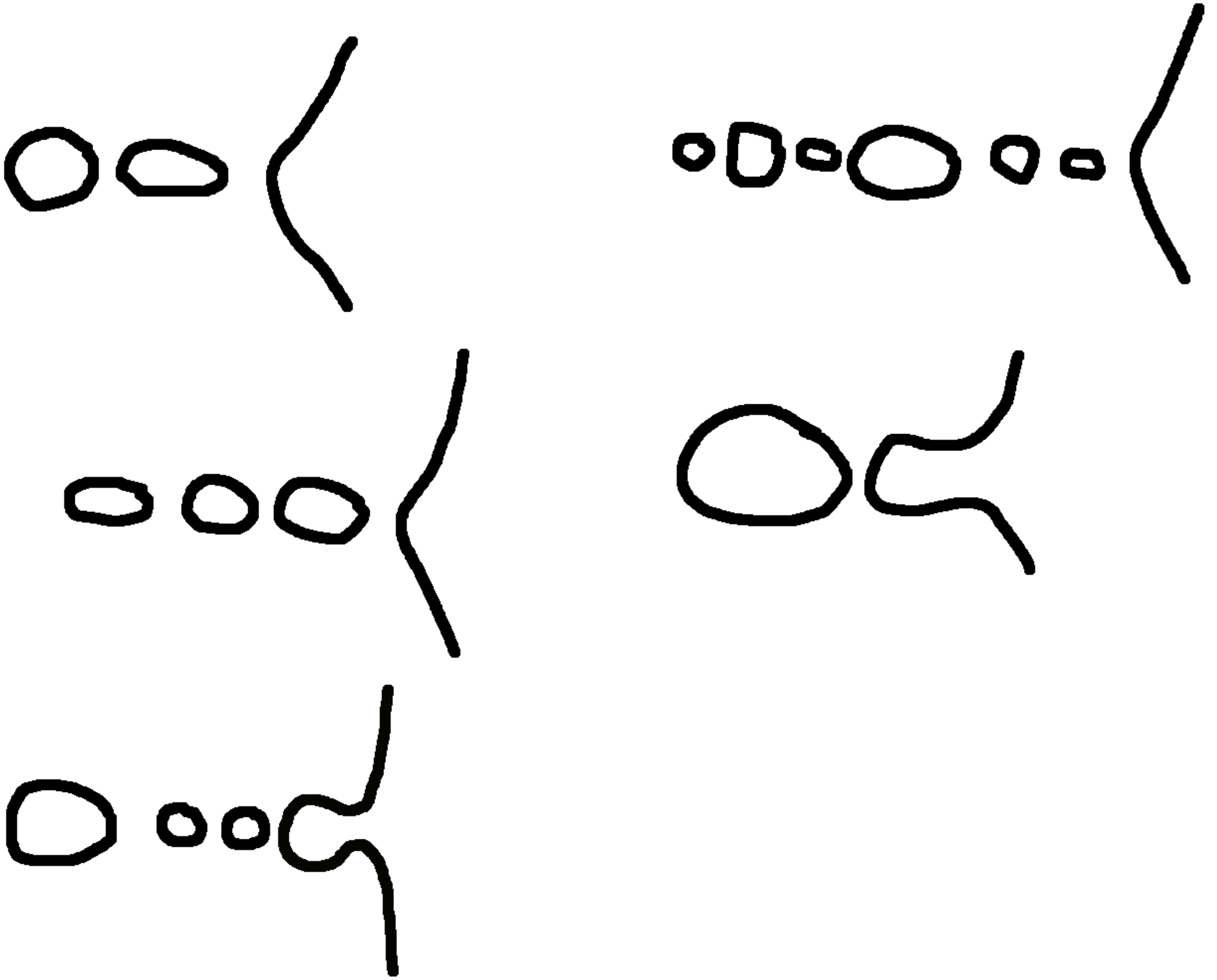
- Adleman, De Marrais, and Huang, A subexponential algorithm over hyperelliptic curves of large genus over GF(q), 1999.
- Gaudry, An algorithm for solving the discrete logarithm problem on hyperelliptic curves, 2000.
- Enge, Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time, 2003.
- Enge and Gaudry, A general framework for subexponential discrete logarithm algorithms, 2002.

… and more …

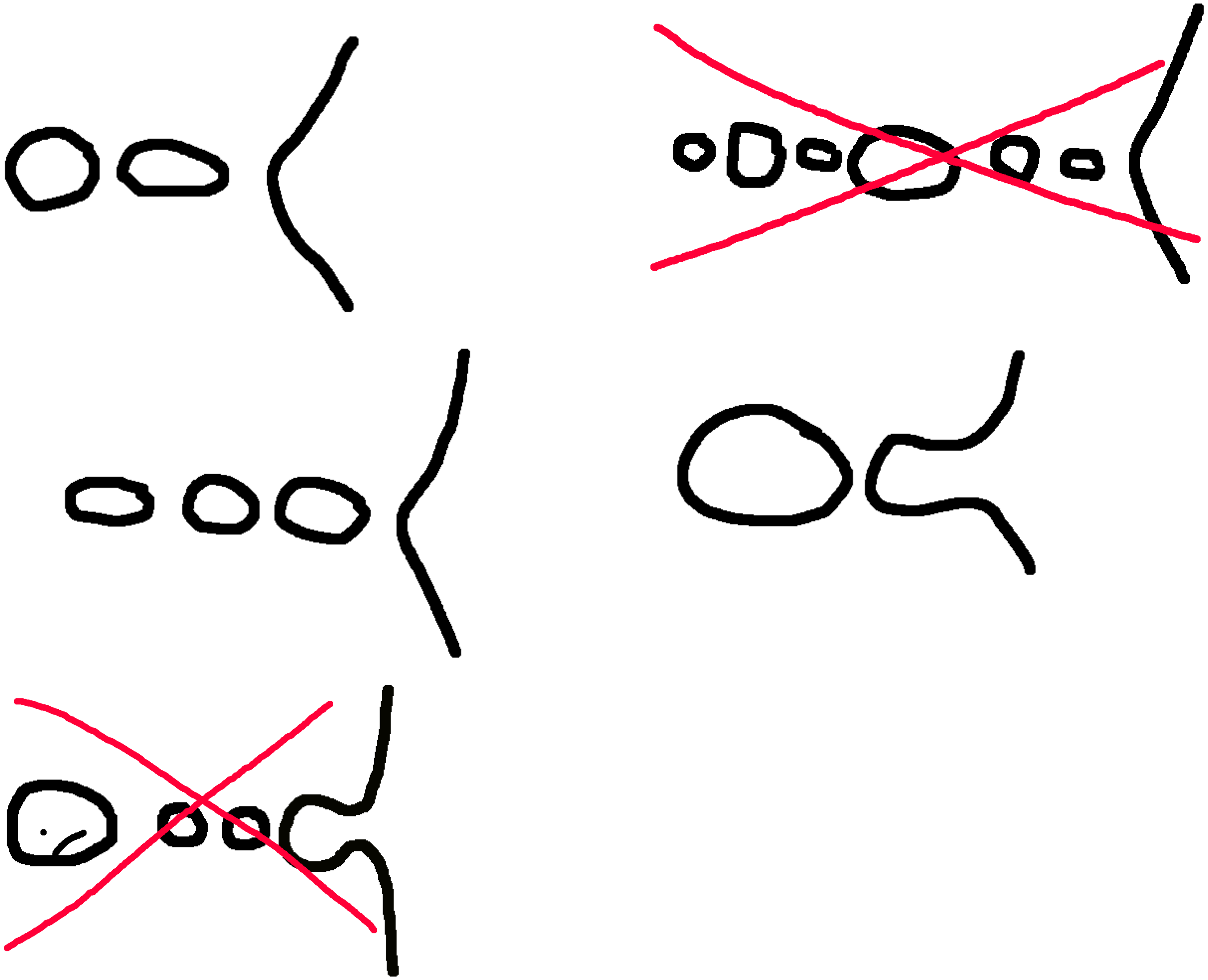
- Thériault. Index calculus attack for hyperelliptic curves of small genus. 2003.
- Gaudry, Thériault, and Thomé. A double large prime variation for small genus hyperelliptic index calculus. 2005.
- Diem. Index calculus with double large prime variation in class groups of plane curves of small degree, 2006.
- Gaudry, Thomé, Thériault, and Diem. A double large prime variation for small genus hyperelliptic index calculus. 2007? (accepted).

This rules out genus $g \geq 4$ and scratches $g = 3$.

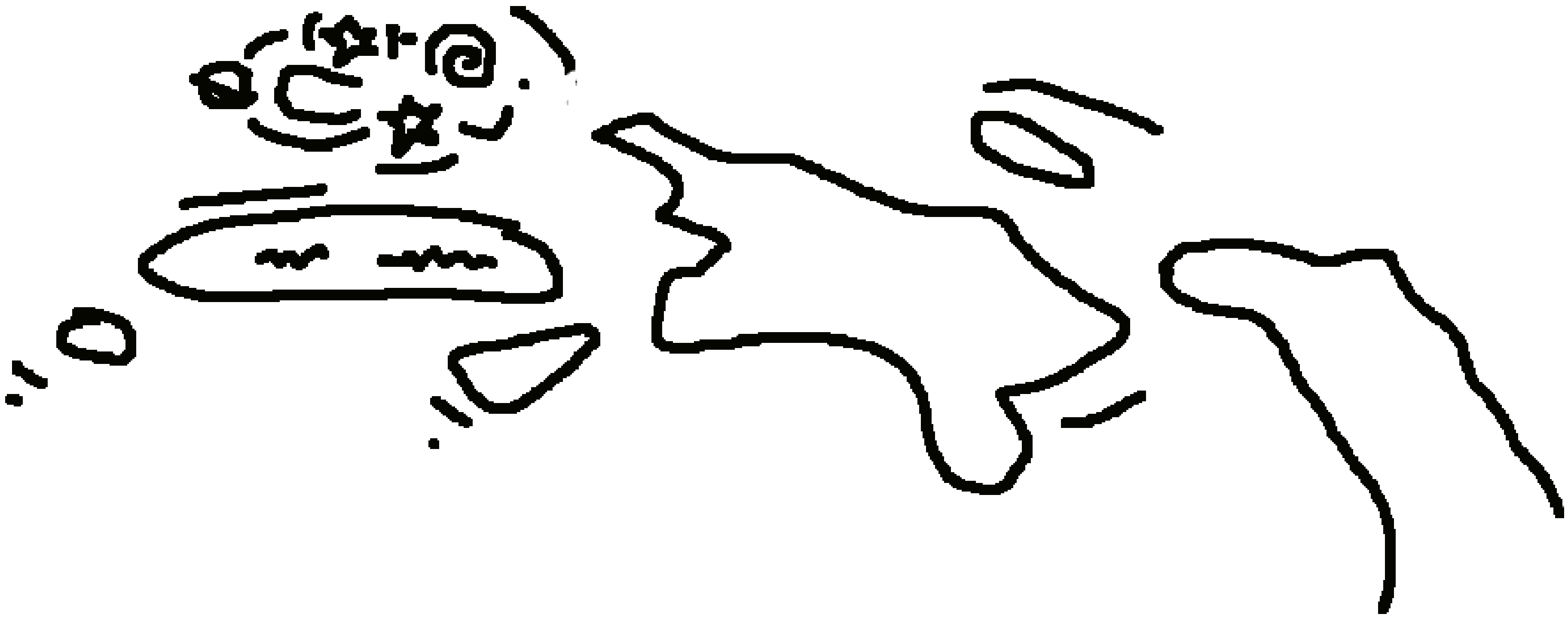
Fewer friends



Fewer friends

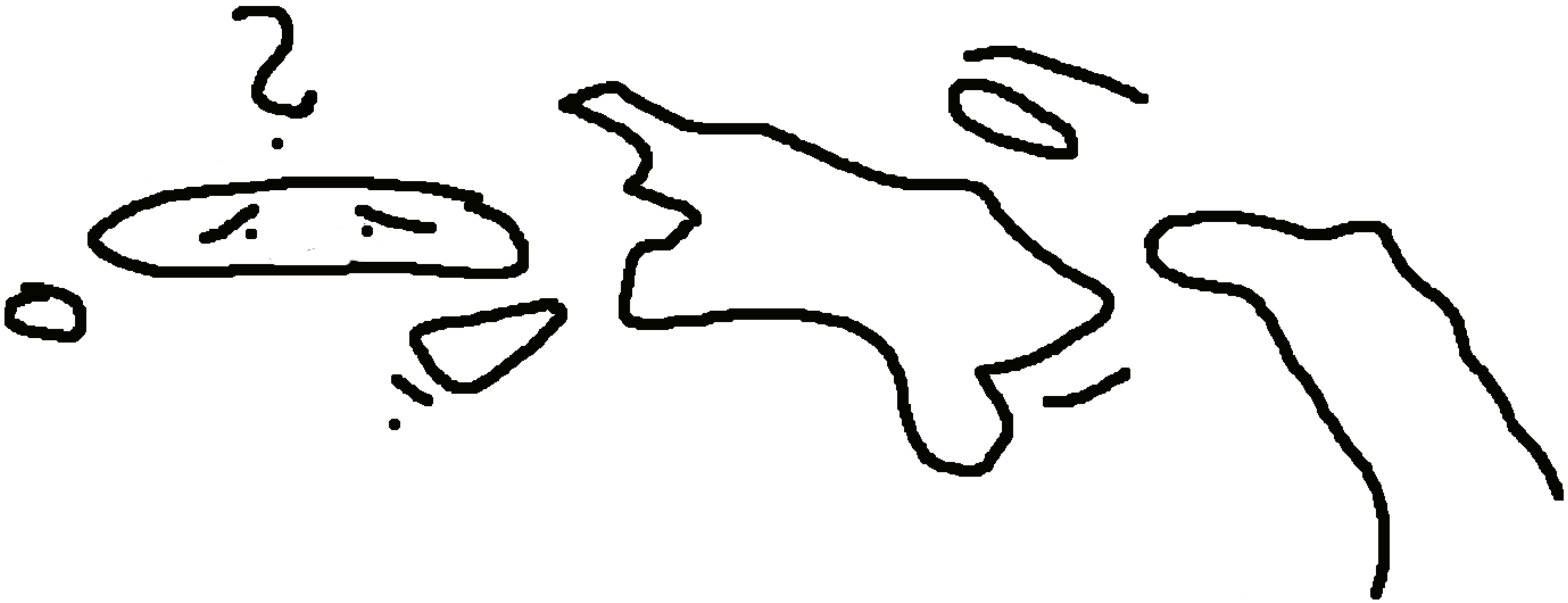


Game over?



. . . slow, most friends insecure, . . .

Any hope?



Maybe not sooo slow

- Robert Harley (while counting points with Pierrick Gaudry) is annoyed about slow $g = 2$ operations We don't use Cantor for $g = 1$ why should we for $g = 2$?
- Develops first "explicit formulae" for genus 2 curves, close to no explanation.
- Main speed-ups by avoiding computation of unused coefficients.
- Explicit use of Karatsuba multiplication, reduction and resultants for inversion modulo polynomials.
- Requires case distinction.

Explicit Arithmetic $g = 2$ and $g = 3$

2000	2	Harley (odd char.)	$\Rightarrow$ 2 inversions
2001	2	L. (arbitrary char.)	
2001	2	Matsuo, Chao, Tsujii (faster)	
2002	2	Miyamoto, Doi, Matsuo, Chao, Tsujii	$\Rightarrow$ 1 inv.
2002	2	Takahashi	
2002	2	L. (arbitrary char.)	
2002	2	Sugizaki, Matsuo, Chao, Tsujii (even)	
2004	2	L., Stevens (even char., all cases)	
2002	3	Kuroki, Gonda, Matsuo, Chao, Tsujii	$\Rightarrow$ 1 inv.
2002	3	Pelzl	
2002	3	Guyot & Patankar	

Some more support

- Gaudry, Harley. Counting points on hyperelliptic curves over finite fields. 2000.
- Hess, Seroussi, and Smart, Two topics in hyperelliptic cryptography, 2000.
  Gives efficient compression method.
- Weng. Konstruktion kryptographisch geeigneter Kurven mit komplexer Multiplikation. 2001.
  One can efficiently construct curves with the CM method over large prime fields.
- Gaudry, Schost. Construction of secure random curves of genus 2 over prime fields. 2004.
  Reaches prime fields of 80 bits & proposes some curves with almost prime order divisor class group.

General doubling, $g = 2$

Doubling, $\deg u = 2$			
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$		
Output	$[u', v'] = 2[u, v]$		
Step	**Expression**	**odd**	**even**
1	compute $\tilde{v} \equiv (h + 2v) \bmod u = \tilde{v}_1 x + \tilde{v}_0$: $\tilde{v}_1 = h_1 + 2v_1 - h_2 u_1,\ \tilde{v}_0 = h_0 + 2v_0 - h_2 u_0$;		
2	compute resultant $r =$res$(\tilde{v}, u)$: $w_0 = v_1^2,\ w_1 = u_1^2,\ w_2 = \tilde{v}_1^2,\ w_3 = u_1 \tilde{v}_1,\ r = u_0 w_2 + \tilde{v}_0(\tilde{v}_0 - w_3)$;	2S, 3M $(w_2 = 4w_0)$	2S, 3M (see below)
3	compute almost inverse $inv' = invr$: $inv'_1 = -\tilde{v}_1,\ inv'_0 = \tilde{v}_0 - w_3$;		
4	compute $k' = (f - hv - v^2)/u \bmod u = k'_1 x + k'_0$: $w_3 = f_3 + w_1,\ w_4 = 2u_0,\ k'_1 = 2(w_1 - f_4 u_1) + w_3 - w_4 - h_2 v_1$; $k'_0 = u_1(2w_4 - w_3 + f_4 u_1 + h_2 v_1) + f_2 - w_0 - 2f_4 u_0 - h_1 v_1 - h_2 v_0$;	1M	2M (see below)
5	compute $s' = k' inv' \bmod u$: $w_0 = k'_0 inv'_0,\ w_1 = k'_1 inv'_1,\ s'_1 = (inv'_0 + inv'_1)(k'_0 + k'_1) - w_0 - w_1(1 + u_1),\ s'_0 = w_0 - u_0 w_1$;	5M	5M
6	compute $s'' = x + s_0/s_1$ and s_1: $w_1 = 1/(rs'_1)(= 1/r^2 s_1),\ w_2 = rw_1(= 1/s'_1),\ w_3 = s'^2_1 w_1(= s_1)$; $w_4 = rw_2(= 1/s_1),\ w_5 = w_4^2,\ s''_0 = s'_0 w_2$;	I, 2S, 5M	I, 2S, 5M
7	compute $l' = s'' u = x^3 + l'_2 x^2 + l'_1 x + l'_0$: $l'_2 = u_1 + s''_0,\ l'_1 = u_1 s''_0 + u_0,\ l'_0 = u_0 s''_0$;	2M	2M
8	compute $u' = s^2 + (h + 2v)s/u + (v^2 + hv - f)/u^2$: $u'_0 = s''^2_0 + w_4(h_2(s''_0 - u_1) + 2v_1 + h_1) + w_5(2u_1 - f_4),\ u'_1 = 2s''_0 + h_2 w_4 - w_5$;	S, 2M	S, M
9	compute $v' \equiv -h - (l + v) \bmod u' = v'_1 x + v'_0$: $w_1 = l'_2 - u'_1,\ w_2 = u'_1 w_1 + u'_0 - l'_1,\ v'_1 = w_2 w_3 - v_1 - h_1 + h_2 u'_1$; $w_2 = u'_0 w_1 - l'_0,\ v'_0 = w_2 w_3 - v_0 - h_0 + h_2 u'_0$;	4M	4M
total			each I, 22M, 5S

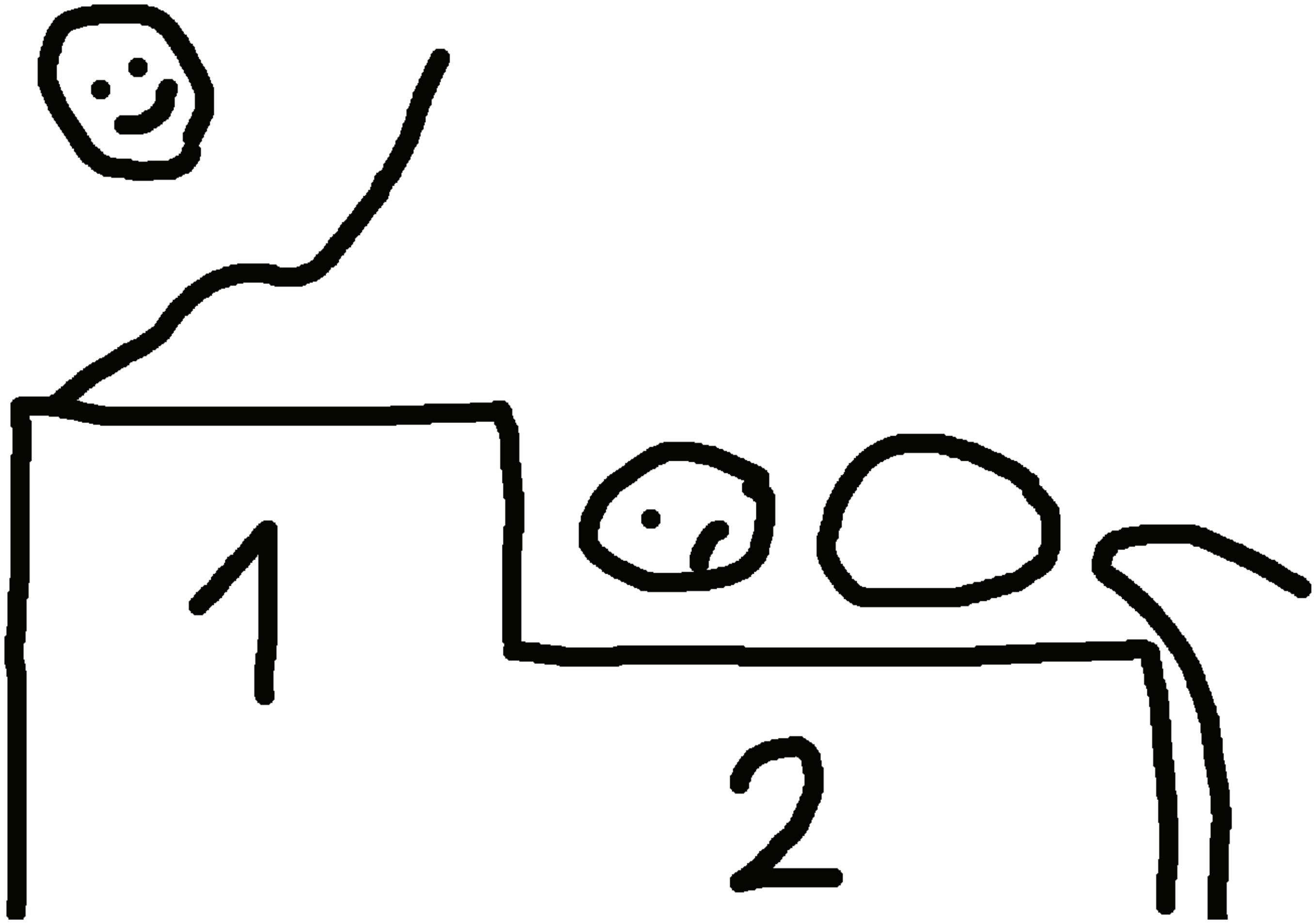
Elliptic curves vs. $g = 2$

- Let's look at the numbers first – sorry Dan!
- Doubling on elliptic curve needs $1I_q, 2M_q, 2S_q$.
- Doubling on genus 2 curve needs $1I_{q'}, 22M_{q'}, 5S_{q'}$.
- $\log q \sim 2 \log q'$ – naive field arithmetic $M_q = 3M_{q'}$ and $M = S \Rightarrow$ comparison depends on $1I_q <=> 1I_{q'} + 15M_{q'}$ (and on whether the naive assumptions hold).
- Theoretical analysis depends on inversion/multiplication (I/M)-ratio, S/M-ratio and ratio $M_q/M_{q'}$.
- Hard to take into account load instructions etc.
- My GMP and NTL implementation didn't look too promising for HEC ...
- ... which could be explained by too general software (one could hope).

Roberto Avanzi, CHES 2004

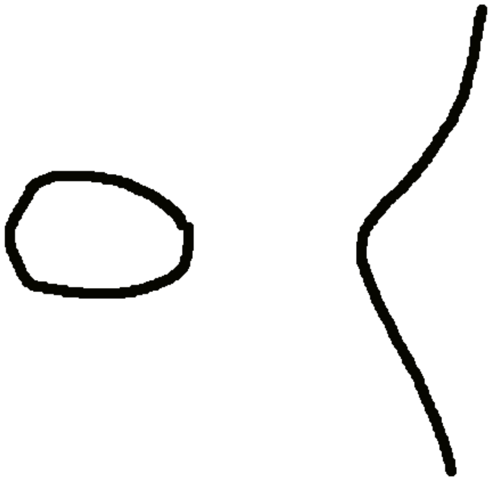
“We present an implementation of elliptic curves and of hyperelliptic curves of genus 2 and 3 over prime fields. To achieve a fair comparison between the different types of groups, we developed an ad-hoc arithmetic library, designed to remove most of the overheads that penalize implementations of curve-based cryptography over prime fields. These overheads get worse for smaller fields, and thus for larger genera for a fixed group size. We also use techniques for delaying modular reductions to reduce the amount of modular reductions in the formulae for the group operations. The result is that the performance of hyperelliptic curves of genus 2 over prime fields is much closer to the performance of elliptic curves than previously thought. For groups of 192 and 256 bits the difference is about 14% and 15% respectively.

End of Round 1



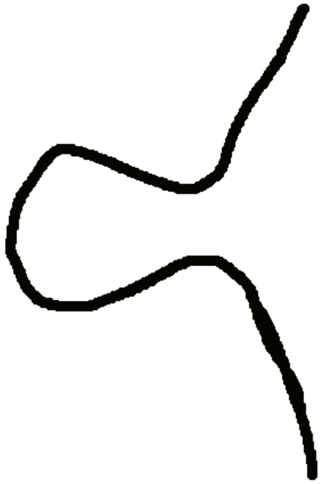
Round 2

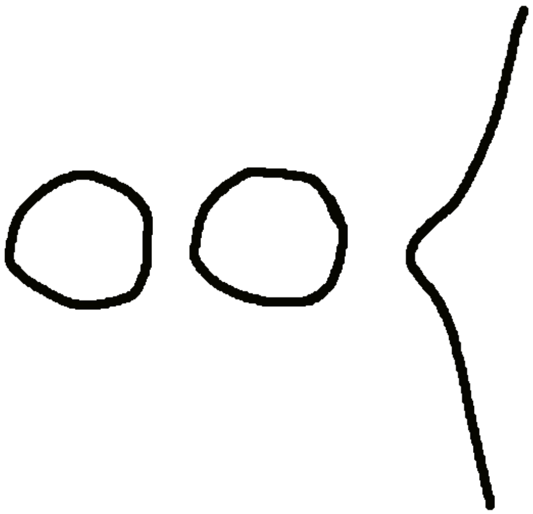
Let’s have a closer look at the opponents . . .

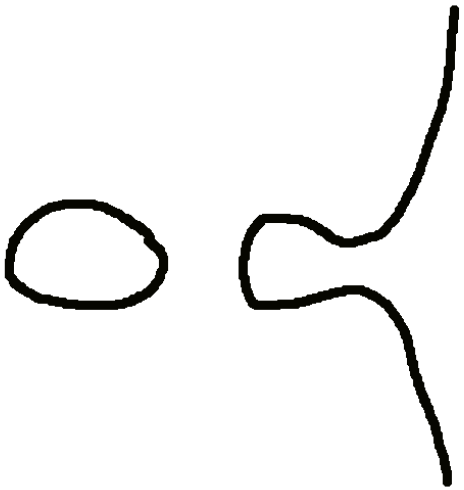


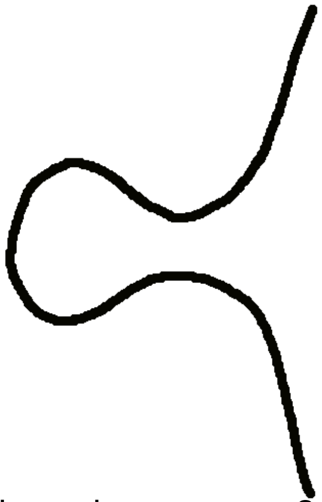
Three types for $g = 1$ (the one with no root cannot be drawn).

Four types for $g = 2$.









Zoom on the candidates

- Larger genus curves have more parameters and thus allow more choices.
- Hope to find efficient and secure families of curves.
- Special family of HEC might be faster than comparable family of EC – or there might be no similar family.
- At ECC 2006 Dan Bernstein reported success for $g = 2$. His implementation of Gaudry's formulas (based on Chudnovsky & Chudnovsky) is faster than the so far record using Montgomery's formulas on elliptic curves.
- Story for $g = 3$ still unwritten but one can hope for even larger speed-up in the Montgomery approach.

Genus 2 curves over $\mathbb{F}_{2^n}$

Get higher efficiency by specializing to degree of h.

$$C : y^2 + (h_2x^2 + h_1x + h_0)y = f(x), \deg(f) = 5.$$

- $h = 0$ belongs to singular curves, so not hyperelliptic.
- $\deg h = 0$: the curve is supersingular.
- $\deg(h) = 1$: group has cofactor 2, no weaknesses up to our present knowledge. This is a family of 2^{2n+1} isomorphism classes. More details on this family now.
- If $\deg h = 2$ and h has a root then one can achieve $h(x) = x^2 + h_1x$, otherwise $h(x) = x^2 + h_1x + h_0$. In the former case the group order is divisible by 4, in the latter by 2.

Isomorphic transformations $\deg(h) = 1$

Changes of the form $y \to a^5 y' + bx'^2 + cx' + d, x \to a^2 x' + e$ allow $f_4 = f_1 = 0$ and some more.

Usual situation:

$$C : y^2 + h_1 xy = x^5 + f_3 x^3 + f_2 x^2 + f_0.$$

If extension degree n of $\mathbb{F}_{2^n}$ is odd one can additionally achieve $h_1 = 1, f_2 \in \mathbb{F}_2$.

In scalar multiplications DBL more important than ADD (use precomputation) $\Rightarrow$ speed up DBL.

Joint work with Mark Stevens, 2004.

Doubling, $g = 2$, $\deg h = 1$

Doubling $\deg h = 1$, $\deg u = 2$				
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$; h_1^2, h_1^{-1}			
Output	$[u', v'] = 2[u, v]$			
Step	Expression	$h_1 = 1$	h_1^{-1} small	h_1 arbitrary
1	compute rs_1: $z_0 = u_0^2$, $k_1' = u_1^2 + f_3$; $w_0 = f_0 + v_0^2 (= rs_1'/h_1^3)$;	3S	3S	3S
2	compute $1/s_1$ and s_0'': $w_1 = (1/w_0) z_0 (= h_1/s_1)$; $z_1 = k_1' w_1$, $s_0'' = z_1 + u_1$;	I, 2M	I, 2M	I, 2M
3	compute u': $w_2 = h_1^2 w_1$, $u_1' = w_2 w_1$; $u_0' = s_0''^2 + w_2$;	2S	S, 2M	S, 2M
4	compute v': $w_3 = w_2 + k_1'$; $v_1' = h_1^{-1}(w_3 z_1 + w_2 u_1' + f_2 + v_1^2)$; $v_0' = h_1^{-1}(w_3 u_0' + f_1 + z_0)$;	S, 3M	S, 3M	S, 5M
total		I, 6S, 5M	I, 5S, 7M	I, 5S, 9M

Elliptic vs. Hyperelliptic

- This family of curves has 2-rank 1, is thus neither supersingular nor ordinary.
- For elliptic curves we do not have anything intermediate.
- Scalar multiplication much faster on genus-2 curves than on EC (time for precomputations is included, each with optimal window size).
- Avanzi, Thériault, and Wang, Rethinking Low Genus Hyperelliptic . . . , 2006. Gives study of field and explicit curve arithmetic for genera 3 and 4. Concentrates on special cases

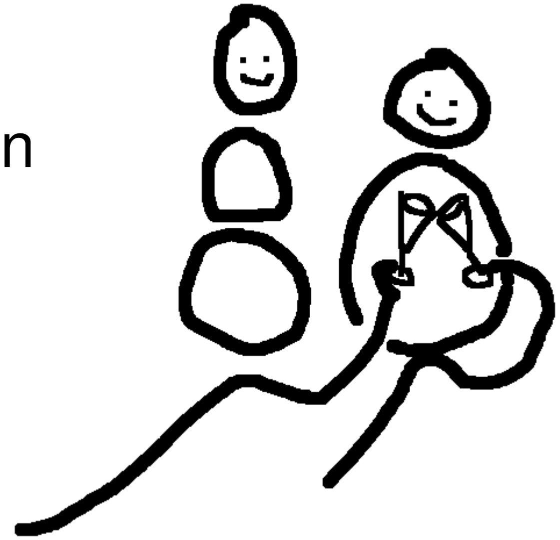
$$\begin{aligned} y^2 + y &= x^7 + f_5 x^5 \cdots \\ y^2 + y &= x^9 + f_7 x^7 \cdots, f_7 \neq 0. \end{aligned}$$

Avanzi, Thériault, and Wang

- Careful implementation of binary field arithmetic.
- Timings for $g = 1, 2, 3$, and 4, each with best coordinate system available.
- Performance of curves of genus four is comparable to that of EC often better.
- Curves of genus two and three similar performance.
- Genus three wins in some cases.
  Depends mainly on relation between
  finite field sizes in relation to
  word size.
- Genus $g = 4$ performance similar to elliptic curves.

Avanzi, Thériault, and Wang

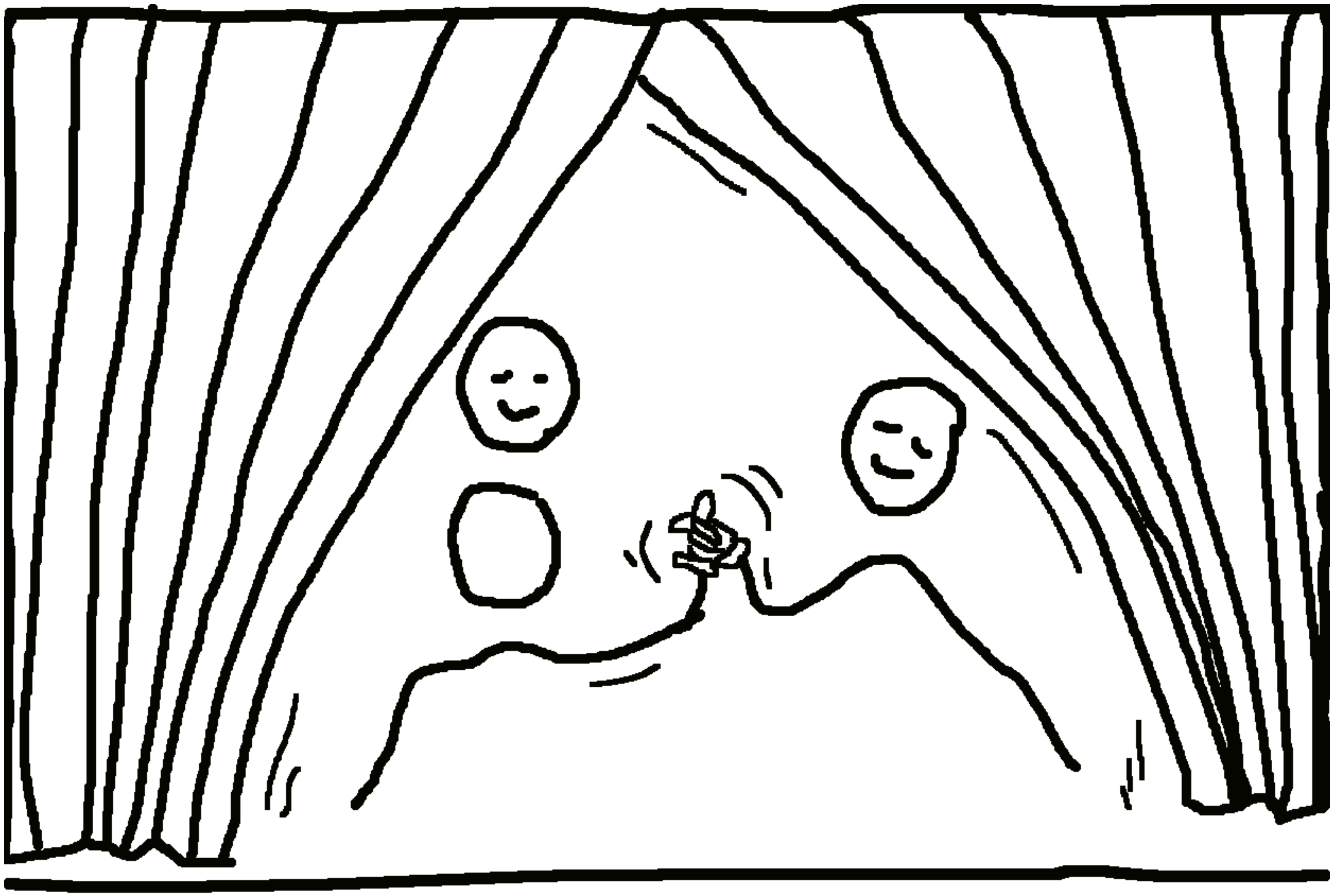
- Careful implementation of binary field arithmetic.
- Timings for $g = 1, 2, 3$, and 4, each with best coordinate system available.
- Performance of curves of genus four is comparable to that of EC often better.
- Curves of genus two and three similar performance.
- Genus three wins in some cases. Depends mainly on relation between finite field sizes in relation to word size.
- Genus $g = 4$ performance similar to elliptic curves.



Further improvements

- Inversion-free coordinate systems lead to extra speed up.
- Halving (inverse operation to doubling) can lead to faster computation and was generalized to genus 2 and 3.
- Hyperelliptic curves offer speed advantages in the computation of pairings.
  - Use special representatives of the divisor classes.
  - Main loop can be shortened more than for elliptic curves.
  - Paper with G. Frey at ANTS 2006 showing that this approach is applicable in general.

Special $g = 2$ families are very fast



The end