Appendix B

Hints and Solutions to Exercises

Chapter 1: Introduction

1.3.3: Encryption is deterministic so one can compare the challenge ciphertext c with $\mathsf{m}_0^e \pmod N$.

1.3.4: Given c, submit $\mathsf{c}' = \mathsf{c}2^e \pmod N$ to the decryption oracle to get $2\mathsf{m} \pmod N$ and hence compute m.

1.3.5: If it does not have semantic security there is some function $f : \mathsf{M}_\kappa \to \{0,1\}$ which can be computed given a ciphertext, so choose messages $\mathsf{m}_0, \mathsf{m}_1$ such that $f(\mathsf{m}_i) = i$ and then one can break IND security.

1.3.7: A UF-CMA adversary is a randomised polynomial-time algorithm which takes as input a public key pk for a signature scheme, is allowed to query a signing oracle on messages of its choice, and outputs a message m and a signature s. The adversary wins if the signature s satisfies the verification algorithm on message m and key pk and if m was not one of the queries to the signing oracle. A scheme has UF-CMA security if every adversary succeeds with only negligible probability (in the security parameter). See Definition 12.2 of Katz and Lindell [334].

1.3.8: Yes, if the RSA problem is hard.

1.3.9: Choose random s and set $\mathsf{m} = \mathsf{s}^e \pmod N$.

1.3.10: Given m call signing oracle on $2^e\mathsf{m} \pmod N$ to get s'. Output $\mathsf{s} = \mathsf{s}'2^{-1} \pmod N$.

Chapter 2: Basic Algorithms

2.2.3: See Section 9.5.1 of Crandall and Pomerance [162], Section 3.5 of Shoup [556] or Section 8.1 of von zur Gathen and Gerhard [238].

2.2.8: See Section 9.2.2 of [162], Sections 1.5.1 and 1.5.2 of Brent and Zimmermann [100] or Section 1.7.1 of Cohen [136]. The complexity is $O(M(\log(a)))$,

2.2.9: Since $e \leq \log_2(N)$ the idea is to compute $N^{1/e}$ for $e = 1, 2, \ldots, \lfloor \log_2(N) \rfloor$ to find the largest e such that $N^{1/e} \in \mathbb{N}$.

2.4.4: See Section 5.9 of Bach and Shallit [22] or Algorithm 2.3.5 of Crandall and Pomerance [162].

2.4.5: Show the algorithm is faster than computing $\gcd(m, n)$ using Euclid; the fundamental step is computing m mod n but the numbers are always at most as large as those in Euclid's algorithm.

2.4.6: Choose random $1 < a < p$ and compute $(\frac{a}{p})$ until the Legendre symbol is -1. Since the probability of success is at least 0.5 for each trial, the expected number of trials is 2. This is a Las Vegas algorithm (it may never terminate, but the answer is always correct). See the proof of Lemma 2.9.5 for further details.

2.4.9: This goes back to A. Cobham. See Shoup [557] or Bach and Sorensen [23] for the analysis.

2.4.10: Computing Legendre symbols using quadratic reciprocity requires $O(\log(p)^2)$ bit operations while computing $a^{(p-1)/2} \pmod p$ needs $O(\log(p)M(\log(p)))$ bit operations (see Corollary 2.8.4). So using quadratic reciprocity is (theoretically) always better.

2.5.5: First compute and store $b_2, \ldots, b_m$ where $b_i = \prod_{j=1}^{i} a_j$, then b_m^{-1}, then, for $i = m$ downto 2 compute $a_i^{-1} = b_{i-1}b_i^{-1}, b_{i-1}^{-1} = a_i b_i^{-1}$. See Algorithm 11.15 of [16], Section 2.4.1 of [100] or Section 2.2.5 of [274].

2.6.4:

2.8.6: For pseudocode of the algorithm see Algorithm IV.4 of [64] or Algorithm 9.10 of [16].

2.8.7: Consider a window of length w representing an odd integer. If the bits of m are uniformly distributed then the probability the next most significant bit after the window is 0 is 0.5. Hence the expected number of zeroes before the first 1 is $0.5 \cdot 0 + 0.25 \cdot 1 + 0.125 \cdot 2 + \cdots = \sum_{i=1}^{\infty} i/2^{i+1} = 1$.

2.8.10: For pseudocode see Section 2.2 of Möller [432]. The problem is that there is no fixed precomputation of powers of g.

2.8.11: The values m_i in addition chain satisfy $m_i \leq 2^i$.

2.9.8: See Adleman, Manders and Miller [3].

2.10.2: See Chapters 2 and 3 of von zur Gathen and Gerhard [238].

2.10.4: Assume that $F(0) \neq 0$ so that x is coprime to $F(x)$. Replace $R = 2^k$ in the description of Montgomery reduction in Section 2.5 by $R = x^d \pmod{F(x)}$. See Koç and Acar [349] for details and discussion.

2.10.5: See Section 9.6.3 of [162].

2.12.1: Compute $F'(x)$. If $F'(x) = 0$ then, by Lemma A.5.2, $F(x) = G(x)^p$ where $p = \mathrm{char}\mathbb{F}_q$, and $F(x)$ is not square-free. Otherwise, compute $\gcd(F(x), F'(x))$ and if this is not 1 then $F(x)$ is not square-free. The computation requires $O(\deg(F)^2)$ field operations.

2.12.2: Given $F(x) \in \mathbb{F}_q[x]$ compute $F'(x)$. If $F'(x) = 0$ then $F(x) = G(x)^p$ and repeat process on $G(x)$. Otherwise, compute $F_1(x) = F(x)/\gcd(F(x), F'(x))$ and factor $F_1(x)$. Once the factors of $F_1(x)$ are known then one can factor $F(x)/F_1(x)$ efficiently by testing divisibility by the known factors of $F_1(x)$.

2.12.3: One computes $x^q \pmod{F(x)}$ in $O(\log(q)d^2)$ field operations and a further $O(d^2)$ operations are needed for the gcd computation. The answer to the decision problem is "yes" if and only if $\deg(R_1(x)) > 0$.

2.12.5: One is essentially doing "divide and conquer" to separate the $\deg(R_1(x)) \leq d$ roots. Each trial is expected to split $R_1(x)$ into two polynomials of degree $\approx \deg(R_1(x))/2$. Hence, an expected $O(\log_2(\deg(R_1(x)))) = O(\log_2(d))$ trials are required to factor $R_1(x)$. Since each trial involves computing $u(x)^{(q-1)/2} \pmod{R_1(x)}$ (which has complexity $O(\log(q)M(d))$ operations in $\mathbb{F}_q$) and then computing the gcd of polynomials of degree $\leq d$ (which requires $O(d^2)$ operations in $\mathbb{F}_q$) the total expected cost is $O(\log(q)\log(d)d^2)$ field operations. For a detailed analysis of the probabilities see Section 21.3.2 of Shoup [556].

2.12.10: If $F(x)$ is not irreducible then it has a factor of degree $\leq \deg(F(x))/2$. Hence it is sufficient that $\gcd(F(x), x^{q^i} - x) = 1$ for $1 \leq i \leq d/2$. To compute all these polynomials efficiently one computes $s_1(x) \equiv x^q \pmod{F(x)}$ and then $\gcd(F(x), s_1(x) - x)$. For the next step compute $s_2(x) = s_1(x)^q \pmod{F(x)}$ and $\gcd(F(x), s_2(x) - x)$,

and so on. There are $d/2$ steps, each taking $O(\log(q)d^2)$ field operations, so the overall complexity is $O(d^3 \log(q))$. See Algorithm IPT of Section 21.1 of [556] or Section 3 of [370].

2.12.11: One first computes $\gcd(x^{q^b} - x, F(x))$ in time $O(b \log(q))$ operations on polynomials of degree at most d. The rest of the algorithm is the same as the root finding algorithm of Exercise 2.12.5 where we raise polynomials to at most the power q^b.

2.14.1: Construct $\{\theta, \theta^q, \ldots, \theta^{q^{m-1}}\}$ in $O(m^3)$ operations in $\mathbb{F}_q$ (using the fact that q-th powering is linear) and then $O(m^3)$ operations in $\mathbb{F}_q$ for the Gaussian elimination. Hence the complexity is an expected $O(m^3 \log_q(m))$ bit operations.

2.14.4: Suppose $\theta^2 = d$. Given $g = g_0 + g_1\theta$ we must solve for $x, y \in \mathbb{F}_q$ such that $(x + \theta y)^2 = g_0 + g_1\theta$. Expanding gives $x^2 + dy^2 = g_0$ and $2xy = g_1$ and one can eliminate y to get $4x^4 - 4g_0x^2 + dg_1^2 = 0$ which can be solved using two square roots in $\mathbb{F}_q$.

2.14.5: See Fong-Hankerson-López-Menezes [207].

2.14.10: Computing S_m requires $O(m^3)$ operations in $\mathbb{F}_q$. Computing a linear dependence among m vectors can be done using Gaussian elimination in $O(m^3)$ field operations.

2.14.11: Since $1 - 1/q \geq 1/2$ the number of trials is at most 2. The complexity is therefore an expected $O(m^3)$ operations in $\mathbb{F}_q$.

2.14.12: Expected $O(m^3)$ operations in $\mathbb{F}_{q^m}$.

2.14.13: The cost of finding a root of the polynomial $F_1(x)$ of degree m is an expected $O(\log(m)\log(q)m^2)$ operations in $\mathbb{F}_q$ and this dominates the cost of the isomorphism algorithm in the forwards direction. The cost of the linear algebra to compute the inverse to this isomorphism is $O(m^3)$. (If one repeats the algorithm with the roles of F_1 and F_2 swapped then one gets a field isomorphism from $\mathbb{F}_q[y]/(F_2(y))$ to $\mathbb{F}_q[x]/(F_1(x))$ but it is not necessarily the inverse of the function computed in the first step.

2.15.1: Writing $q - 1 = \prod_{i=1}^{m} l_i^{e_i}$ we have $m = O(\log(q))$ and then computing $g^{(q-1)/l_i}$ requires $O(\log(q)^3)$ bit operations.

2.15.8: The naive solution requires $\approx 2/3 \log_2(N)$ group operations in the precomputation, followed by $\approx 3\frac{1}{2} \log_2(N^{2/3}) = \log_2(N)$ group operations. The same trick can be used in the first stage of Algorithm 4, giving $\approx \frac{2}{3} \log_2(N)$ group operations. But the three exponentiations in the second stage are all to different bases and so the total work is $\approx \frac{3}{2} \log_2(N)$ group operations. The naive solution is therefore better. The naive method becomes even faster compared with Algorithm 4 if one uses window methods.

Chapter 5: Varieties

5.1.3: $V(y - x^2), V(y^2 - x), V((x-1)^3 - y^2)$.

5.1.4: If $f(x, y)$ has no monomials featuring y then it is a polynomial in x only; take any root $a \in \overline{\mathbb{k}}$ and then $\{(a, b) : b \in \overline{\mathbb{k}}\} \subseteq V(f)$. For the remaining case, for each $a \in \overline{\mathbb{k}}$ then $f(a, y)$ is a polynomial in y and so has at least one root $b \in \overline{\mathbb{k}}$.

5.1.7: Let $z = x + iy \in \mathbb{F}_{p^2}$. Then $z^p = x - iy$. Hence $z^{p+1} = 1$ if and only if $1 = (x + iy)(x - iy) = x^2 + y^2$. If follows that the map $x + iy \mapsto (x, y)$ is a bijection from G to $X(\mathbb{F}_p)$. Showing that this is a group isomorphism is straightforward.

5.1.10: In $X = \mathbb{A}^1(\mathbb{F}_p)$ we have $X = V(x^p - x)$.

5.2.8: $V(x^2 + y^2 - z^2)(\mathbb{R})$ is the same as the circle $V(x^2 + y^2 - 1)(\mathbb{R}) \subseteq \mathbb{A}^2(\mathbb{R})$. Over $\mathbb{C}$ it would also contain the points $(1 : \pm i : 0)$.

$V(yz - x^2)$ is the parabola $y = x^2$ in $\mathbb{A}^2$ together with the single point $(0 : 1 : 0)$ at infinity. Note that this algebraic set is exactly the same as the one in Example 5.2.7 under a change of coordinates.

5.2.21: $(1 : 0 : 0), (0 : 1 : 0), (0 : 0 : 1)$.

5.2.35: We have $\overline{X} = V(f(x_0, x_1, x_2)) = V(\overline{f})$. A point in $\overline{X} - X$ has $x_2 = 0$ and at least one of $x_0, x_1 \neq 0$. Hence, the points at infinity are in the union of $(\overline{X} \cap U_0) \cap V(x_2) = \{(1 : y : 0) : f(1, y, 0) = 0\}$ and $(\overline{X} \cap U_1) \cap V(x_2) = \{(x : 1 : 0) : f(x, 1, 0) = 0\}$. Both sets are finite.

5.3.3: It decomposes as $V(x, z) \cup V(w - x, x^2 - yz)$ and one can see that neither of these sets is equal to X.

5.3.5: $k[x, y]/(y - x^2) \cong k[x]$ is an integral domain, so $(y - x^2)$ is prime.

5.3.6: If $gh \in I_{\Bbbk}(X)$ then $g(f_1(t), \ldots, f_n(t))h(f_1(t), \ldots, f_n(t))$ is a polynomial in t which is identically zero on $\overline{\Bbbk}$. The result follows. See Proposition 5 of Section 4.5 of Cox, Little and O'Shea [158] for details and a generalisation.

5.3.14: If $U_1 \cap U_2 = \varnothing$ then $X = (X - U_1) \cup (X - U_2)$ is a union of proper closed sets.

5.4.6: We show that $(f_1/f_2)(P) - (f_3/f_4)(P) = 0$. By assumption $f_2(P), f_4(P) \neq 0$ and so it suffices to show $(f_1 f_4 - f_2 f_3)(P) = 0$. This follows since $(f_1 f_4 - f_2 f_3) \in I_{\Bbbk}(X)$.

5.5.7: Recall from Exercise 5.3.14 that $U_1 \cap U_2 \neq \varnothing$. Then apply the same arguments as in the proof of Theorem 5.4.8.

5.5.9: The solution is not unique. An example of maps $\phi : X \to Y$ and $\psi : Y \to X$ is $\phi(x, y) = (x : 1 : 1)$ and $\psi(x_0 : x_1 : x_2) = (x_0/x_1, x_1/x_0)$. One can check that $\phi \circ \psi$ and $\psi \circ \phi$ are the identity when they are defined.

5.5.14: The line of slope t through $(-1, 0)$ has equation $y = t(x + 1)$ and hits X and $(-1, 0)$ and a point $\phi(t) = (x(t), y(t)) = ((1 - t^2)/(1 + t^2), 2t/(1 + t^2))$. Hence, define $\phi : \mathbb{P}^1 \to X$ by $f([t : u]) = ((u^2 - t^2)/(u^2 + t^2), 2ut/(u^2 + t^2))$. The inverse morphism $\psi : X \to \mathbb{P}^1$ is $\psi(x, y) = (y : x + 1) = (x - 1 : y)$; note that these two descriptions of ψ are equivalent (multiply the former through by $y/(x + 1)$) and that the former is regular away from $(-1 : 0)$ and the latter regular away from $(1 : 0)$. Since the slope of the line between (-1.0) and (x, y) is $y/(x + 1)$ it follows that $\psi \circ \phi$ and $\phi \circ \psi$ are the identity.

5.5.19: Write Z for the Zariski closure of $\phi(X)$ in Y. If $Z = Z_1 \cup Z_2$ is a union of closed sets then $X = \phi^{-1}(Z_1) \cup \phi^{-1}(Z_2)$ is also a union of closed sets so, without loss of generality, $\phi^{-1}(Z_1) = X$ and so Z_1 contains the Zariski closure of $\phi(X)$.

5.5.26: Suppose $\theta(f) = 0$ for some $f \in K_1^*$. Then $ff^{-1} = 1$ and so $1 = \theta(1) = \theta(f)\theta(f^{-1}) = 0$ which is a contradiction.

5.6.5: A proof is given in Proposition 1.13 of Hartshorne [278]. We also give a proof here: Let $f \in \Bbbk[x_1, \ldots, x_n]$. Without loss of generality suppose f contains at least one monomial featuring x_n. Then write $f = f_r x_n^r + f_{r-1} x_n^{r-1} + \cdots + f_0$ with $f_i \in \Bbbk[x_1, \ldots, x_{n-1}]$ or $\Bbbk[x_0, \ldots, x_{n-1}]$. One can check that the field $\Bbbk(X)$ contains a subfield isomorphic to $\Bbbk(x_1, \ldots, x_{n-1})$. Finally, $\Bbbk(X)$ is an algebraic extension of $\Bbbk(x_1, \ldots, x_n)$.

5.6.6: $\overline{\Bbbk}(X) = \overline{\Bbbk}[X] = \overline{\Bbbk}$ and so $I_{\overline{\Bbbk}}(X)$ is a maximal ideal and so by the Nullstellensatz $X = \{P\}$.

5.6.10: Let X be a variety of dimension 1 and let Y be a proper closed subset. Let $Y = \cup_{i=1}^n Y_i$ be the decomposition of Y into irreducible components. By Corollary 5.6.9 we have $\dim(Y_i) = 0$. Exercise 5.6.6 implies $\#Y_i(\overline{\Bbbk}) = 1$ and so $\#Y(\overline{\Bbbk}) = n$.

5.7.5: $V(y_1^2 - y_2^2, y_1 y_2 - 1)$.

5.7.6: $V(y_{1,1}^2 - y_{1,2}^2 + y_{2,1}^2 - y_{2,2}^2 - 1, y_{1,1}y_{1,2} + y_{2,1}y_{2,2} - 1)$.

Chapter 6: Tori, LUC and XTR

6.1.4: If $p \nmid n$ then

$$\Phi_{np}(x) = \prod_{d|n} \left((x^{np/d} - 1)^{\mu(d)} (x^{n/d} - 1)^{\mu(dp)} \right)$$

and use $\mu(dp) = -\mu(d)$. If $p \mid n$ then

$$\Phi_{np}(x) = \left(\prod_{d|n}(x^{np/d}-1)^{\mu(d)}\right)\left(\prod_{d|np, d\nmid n}(x^{np/d}-1)^{\mu(d)}\right).$$

and note that if $d \mid np$ but $d \nmid n$ then $p^2 \mid d$ and so $\mu(d) = 0$. The final statement follows since, when n is odd, $z^n = 1$ if and only if $(-z)^{2n} = 1$.

6.3.2: To compute $(u+v\theta)(u'+v'\theta)$ compute uu', vv' and $(u+v)(u'+v')$. To square $(u+v\theta)$ compute u^2, v^2 and $(u+v)^2$. One inverts $(u+v\theta)$ by computing $(u+v\overline{\theta})/(u^2 - Auv + Bv^2)$.

6.3.3: The first two parts are similar to Exercise 6.3.2. For inversion, note that $(u+v\theta)^{-1} = (u-v\theta)/(u^2+v^2)$. For square roots, note that $(u+v\theta)^2 = (u^2-v^2)+2uv\theta$ so to compute the square root of $a+b\theta$ requires solving these equations. We assume here that $a+b\theta$ is a square, and so $(\frac{a^2+b^2}{q}) = 1$. One finds that $u^4 - au^2 - b^2/4 = 0$ and so $u^2 = (a \pm \sqrt{a^2+b^2})2^{-1}$. Only one of the two choices is a square, so one must compute the Legendre symbol $(\frac{a+\sqrt{a^2+b^2}}{q})$ to determine the answer (taking into account that $(\frac{2}{q})$ is known from $q \pmod 8$). One then multiplies the appropriate $(a \pm \sqrt{a^2+b^2})$ by 2^{-1}, which is easy (either shift right or add q and shift right). Taking the square root gives u. An inversion and multiplication gives v.

6.3.15: See Williams [634].

6.3.17: The characteristic polynomial of g is $(x-g)(x-g^q) = x^2 - \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(g)x + 1$ so $\mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(g) = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(g')$ implies g' is a root of the same polynomial as g.

6.3.22: 34.

6.3.23: Exercises 6.3.2 and 6.3.3 give costs of 3 squarings in $\mathbb{F}_q$ for computing a squaring in $\mathbb{F}_{q^2}$ and 3 squarings plus 3 multiplications in $\mathbb{F}_q$ for a square-and-multiply. Lucas sequences give just one multiplication and squaring for each operation; only one third the number of operations in the worst case.

6.4.5: Need $q \equiv 2 \pmod 3$ or else $\zeta_3 \in \mathbb{F}_q$. Also need $q^2 \not\equiv 1 \pmod 9$ or else $\zeta_9 \in \mathbb{F}_{q^2}$.

6.4.7: Consider $0 = f(a)^q = a^{3q} - t^q a^{2q} + t^{q^2} a^q - 1 = (-1 + t^q a^{-q} - t a^{-2q} + a^{-3q})(-a^{3q})$. This shows that $f(a^{-q}) = 0$. Now, suppose $f(x)$ is neither irreducible nor split completely. Then $f(x) = (x-u)g(x)$ where $u \in \mathbb{F}_{q^2}$ and $g(x)$ is an irreducible quadratic over $\mathbb{F}_{q^2}$. Let $a \in \mathbb{F}_{q^4} - \mathbb{F}_{q^2}$ be a root of $g(x)$. Then a^{-q} is the other root, but then $(a^{-q})^{-q} = a^{q^2} = a$ and we have $a \in \mathbb{F}_{q^2}$.

6.4.10: See Lenstra and Verheul [375].

6.4.11: Squaring requires computing $(t_{2n+1}, t_{2n}, t_{2n-1})$ which is one iteration of each formula in the previous exercise. Square-and-multiply requires computing $(t_{2n+2}, t_{2n+1}, t_{2n})$ which is twice formula 2 and once formula 3 above. Let M (respectively S, P) be the costs of a multiplication (respectively, squaring, q-th powering) in $\mathbb{F}_{q^2}$. Then a squaring (i.e., going from t_n to t_{2n}) requires $4M + S + 5P$ while a square-and-multiply requires $2M + 2S + 4P$. The reader is warned that in practice it is often more efficient to use other ladder algorithms than traditional square-and-multiply, see Chapter 3 of Stam [579].

6.4.12: The trick is to represent t_n using the triple (t_{n+1}, t_n, t_{n-1}) as above when n is odd, but (t_{n+2}, t_{n+1}, t_n) when n is even. One then must use formula 1 of Exercise 6.4.10. For example, if n is even and one is computing $(t_{2n+2}, t_{2n+1}, t_{2n})$ then the middle term is computed as $t_{2(n+1)-1}$. See Lenstra and Verheul [375] for details.

6.4.13: $t_6 = 7 + 48i, t_7 = 35 + 65i, t_8 = 6 + 8i$.

6.4.14: One can represent the equivalence class of g^n by (t_n, t_{-n}). For the ladder one can store the 6-tuple $(t_{n+1}, t_n, t_{n-1}, t_{-n-1}, t_{-n}, t_{-n+1})$ and compute an analogous 6-tuple

centered at (t_{2n}, t_{-2n}) or (t_{2n+1}, t_{-2n-1}). See Gong and Harn [259] and also Stam [579] for further details.

6.4.15: See Shirase et al [551].

6.6.2: Modulo each prime p_i there are usually two values h such that $\mathrm{Tr}_{\mathbb{F}_{p_i^2}/\mathbb{F}_{p_i}}(h) = \mathrm{Tr}_{\mathbb{F}_{p_i^2}/\mathbb{F}_{p_i}}(g)$. Hence, by the Chinese remainder theorem, there are 2^k values for h in general.

The second claim follows immediately, or one can prove it using the same argument as the first part: modulo each prime p_i, $2 + (p_i - 1)/2 \approx p_i/2$ values arise as the trace of an element in $G_{p_i,2}$.

Chapter 7: Curves and Divisor Class Groups

7.1.7: The first claim is immediate, since $f \in \mathcal{O}_P$ implies $f(P) \in \Bbbk$ is defined, and $(f - f(P)) \in \mathfrak{m}_P$. For the second claim, note that $\mathfrak{m}_P/\mathfrak{m}_P^2$ is an $\mathcal{O}_P/\mathfrak{m}_P$-module, i.e., a $\Bbbk$-vector space. Without loss of generality, $P = (0, \ldots, 0)$. and $\mathfrak{m}_P$ is the $\mathcal{O}_P$-ideal $(x_1, \ldots, x_n)$. Then every monomial $x_i x_j$ for $1 \leq i, j \leq n$ lies in $\mathfrak{m}_P^2$. Therefore every element of $\mathfrak{m}_P$ is of the form $a_1 x_1 + \cdots a_n x_n + g$ where $a_i \in \Bbbk$ for $1 \leq i \leq n$ and $g \in \mathfrak{m}_P^2$. Hence there is a map $d : \mathfrak{m}_P \to \Bbbk^n$ given by

$$d(a_1 x_1 + \cdots + a_n x_n + g) = (a_1, \ldots, a_n).$$

7.1.9: $x = y^2 - yx - x^4 \in \mathfrak{m}_P^2$ so $\{y\}$ is a basis. For the second example there is no linear relation between x and y and so $\{x, y\}$ is a basis.

7.1.15: Corollary 7.1.14: By Exercise 5.6.5 the dimension of X is $d = n - 1$, so $n - d = 1$. Now, the Jacobian matrix is a single row vector and so the rank is 1 unless the vector is the zero vector. Corollary 7.1.14 is just a restatement of this.

7.2.4: Putting $z = 0$ gives the equation $x^3 = 0$ and so $x = 0$. Hence, $(0 : 1 : 0)$ is the only point at infinity. Taking the affine part by setting $y = 1$ gives the equation $E(x, z) = z + a_1 xz + a_3 z^2 - (x^3 + a_2 x^2 z + a_4 x z^2 + a_6 z^3)$ and one can check that $(\partial E/\partial z)(0, 0) = 1 \neq 0$.

7.3.7: Write $\phi(P) = (\phi_0(P) : \cdots : \phi_n(P))$ and let $n = \max\{v_P(\phi_i) : 0 \leq i \leq n\}$. Let t_P be a uniformizer at P. Then $(t_P^{-n}\phi_0 : \cdots : t_P^{-n}\phi_n)$ is regular at P. See Proposition II.2.1 of Silverman [564] for further details.

7.4.3: The first 3 statements are straightforward. The definitions also directly imply that $\mathfrak{m}_v$ is an R_v-ideal. The proof that $\mathfrak{m}_v$ is maximal and that R_v is a local ring is essentially the same as the proof of Lemma 7.1.2. Statement 5 follows from Statement 2.

7.4.6: The result follows since $\mathfrak{m}_{P,\Bbbk}(C)^m = \Bbbk(C) \cap \mathfrak{m}_{P,\Bbbk'}(C)^m$ for any $m \in \mathbb{Z}_{\geq 0}$.

7.4.8: $\mathfrak{m}_P^m = (t_P^m)$ so $f \in \mathfrak{m}_P^m$ and $f \notin \mathfrak{m}_P^{m+1}$ is equivalent to $f = t_P^m u$ where $u \in \mathcal{O}_P - \mathfrak{m}_P$ and hence $u \in \mathcal{O}_P^*$.

7.4.9: Write $f = t_P^a u$ and $h = t_P^b v$ where $u, v \in \mathcal{O}_P^*$ and note that $fh = t_P^{a+b} uv$.

7.4.12: Let $f = f_1/f_2$ with $f_1, f_2 \in \mathcal{O}_{P,\overline{\Bbbk}}(C)$. By Lemma 7.4.7 we have $f_1 = t_P^{v_P(f_1)} u_1$ and $f_2 = t_P^{v_P(f_2)} u_2$ for some $u_1, u_2 \in \mathcal{O}_{P,\overline{\Bbbk}}(C)^*$. If $v_P(f) = v_P(f_1) - v_P(f_2) > 0$ then $f = t_P^{v_P(f_1) - v_P(f_2)} u_1/u_2 \in \mathcal{O}_{P,\overline{\Bbbk}}(C)$ and so P is not a pole of f. On the other hand, if P is a pole of f then $1/f = t_P^{v_P(f_2) - v_P(f_1)} u_2/u_1 \in \mathfrak{m}_{P,\overline{\Bbbk}}(C)$.

7.7.3: If $f \in \overline{\Bbbk}^*$ then $\mathrm{div}(f) = 0$ and the result follows.

7.7.5: (5) follows immediately from part 4 of Lemma 7.4.14.

7.7.7: First, $\mathrm{Prin}_{\Bbbk}(C) \subseteq \mathrm{Div}_{\overline{\Bbbk}}(C)$ since functions only have finitely many poles and zeros. Second, if f is defined over $\Bbbk$ and $\sigma \in \mathrm{Gal}(\overline{\Bbbk}/\Bbbk)$ then $f = \sigma(f)$ and so $\sigma(\mathrm{div}(f)) = \mathrm{div}(f)$ and so $\mathrm{Prin}_{\Bbbk}(C) \subseteq \mathrm{Div}_{\Bbbk}(C)$. Finally, $\mathrm{Prin}_{\Bbbk}(C)$ is closed under addition by Lemma 7.7.4.

7.7.9: Write f as a ratio of homogeneous polynomials of the same degree and then factor them as in equation (7.8).

7.7.15: $f/h \in \Bbbk(C)$ has $\operatorname{div}(f/h) = 0$ and so $f/h \in \Bbbk^*$.

7.9.6: See Cassels [122], Reid [497] or Silverman and Tate [567].

Chapter 8: Rational Maps on Curves and Divisors

8.1.2: Any non-constant morphism ϕ has a representative of the form $(\phi_1 : \phi_2)$ where ϕ_2 is not identically zero, and so define $f = \phi_1/\phi_2$. That ϕ is a morphism follows from Lemma 7.3.6.

8.1.10: $\Bbbk(x, y/x) = \Bbbk(x, y)$.

8.1.11: $\Bbbk(x, y) = \Bbbk(x^2, y)(x) = \phi^*(\Bbbk(C_2))(x)$, which is a quadratic extension.

8.2.3: See Proposition III.1.4 of Stichtenoth [589].

8.2.10: By Lemma 7.4.7, t is a uniformizer at Q if and only if $v_Q(t) = 1$. The problem is therefore equivalent to Lemma 8.2.9.

8.2.11: Since $\phi^*(\Bbbk(C_2)) = \Bbbk(C_1)$ it follows directly from the definition in terms of $\mathcal{O}_P$ and $\mathfrak{m}_P$ that if $\phi(P) = Q$ and $f \in \Bbbk(C_2)$ then $v_Q(f) = v_P(f \circ \phi)$.

8.2.16: Follows from Lemma 8.2.9.

8.3.12: The function can also be written $\phi((x : z)) = (1 : z^2/x^2)$. One has $\phi_*(D) = (1 : 1) + (1 : 0) - (0 : 1)$, $\phi^*(D) = (i : 1) + (-i : 1) + 2(1 : 0) - 2(0 : 1)$, $\phi_*\phi^*(D) = 2(-1 : 1) + 2(1 : 0) - 2(0 : 1) = 2D$ and $\phi^*\phi_*(D) = (1 : 1) + (-1 : 1) + 2(1 : 0) - 2(0 : 1)$.

8.3.15: Follows from Exercise 8.2.17.

8.4.3: See Section 8.2 of Fulton [216] or I.4.5 to I.4.9 of Stichtenoth [589].

8.4.6: By Corollary 7.7.13 all non-constant functions have a pole. Hence $\mathcal{L}_{\Bbbk}(0) = \Bbbk$. Since $\operatorname{div}(f) \equiv 0$ the second result follows from part 6 of Lemma 8.4.2.

8.4.10: The dimensions are $1, 1, 2, 3, 4, 5, 6$.

8.5.15: $\delta(x^p) = px^{p-1}\delta(x) = 0$.

8.5.27: If $\omega_1 = f_1 dt_P$ and $\omega_2 \equiv \omega_1$ then $\omega_2 = f_2 dt_P$ with $f_1 \equiv f_2$ in $\Bbbk(C)$, so there isn't anything to show here.

For the second point suppose t and u are two uniformizers at P. Write $\omega = f_1 dt = f_2 du$, so that $f_1/f_2 = \partial u/\partial t$. We must show that $v_P(f_1) = v_P(f_2)$. Since $u \in \mathfrak{m}_{P,\Bbbk}(C) = (t)$ we have $u = ht$ for some $h \in \Bbbk(C)$ such that $h(P) \neq 0$. Then $\partial u/\partial t = \partial(ht)/\partial t = h + t(\partial h/\partial t)$. It follows that $v_P(\partial u/\partial t) = 0$ and so $v_P(f_1) = v_P(f_2)$.

8.5.29: If $\omega' = f\omega$ then $\operatorname{div}(\omega') = \operatorname{div}(\omega) + \operatorname{div}(f)$.

8.6.3: An elliptic curve with only one point, e.g., $y^2 + y = x^3 + x + 1$ over $\mathbb{F}_2$.

Chapter 9: Elliptic Curves

9.1.1: It is clear that the formula gives the doubling formula when $x_1 = x_2, y_1 = y_2$. For the other case it is sufficient to compute

$$\left(\frac{y_1 - y_2}{x_1 - x_2}\right)\left(\frac{y_1 + y_2 + (a_1x_2 + a_3)}{y_1 + y_2 + (a_1x_2 + a_3)}\right)$$

and simplify to the above formula.

9.1.4: (These ideas originate in the work of Seroussi and Knudsen.) Dividing the equation by x_P^2 gives $(y_P/x_P)^2 + (y_P/x_P) = x_P + a_2 + a_6/x_P^2$ and the result follows by Exercise 2.14.7.

The formula for λ_Q is immediate from equation (9.2) and the formulae for x_P and y_P also follow immediately. If $P = [2]Q$ then $x_P = \lambda_Q^2 + \lambda_Q + a_2$ for some $\lambda_Q \in \mathbb{F}_{2^m}$ and so $\operatorname{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(x_P + a_2) = 0$. Since $\operatorname{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(x_P + a_2 + a_6/x_P^2) = 0$ it follows that $\operatorname{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(a_6/x_P^2) = 0$.

Conversely, if $\mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(a_6/x_P^2) = 0$ then one can solve for $x_Q \in \mathbb{F}_{2^m}$ the equation $x_Q^2 + x_P + a_6/x_Q^2 = 0$. Further, $\mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(x_Q + a_2 + a_6/x_Q^2) = \mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(x_Q + x_Q^2 + a_2 + x_P) = 0$ so there is an element $y_Q \in \mathbb{F}_{2^m}$ such that $Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$. It then follows that $P = [2]Q$.

Finally, the formulae for point halving follow from substituting $y_Q = x_Q(\lambda_Q + x_Q)$ into the formula $y_P = (x_P + x_Q)\lambda_Q + x_Q + y_Q$.

9.1.5: We have $\lambda = (y_1/z_1 - y_2/z_2)/(x_1/z_1 - x_2/z_2) = (y_1 z_2 - y_2 z_1)/(x_1 z_2 - x_2 z_1)$. Similarly, putting $x_1/z_1, x_2/z_2$ and y_1/z_1 in the addition formulae in place of x_1, x_2 and y_1 gives the above formulae.

9.3.2: There is an inverse morphism which is a group homomorphism, also defined over $\Bbbk$ by definition, such that the composition is the identity.

9.3.8: From E_1 to E_2 is just $\phi(x, y) = (x + 1, y + x)$. From E_1 to E_3 is $\phi(x, y) = (x + s^2, y + sx + t)$ where $s \in \mathbb{F}_{2^4}$ satisfies $s^4 + s + 1 = 0$ and $t \in \mathbb{F}_{2^8}$ satisfies $t^2 + t = s^6 + s^2$.

9.4.5: For $t \in \mathbb{F}_{2^4}$ show that $\mathrm{Tr}_{\mathbb{F}_{2^4}/\mathbb{F}_2}(s^6 + s^2) = 1 + u(s^4 + s + 1)^2 = 0$. Theorem 9.3.4 implies every element of $\mathrm{Aut}(E)$ is of this form. Since there are 24 choices for (u, s, t), each giving a different function on $E(\overline{\mathbb{F}}_2)$, one has $\#\mathrm{Aut}(E) = 24$. The fact that $\mathrm{Aut}(E)$ is non-Abelian follows can be shown as follows. Let $\phi_i(x, y) = (u_i^2 x + s_i^2, u_i^3 y + u_i^2 s_i x + t_i)$ for $i = 1, 2$. One can check that the x-coordinates of $\phi_1 \circ \phi_2$ and $\phi_2 \circ \phi_1$ are $u_1^2 u_2^2 x + u_1^2 s_2^2 + s_1^2$ and $u_1^2 u_2^2 x + u_2^2 s_1^2 + s_2^2$ respectively. These are not equal when, for example, $u_1 = 1$, $s_1 = \zeta_3$, $u_2 = \zeta_3$ and s_2 is arbitrary.

9.5.5: The first part of the exercise follows from Theorem 9.3.4. Points $P = (x_P, y_P)$ either satisfy $a_1 x_P + a_3 = 0$ and so $P = -P$ or $a_1 x_P + a_3 \neq 0$ and so y_P is a solution to

$$y^2 + y = F(x)/(a_1 x + a_3)^2.$$

By Exercise 2.14.7 this is soluble if and only if $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(F(x_P)/(a_1 x_P + a_3)^2) = 0$. Hence, every $x_P \in \mathbb{F}_{2^n}$ is such that either $a_1 x_P + a_3 = 0$ (in which case both E and $\widetilde{E}$ have a single point each with x-coordinate x_P) or precisely one of $y^2 + y = F(x_P)/(a_1 x_P + a_3)^2$ and $y^2 + y = F(x_P)/(a_1 x_P + a_3)^2 + \alpha$ has a solution. The result follows.

9.5.6: If $\sigma(\phi^{-1}) \circ \phi = 1 = \phi^{-1} \circ \phi$ for all σ then $\sigma(\phi^{-1}) = \phi^{-1}$ for all σ and ϕ^{-1} is defined over $\Bbbk$.

9.5.8: Follow the method of Lemma 9.5.7; also see Proposition 1 of Hess, Smart and Vercauteren [284].

9.5.9: We have $j(E) = 0$ and the only elliptic curves $\widetilde{E}/\mathbb{F}_2$ with $j(\widetilde{E}) = 0$ in short Weierstrass form are $y^2 + y = x^3 + a_4 x + a_6$ with $(a_4, a_6) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. One can verify by direct calculation that for any pair E_1, E_2 of distinct curves in this form, the isomorphism between them is not defined over $\mathbb{F}_2$.

9.6.2: Let $E_2 : y^2 + H(x)y = F(x)$. If $\phi(x, y) = (\phi_1(x, y), \phi_2(x, y))$ is a morphism then so is $-\phi(x, y) = (\phi_1(x, y), -\phi_2(x, y) - H(\phi_1(x, y))$.

9.6.7: We know that [0] and [1] are isogenies and that the inverse of an isogeny is an isogeny, so we may assume that $n \geq 2$. Lemma 9.6.6 shows that the sum of two isogenies is an isogeny. It follows by induction that $[n] = [n - 1] + [1]$ is an isogeny.

9.6.10: Use the fact that $[2]P = \mathcal{O}_E$ if and only if $P = -P = \iota(P)$. Everything is direct calculation except for the claim that the quartic has distinct roots, which follows by observing that if x_1 is a repeated root then the corresponding point (x_1, y_1) would be singular.

9.6.16: There are many proofs: using differentials; showing that $\ker(\pi_q) = \{\mathcal{O}_E\}$; determining $\Bbbk(E)/\pi_q^*(\Bbbk(E))$. The statement of the degree follows by Lemma 9.6.13.

9.6.20: This is an immediate consequence of Theorem 9.6.18 (the fact that λ is an isomorphism comes from considering degrees). The details are also given in Proposition 12.12 of Washington [626].

9.6.22: We have $(\psi - \widehat{\phi}) \circ \phi = [0]$ and so the result follows by the same argument using degrees as in the proof of Lemma 9.6.11. One can also use Theorem 9.6.18.

9.6.25: This follows since ρ^* is essentially $\rho^{-1}(x,y) = (\zeta_3^{-1}x, y) = (\zeta_3^2 x, y)$.

9.6.29: A point of order 3 means $[2]P = -P$ and the subgroup being rational means either P is defined over $\Bbbk$ or P is defined over $\Bbbk'/\Bbbk$ and $\sigma(P) = -P$ for all non-trivial $\sigma \in \mathrm{Gal}(\Bbbk'/\Bbbk)$. It follows that $\Bbbk'/\Bbbk$ is quadratic. Translating x to 0 gives a point $(0, v)$. Since $\mathrm{char}(\Bbbk) \neq 2$ we assume $E : y^2 = x^3 + a_2x^2 + a_4x + a_6$ and clearly $v^2 = a_6$. The condition $[2](0,v) = (0,-v)$ implies $a_2 = (a_4/2v)^2 = a_4^2/(4a_6)$ and re-arranging gives the first equation. For the twist write $w = a_4/2a_6$, $X = wx$, $Y = w^{3/2}y$ and $A = a_6w^3$. Then

$$X^3+A(X+1)^2 = w^3x^3+(A/a_6^2)(a_6wx+a_6)^2 = w^3(x^3+(A/(w^3a_6^2))((a_4/2)x+a_6)^2 = w^3y^2 = Y^2.$$

It is easy to check that this final equation is singular if and only if $A = 0$ or $27/4$.

9.6.30: See Doche, Icart and Kohel [183].

9.7.6: Let $\phi(x,y) = (\phi_1(x), cy\phi_1(x)' + \phi_3(x))$ be an isogeny from E to $\widetilde{E}$ over a field $\Bbbk$ of characteristic 2. By definition, $X = \phi_1(x)$ and $Y = cy\phi_1(x)' + \phi_3(x)$ satisfy the curve equation $Y^2 + (\widetilde{a}_1X + \widetilde{a}_3)Y = X^3 + \widetilde{a}_2X^2 + \widetilde{a}_4X + \widetilde{a}_6$. Now

$$Y^2 + (\widetilde{a}_1X + \widetilde{a}_3)Y = (cy\phi_1(x)' + \phi_3(x))^2 + (\widetilde{a}_1\phi_1(x) + \widetilde{a}_3)(cy\phi_1(x)' + \phi_3(x))$$

and $\widetilde{a}_1\phi_1(x) + \widetilde{a}_3 = c(a_1x + a_3)\phi_1(x)'$. Hence

$$Y^2 + (\widetilde{a}_1X + \widetilde{a}_3)Y = (c\phi_1(x)')^2(y^2 + (a_1x + a_3)y) + \phi_3(x)^2 + (\widetilde{a}_1\phi_1(x) + \widetilde{a}_3)\phi_3(x).$$

It follows that $\phi_3(x)$ must satisfy the following quadratic equation over $\Bbbk(x)$

$$\phi_3(x)^2+c(\widetilde{a}_1\phi_1(x)+\widetilde{a}_3)\phi_1(x)'\phi_3(x)+(c\phi_1(x)')^2(x^3+a_2x^2+a_4x+a_6)+\phi_1(x)^3+\widetilde{a}_2\phi_1(x)^2+\widetilde{a}_4\phi_1(x)+\widetilde{a}_6 = 0.$$

Since $\Bbbk(x)$ is a field there are at most two possible values for $\phi_3(x)$.

9.8.3: $[3]P = \mathcal{O}_E$ if and only if $[2]P = -P$. Hence, if $P = (x,y)$ is such that $[3]P = \mathcal{O}_E$ then $\lambda^2 = 3x = 0$ where $\lambda = (3x^2 + 2a_2x + a_4)/(2y) = (2a_2x + a_4)/(2y)$. The statements all follow from this.

9.9.5: Use the fact that the dual isogeny of $\phi^2 - [t]\phi + [d]$ is $\widehat{\phi}^2 - [t]\widehat{\phi} + [d]$ since $[\widehat{t}] = [t]$ etc.

9.10.4: For each $y \in \mathbb{F}_p$ there is a unique solution to $x^3 = y^2 - a_6$ and so there are p solutions to the affine equation. Counting the point at infinity gives the result.

9.10.5: Write the curve as $y^2 = x(x^2 + a_4)$. Since -1 is not a square, for any $x \in \mathbb{F}_p$ we have either $x(x^2 + a_4) = 0$ or exactly one of $x(x^2 + a_4)$ and $-x(x^2 + a_4)$ is a square. Hence there are p solutions to the affine equation.

9.10.10: Maintain a pair of the form (t_i, t_{i+1}). So start with $(t_1 = t, t_2 = t^2 - 2q)$. Given (t_i, t_{i+1}) we can compute (t_{2i}, t_{2i+1}) by first computing t_{2i} and then t_{2i+1} using the above formulae. To go from (t_i, t_{i+1}) to (t_{2i+1}, t_{2i+2}) one computes $t_{2i+1} = t_it_{i+1} - q^it$ and $t_{2i+2} = t_{i+1}^2 - 2q^{i+1}$. One can then perform a ladder algorithm (working through the binary expansion of n) as in Lemma 6.3.19.

9.10.11: The first statement is easy to verify by counting points. The second statement follows since if $m \mid n$ then $E_a(\mathbb{F}_{2^m})$ is a subgroup of $E_a(\mathbb{F}_{2^n})$ and so $\#E_a(\mathbb{F}_{2^m}) \mid \#E_a(\mathbb{F}_{2^n})$. Also, if $m \geq 3$ then $\#E_a(\mathbb{F}_{2^m}) > 4$. Hence, it suffices to restrict attention to prime values for n in the third part. The values for E_1 are 5, 7, 11, 17, 19, 23, 101, 107, 109, 113, 163 and the values for E_0 are 5, 7, 9, 13, 19, 23, 41, 83, 97, 103, 107, 131.

9.10.21: The possibilities are $p + 1 \pm 46$ and $p + 1 \pm 60$. One can test which arises by choosing a random point $P \in E(\mathbb{F}_p)$ and computing $[p + 1 - t]P$ for each choice of t. One finds that the orders are $p + 1 + 46, p + 1 - 60, p + 1 - 46, p + 1 - 46$ respectively.

9.10.22: The equation $p = a^2 + ab + b^2$ implies $p = (a - b\zeta_3)(a - b\zeta_3^2)$ and the corresponding value for $\pi + \overline{\pi}$ as in Theorem 9.10.18 is $(2a - b(\zeta_3 + \zeta_3^2)) = 2a + b$. To get the other solutions one notes that the solutions to $x^2 + xy + y^2 = p$ are obtained from (a, b) via the symmetries $(a, b) \mapsto (-a, -b)$; $(a, b) \mapsto (b, a)$ and $(a, b) \mapsto (b, a + b)$ which corresponds to $(a - b\zeta_3) \mapsto \zeta_3(a - b\zeta_3)$. One finds 12 pairs (a, b) and these give rise to the 6 values for t claimed.

9.11.1: The first claim is proved by induction, starting with $t \equiv 0 \pmod{p}$. For the second claim, recall that points in $E[p]$ correspond to roots of the p-th division polynomial and hence must be defined over some finite extension of $\mathbb{F}_q$. But if $E(\mathbb{F}_{q^n})$ has a point of order p then $\#E(\mathbb{F}_{q^n}) \equiv 0 \pmod{p}$.

9.11.10: We have (ignoring for the moment the easy case $P(T) = (T \pm \sqrt{q})^2$) $\frac{1}{q}P(T\sqrt{q}) \mid \Phi_m(T^2) \mid T^{2m} - 1 = (T^m - 1)(T^m + 1)$ in $\mathbb{R}[x]$ for some $m \in \{1, 2, 3, 4, 6\}$. It follows (this requires some work) that $P(T) \mid (T^m - q^{m/2})(T^m + q^{m/2})$ in $\mathbb{Z}[x]$. Hence, if α is a root of $P(T)$ then $\alpha^m = \pm q^{m/2} \in \mathbb{Z}$. Similarly, $\#E(\mathbb{F}_q) = P(1) \mid (q^m - 1)$.

9.11.14: Take any supersingular elliptic curve E_1 over $\mathbb{F}_p$ with $\#E(\mathbb{F}_p) = p + 1$ and let E_2 be a non-trivial quadratic twist.

9.12.6: The statements about (X_1, Z_1) and (X_2, Z_2) are immediate. For the others, substitute X_n/Z_n, X_m/Z_m and X_{m-n}/Z_{m-n} for x_1, x_2 and x_4 in Lemma 9.12.5.

9.12.7: The idea is to store $(X_n, Z_n, X_{n+1}, Z_{n+1})$ (taking $m = n+1$ we have $(X_{m-n}, Z_{m-n}) = (x_P : 1)$ and so the above formulae may be used). The exponentiation is done using a "ladder algorithm" (analogous to Lemma 6.3.19) which computes either $(X_{2n}, Z_{2n}, X_{2n+1}, Z_{2n+1})$ or $(X_{2n+1}, Z_{2n+1}, X_{2n+2}, Z_{2n+2})$. Every step is therefore a doubling and an addition. See Algorithm 13.35 of [16]. For the improved formulae see [436] or Section 13.2.3.a of [16].

9.12.8: One has

$$[2](x, y) = (B\lambda^2 - 2x - a_2, \star)$$

where $\lambda = (3x^2 + 2a_2x + a_4)/(2By)$ and where the formula for the y-coordinate is not relevant here. Solving $[2](x, y) = (0, 0)$ is solving $B\lambda^2 = 2x + a_2$ and so $0 = (3x^2 + 2a_2x + a_4)^2 - 4(x^3 + a_2x^2 + a_4x)(2x + a_2) = (x^2 - a_4)^2$. Hence, $x = \pm\sqrt{a_4}$ and the y-values are $\pm\sqrt{x(x^2 + a_2x + a_4)/B}$ as stated.

9.12.10: The point $(1, 1)$ on $(A + 2)y^2 = x(x^2 + Ax + 1)$ has order 4.

9.12.17: Homogenising gives the equation $(ax^2 + y^2)z^2 = z^4 + dx^2y^2$ and setting $z = 0$ leads to the equation $dx^2y^2 = 0$ and hence the two points $(1 : 0 : 0)$ and $(0 : 1 : 0)$. To show that $(1 : 0 : 0)$ is singular it suffices to set $x = 1$ and show that the point $(0, 0)$ on $(a + y^2)z^2 = z^4 + dy^2$ is singular, which is easy. The other case is similar.

9.12.23: Writing the curve equation as $x^2 = (1 - y^2/(a - dy^2)$ the quadratic twist is $ux^2 = (1 - y^2)/(a - dy^2)$ which gives the result.

9.12.24: Follows directly from Theorem 9.12.12.

9.12.27: Irreducibility follows since $y^2 - g(x)$ must factor as $(y + g_1(x))(y + g_2(x))$ and it follows that $g_2(x) = -g_1(x)$ (which is prohibited by $a^2 \neq d$). The point $(0 : 1 : 0)$ on $y^2z^2 = x^4 + 2ax^2z^2 + z^4$ is singular. An affine singular point on C must satisfy $y = 0$, $4x(dx^2 + a) = 0$ and $dx^4 + 2ax^2 + 1 = 0$, which again is impossible if $a^2 \neq d$.

The birational map is easy to verify: $2Y^2 = X(2X/x^2)$ and $X(X^2 - 2aX + (a^2 - d)) = X(a^2 + 2a(y + 1)/x^2 + (y + 1)^2/x^4 - 2a^2 - 2a(y + 1)/x^2 + a^2 - d)$ and the result follows by substituting for y^2. Finally, the discriminant of $X^2 - 2aX + (a^2 - d)$ is $4d$ so when d is a square in $\Bbbk$ then there are three points $(0, 0), (a \pm \sqrt{d}, 0)$ of order 2.

9.12.29: Let P be a point of order 2 and move P to $(0, 0)$ so that the curve is $Y^2 = X(X^2 + Ax + B)$. Write $a = -A/2$ and $d = a^2 - B$ so that a twist of the curve is in the form of equation (9.16). The result follows from Exercise 9.12.27.

Chapter 10: Hyperelliptic Curves

10.1.3: When we substitute $z = 0$ we get the equation $H_{D-1}x^{D-1}y = F_D x^D$ where H_{D-1} (respectively, F_D) is the coefficient of the monomial x^{D-1} (resp., x^D) in $H(x)$ (resp., $F(x)$). The possible solutions are $x = 0$, giving the point $(x : y : z) = (0 : 1 : 0)$, or $H_{D-1}y = F_D x$ giving the point $(H_{D-1} : F_D : 0)$.

For the second part, we have $H_D = 0$ and so the only point at infinity is $(0 : 1 : 0)$. Making the curve affine by setting $y = 1$ yields $z^{D-2} + z^{D-1}H(x/z) = z^D F(x/z)$ and when $D > 3$ every monomial has degree at least 2. Hence the two partial derivatives vanish.

10.1.5: Make the change of variable $y = Y + x^3 + 1$. Then

$$y^2 + (1 - 2x^3)y = Y^2 + 3Y - x^6 + x^3 + 2$$

and so the affine curve is isomorphic to $Y^2 + 3Y = x - 1$. This curve is birational to $\mathbb{P}^1$ (taking the map $(x, Y) \mapsto Y$) and hence, by Theorem 8.6.1 and Definition 8.6.2, the curve has genus 0.

10.1.13: If $H_0 = F_1 = F_0 = 0$ then $(0, 0) \in C(\Bbbk)$ is singular. (This also follows since $C^\dagger$ is birational to C and the genus is a birational invariant.)

10.1.20: 1. One way is $(x, y) \mapsto (y : x^d : x^{d-1} : \cdots : x : 1)$. The other way is $(Y : X_d : X_{d-1} : \cdots : X_1 : X_0) \mapsto (Y, X_d/X_{d-1})$.

2. The statement about ι is clear.

3. Identifying $X_i = x^i z^{d-i}$ it follows that points at infinity (i.e., with $z = 0$) have $X_i = 0$ when $0 \leq i < d$.

4. Set $X_d = 1$ to give the point $(Y, X_{d-1}, \ldots, X_0) = (\alpha, 0, \ldots, 0)$ on the affine curve

$$Y^2 + H(X_{d-1}, \ldots, X_0)y = F(X_{d-1}, \ldots, X_0)$$

together with various other equations, including $X_i = X_{\lceil (d+i)/2 \rceil} X_{\lfloor (d+i)/2 \rfloor}$ for $0 \leq i \leq d-2$. One must determine the Jacobian matrix (see Theorem 7.1.12 and Corollary 7.1.13) at the point $(\alpha, 0, \ldots, 0)$. The number of variables is $d+1$ and the dimension is 1, so we need to show that the rank is d. Note that each of the $d-1$ equations $X_i = X_{\lceil (d+i)/2 \rceil} X_{\lfloor (d+i)/2 \rfloor}$ yields a row $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ in the Jacobian matrix (with the 1 corresponding to variable X_i). Hence, the rank is at least $d - 1$. The curve equation yields the row

$$(2\alpha + H_d, H_{d-1}\alpha - F_{2d-1}, 0, \ldots, 0)$$

in the Jacobian matrix, so to complete the proof we must show that either $2\alpha + H_d \neq 0$ or $H_{d-1}\alpha + F_{2d-1} \neq 0$. Note that at least one of $\{F_{2d}, F_{2d-1}, H_d\}$ is non-zero, otherwise we would replace d by $d - 1$.

If $\mathrm{char}(\Bbbk) \neq 2$ and both terms are zero then $\alpha = -H_d/2$ which implies that $F_{2d} = -(H_d/2)^2$ and $F_{2d-1} = -H_d H_{d-1}/2$ which violates the condition of Lemma 10.1.8. If $\mathrm{char}(\Bbbk) = 2$ then we must consider the three cases in Lemma 10.1.6. The first case is $\alpha = 0$ and $F_{2d-1} \neq 0$ and so $H_{d-1}\alpha + F_{2d-1} \neq 0$. The second and third cases have $H_d \neq 0$ and so $2\alpha + H_d = H_d \neq 0$.

10.1.23: The statement $v_\infty(x) = -2$ follows from the fact that $v_{\rho(\infty)}(Z) = 2$ as in Lemma 10.1.21. The second claim follows since $\infty = (1 : 0 : 0)$ and $1/x = c(y/x^d)^2$ for some constant c, in which case $c(y/x^d) = x^{d-1}/y$. The third claim is immediate from Lemma 10.1.22.

10.1.25: Write $d = g + 1$. Note that

$$v_\infty(y) = v_\infty((y/x^d)x^d) = v_\infty(y/x^d) + dv_\infty(x).$$

Hence, if C is in ramified model then $v_\infty(y) = 1-2d = -(2d-1) = -(2g+1)$. Similarly, if C is in split model then $v_{\infty^+}(y) = 0-d = -(g+1)$.

10.1.26: $v_P(A-yB)+v_P(A-(-y-H)B) = v_P(A^2+HAB-FB^2)$ which is $2e$ when $P=\iota(P)$ and is e otherwise.

10.1.27: A uniformizer at ∞^+ is X_{d-1}/X_d if $\iota(\infty^+) \neq \iota(\infty^+)$ and $Y/X_d - \alpha^+$ otherwise.

10.1.28: The choice of the leading coefficient already implies $\deg(G^+(x)^2+H(x)G^+(x)-F(x)) \leq 2d-1$. Write $G^+(x) = \alpha^+x^d + G_{d-1}x^{d-1} + \cdots + G_0$. One can solve a linear equation in G_{d-1} so that the degree of $G^+(x)^2 + H(x)G^+(x) - F(x)$ is at most $2d-2$. Continuing this way one solves a linear equation for each variable G_i to lower the degree to at most $d+i-1$. The result for $G^-(x)$ follows by starting with α^- instead of α^+.

10.1.29: First note that $(y-G^+(x))/x^d = (y/x^d - \alpha^+) + (1/x)G(1/x)$ for some $G(x) \in \Bbbk[x]$ and so is zero at ∞^+. Since $\alpha^+ \neq \alpha^-$ it follows that $(y-G^+(x))/x^d$ is not zero at ∞^-. Also, $v_{\infty^-}(y-G^+(x)) \geq \max\{v_{\infty^-}(y), v_{\infty^-}(G^+(x))\} = -d = -(g+1)$ Now, $\deg(G^+(x)^2 + H(x)G^+(x) - F(x)) \leq d-1 = g$ so the affine effective divisor $\operatorname{div}(y-G^+(x)) \cap \mathbb{A}^2$ has degree at most g. Since $\operatorname{div}(y-G^+(x))$ must have degree 0 and $v_{\infty^-}(y-G^+(x)) = -(g+1)$ it follows that $v_{\infty^+}(y-G^+(x)) \geq 1$.

One can also complete the proof directly from the equation

$$(y-G^+(x))(-y-H(x)-G^+(x)) = G^+(x)^2 + H(x)G^+(x) - F(x),$$

the right hand side of which is a polynomial of degree at most g. It follows that the negative integer $v_{\infty^+}(y-G^+(x)) + v_{\infty^+}(-y-H(x)-G^+(x))$ is at least $-g$.

10.2.4: This is similar to Exercise 9.5.5.

10.2.5: First take a root α of $F(x)$ and move it to ∞. Then perform a change of variable on the remaining roots so that one is 0, another is 1 and change y so that the polynomial is monic.

10.2.7: Follows immediately from Theorem 10.2.1.

10.3.8: If $P = (x_P, y_P) \neq \iota(P)$ and $(x-x_P)^{e_P} \| u(x)$ then the claim is immediate from $v_P(u(x)) = e_P$ and $v_P(y-v(x)) = v_P((y-v(x))(-y-H(x)-v(x)) = v_P(v(x)^2 + H(x)v(x) - F(x)) \geq v_P(u(x))$. If $P = \iota(P) = (x_P, y_P)$ then $y - y_P$ is a uniformizer at P. We have $v_P(u(x)) = v_P(x-x_P) = 2$ and since $v(x) = y_P + (x-x_P)G(x)$ for some polynomial $G(x) \in \overline{\Bbbk}[x]$ we have $v_P(y-v(x)) = v_P((y-y_P) - (x-x_P)G(x)) = 1$.

10.3.11: The first statement follows from the following two facts: (1) if D is effective then so is $\sigma(D)$; (2) $\sigma(\iota(P)) = \iota(\sigma(P))$. The second follows since $\sigma(D) = \sum_P n_P(\sigma(P))$ and if $u(x) = \prod_P (x-x_P)^{n_P}$ then $\sigma(u(x)) = \prod_P (x-\sigma(x_P))^{n_P} = \prod_P (x-x_{\sigma(P)})^{n_P}$ etc. The third statement follows from the second and the uniqueness of the Mumford representation.

10.3.12: Since the $u_i(x)$ are co-prime the composition does not have any special cases and the equality of divisors is clear. This can also be seen by considering points over $\overline{\Bbbk}$.

10.3.15: Note that $\iota(6,4) = (6,4)$ and so $2D_1 \equiv 2(0,4)$. The answers are $(x^2+5x, 4), (x^2-x, -3x+4), (x^2, 10x+4), (x^4+4x^3+6x^2, 4+10x+6x^2+3x^3)$.

10.3.16: All polynomials are of degree $O(m)$ and multiplication, computing the extended gcd, and division can be performed in $O(m^2)$ field multiplications.

10.4.2: The inverse of $D = \sum_P n_P(P)$ is $\iota_*(D) = \sum_P n_P(\iota(P))$. Hence, since the points P in the divisor class represented by $(u(x), v(x))$ are of the form $(x_i, v(x_i))$ where $u(x_i) = 0$ the corresponding points for the inverse of D are $(x_i, -v(x_i) - H(x_i))$. The result follows.

10.4.5: Just re-arrange the equation $D_3 - d_3(\infty) \equiv D_1 - d_1(\infty) + D_2 - d_2(\infty)$.

10.4.8: One has $v_{\infty^-}(y-v^\ddagger(x)) = v_{\infty^-}(y-G^+(x)) = -d$. The affine part of the second claim is already proved in the proof of Lemma 10.4.6. Since $\deg(\operatorname{div}(y-v^\ddagger(x))) = 0$ it

follows that $v_{\infty^+}(y - v^\ddagger(x)) = -(d_u + d_{u^\dagger} - d)$. (There is also a direct proof of this fact.)

10.4.15: This is very similar to the proofs of Lemma 10.4.6 and Lemma 10.4.11.

Let $d_u = \deg(u(x))$. The leading $d - (d_u - 1)$ coefficients of $v^\ddagger(x)$ agree with $G^+(x)$ and hence $\deg(v^\ddagger(x)^2 + H(x)v^\ddagger(x) - F(x)) \leq 2d - (d - (d_u - 1)) = d + d_u - 1$. It follows that $\deg(u^\dagger(x)) \leq d - 1$. Since $v_{\infty^-}(y - v^\ddagger(x)) = -d$ and since $\mathrm{div}(y - v^\ddagger(x))$ has degree 0 one has $v_{\infty^+}(y - v^\ddagger(x)) = -(\deg(u(x) + \deg(u^\dagger(x)) - d)$. The result follows from the analogue of equation (10.17).

10.4.17: Clearly,

$$\begin{aligned} &\iota_*(D + n(\infty^+) + (g - d_u - n)(\infty^-) - D_\infty) \\ &= \iota_*(D) + n(\infty^-) + (g - d_u - n)(\infty^+) - \iota_*(D_\infty)) \end{aligned}$$

When g is even then $\iota_*(D_\infty) = D_\infty$ and the result is immediate by Exercise 10.4.2. When g is odd note that $\iota_*(D_\infty) = D_\infty + (\infty^-) - (\infty^+)$ and so

$$\begin{aligned} &\iota_*\left(D + n(\infty^+) + (g - \deg(u(x)) - n)(\infty^-) - D_\infty\right) \\ &= \iota_*(D) + (n-1)(\infty^-) + (g - \deg(u(x)) - n + 1)(\infty^+) - D_\infty. \end{aligned}$$

If $n = 0$ then this divisor is not effective and so it is necessary to perform composition and reduction at infinity.

10.4.20: This is the special case $D_1 = D_2 = 0, n_1 = 0, n_2 = \lceil g/2 \rceil + 1$ of Theorem 10.4.19, since $D = (1, 0, 0)$ is principal.

10.4.21: This follows from Theorem 10.4.19.

10.4.22: Let $\rho_P(x, y) = (1/(x - x_P), y/(x - x_P)^{g+1})$ map P to ∞^+. If $\mathrm{div}(f(x, y)) = n((P) - (\iota(P))$ then $\mathrm{div}(f \circ \rho_P^{-1}) = n((\infty^+) - (\infty^-))$.

10.5.2: By Lemma 8.1.3 and Theorem 8.2.7 we know ϕ is surjective. Hence, for any $P \in E(\overline{\Bbbk})$ choose any point $P' \in C(\overline{\Bbbk})$ such that $\phi(P') = P$ and let $P'' \in C(\overline{\Bbbk})$ be such that $\phi(P'') = \mathcal{O}_E$. Then $\phi_*((P') - (P'')) = (P) - (\mathcal{O}_E)$ as required. The second statement follows since the kernel of ϕ^* is contained in the kernel of $\phi_*\phi^* = [\deg(\phi)]$.

10.6.3: See Mumford [445] pages 3.31-3.32.

10.7.7: An elegant proof using power series is given in Exercises 17.19 and 17.20 of Shoup [556].

10.7.8: One has $t_1 = 0, t_2 = -42, t_3 = 0$. Hence $a_1 = 0, a_2 = 21$ and $a_3 = 0$. It follows that $\#\mathrm{Pic}^0_{\mathbb{F}_7}(C) = 512$.

10.7.14: First, one needs to count various types of points in $C(\mathbb{F}_q)$ and $C(\mathbb{F}_{q^2})$. Note that there is a single point at infinity ∞ on $C(\mathbb{F}_{q^n})$ for all $n \in \mathbb{N}$. Write m for the number of roots of $F(x)$ in $\mathbb{F}_q$. Let $u = \frac{1}{2}(N_1 - 1 - m) = \frac{1}{2}(q - t_1 - m)$. Then there are u values for $x_P \in \mathbb{F}_q$ such that there are points $P = (x_P, y_P) \in C(\mathbb{F}_q)$ with $P \neq \iota(P)$. Hence $\#C(\mathbb{F}_q) = 1 + m + 2u$. There are q possible values for $x_P \in \mathbb{F}_q$. We have shown that $m + u$ of them arise as x-coordinates of points in $C(\mathbb{F}_q)$ (i.e., $F(x_P)$ is a quadratic residue for those values). For the remaining $q - m - u$ values x_P it follows that $F(x_P)$ is not a square in $\mathbb{F}_q$. But $F(x_P)$ is a square in $\mathbb{F}_{q^2}$ so x_P must arise as the x-coordinate of a point in $C(\mathbb{F}_{q^2})$. Hence there are $q - m - u = \frac{1}{2}(q + t_1 - m)$ such values for x_P. Hence, there are $\frac{1}{2}(q - m + t_1)$ points $P = (x_P, y_p) \in C(\mathbb{F}_{q^2})$ with $x_P \in \mathbb{F}_q$, $y_P \notin \mathbb{F}_q$ and $P \neq \iota(P)$.

Exercise 10.4.4 classifies divisor classes so it is sufficient to count the number of representatives for each case. Case 1 gives N_1 divisor classes.

For case 2, let m be the number of roots of $F(x)$ in $\mathbb{F}_q$. Therefore, there are $m + 1$ points $P \in C(\mathbb{F}_q)$ such that $P = \iota(P)$. Hence, case 2 gives $N_1 - 1 - m$ divisor classes.

For case 3, we first count the set of pairs (P, Q) such that $P, Q \neq \infty$, and $Q \notin \{P, \iota(P)\}$. There are $N_1 - 1$ choices for P, and for each there are either $N_1 - 2$ or $N_1 - 3$ choices for Q (depending on whether $P = \iota(P)$ or not). Finally, since in this case we always have

$P \neq Q$, the number of choices for $\{P, Q\}$ is half the number of pairs (P, Q). Hence the total number of divisor classes in case 3 is

$$\tfrac{1}{2}((N_1 - m - 1)(N_1 - 3) + m(N_1 - 2)).$$

We finally consider case 4 of Exercise 10.4.4. Note that $\mathbb{K} = \mathbb{F}_{q^2}$ is the unique quadratic extension of $\mathbb{F}_q$. We have $P = \sigma(P)$ if and only if $P \in C(\mathbb{F}_q)$. Hence, the number of choices for $\{P, \sigma(P)\}$ is $\frac{1}{2}(N_2 - N_1)$. From these choices we must subtract the number of pairs $\{P, \sigma(P)\}$ such that $\sigma(P) = \iota(P)$. This happens when $x_P \in \mathbb{F}_q$ but $y_P \notin \mathbb{F}_q$ and by the above argument there are $\frac{1}{2}(q - m + t_1)$ such points. Hence, the total number of divisor classes in case 4 is

$$\tfrac{1}{2}(q^2 + 1 - t_2 - (q + 1 - t_1) - (q - m + t_1)) = \tfrac{1}{2}(q^2 - 2q - t_2 + m).$$

Adding up all the cases gives

$$(q + 1 - t_1) + (q - t_1 - m) + \tfrac{1}{2}(q^2 - q(2t_1 + 2) + t_1^2 + 2t_1 + m) + \tfrac{1}{2}(q^2 - 2q - t_2 + m)$$

which simplifies to

$$q^2 - t_1 q + (t_1^2 - t_2)/2 - t_1 + 1$$

as required.

10.9.4: This proof follows the same layout as Lemma 9.11.8 and Corollary 9.11.9. The details are given in [218].

10.9.6: We have $p \mid a_1$ and $p \mid a_2$, which is equivalent to the conditions for supersingularity.

10.9.7: First, simply show that $\#C(\mathbb{F}_{2^{nk}})$ is always odd and so t_1 and t_2 (notation as in Lemma 10.7.6) are even. Hence a_1, a_2 are even (for the details of this calculation see Lemma 2 of [218]). The result then follows from part 1 of Theorem 10.9.5 and Exercise 10.9.6.

Chapter 11: Basic Algorithms for Algebraic Groups

11.1.2: If a is odd at any stage then the new value $(a - a_i) \equiv 0 \pmod 4$ so the non-zero coefficient a_i is followed by $a_{i+1} = 0$.

11.1.6: $100 = 10\overline{1}00100$ etc.

11.1.9: The largest possible NAF of length $l + 1$ is $2^l + 2^{l-2} + 2^{l-4} + \cdots$ and taking $d_l = 2^{l-2} + 2^{l-4} + \cdots + 1$ when l is even and $d_l = 2^{l-2} + 2^{l-4} + \cdots + 2$ when l is odd gives the formula for d_l. If $a_l = -1$ then $a = -2^l + d_l < 0$, which is a contradiction. The minimal value for a is therefore $2^l - d_l$. One can check that $2^l + d_l + 1 = 2^{l+1} - d_{l+1}$. Finally, $2^{l+1}/3 < 2^l - d_L \leq a$ so $l \leq \log_2(a) - \log_2(2/3) \approx \log_2(a) + 0.585$. Since the binary expansion of a requires $\lfloor \log_2(a) \rfloor + 1$ bits the result follows.

11.1.15: Write N_k for the number of NAFs of length k (so $k = l + 1$ in the above notation). We have $N_1 = 3$ and $N_2 = 5$. Also $N_{k+1} = N_k + 2N_{k-1}$ from which the formula follows.

11.1.21: Essentially the same proof as Lemma 11.1.8. See Proposition 2.1 or Muir and Stinson [442].

11.1.23: Proposition 2.4 of Muir and Stinson [442].

11.1.26: 300070007 and $100\overline{1}000\overline{1}00\overline{1}$.

11.2.2: There are 2^n values $u_{b_1,\ldots,b_n}$ to compute: $n + 1$ of them come for free (corresponding to the vectors $(b_1, \ldots, b_n)$ that are all zero or have only one non-zero entry) and, if the computation is organised appropriately, each of the others can be obtained from

an earlier value using one additional multiplication. The second claim of the exercise is clear.

11.2.3: For windows of length w one must precompute all $u_{b_1,\ldots,b_n} = \prod_{i=1}^{n} g_i^{b_i}$ for $0 \leq b_i < 2^w$ under the constraint that at least one of the b_i is odd. The number of such values is $2^{nw} - 2^{n(w-1)}$ of which n come for free. Hence the precomputation requires $2^{nw} - 2^{n(w-1)} - n$ multiplications (actually, when $w > 1$ the computation can be arranged so that some of these are squarings). See Yen, Laih and Lenstra [638] for details.

11.2.4: See Algorithm 3.51 of [274].

11.3.1: Given $P = (x_P, y_P)$ compute a solution to $\lambda_Q^2 + \lambda_Q = x_P + a_2$, determine whether the corresponding value of x_Q^2 satisfies the trace condition (and if not, replace λ_Q by $\lambda_Q + 1$), compute a square root (which is easy in finite fields of characteristic two), and solve for y_Q. See Knudsen [342] for details.

11.3.4: The cost depends on the number of non-zero values for a_i, which is the weight. The number of elliptic curve additions is one less than the weight. We ignore the cost of computing the Frobenius maps.

11.3.10: Let $n_0 = 107$ and $n_1 = 126$ so $a_0 = -1$ and the new values for (n_0, n_1) are $(180, -54)$. The Frobenius expansion is

$$-1 - \pi^5 - \pi^7 - \pi^9 + \pi^{11} + \pi^{13} + \pi^{16}.$$

11.3.13: $P \in E(\mathbb{F}_{p^m})$ if and only if $(\pi^m - 1)P = \mathcal{O}_E$.

11.3.16: It is clear that P is in the kernel of the endomorphism corresponding to this polynomial. Now, note that $(x^{19} - 1)/(x - 1) \equiv -457 + 314x \pmod{x^2 - x + 2}$ and that $-457 + 314\pi$ has norm r. Rounding $n(-457 + 314\bar{\pi})/r$ gives $-67 - 148\pi$ from which we get

$$n - (-67 - 148\pi)(-457 + 314\pi) = -107 - 126\pi$$

as was found using the GLV method.

11.3.19: One first precomputes all $\sum_{j=0}^{w-1} [a_j]\pi^j(P)$. To do this one computes all $[a_j]P$, which needs one doubling and (when q is odd) $(q-5)/2$ additions or (when q is even) $q/2 - 2$ additions. One computes $[a_j]\pi^j(P)$ as $\pi^j([a_j]P)$ and we ignore the cost of this. To compute all the terms requires at most q^{w-1} additions. Hence, the total cost of precomputation is at most $q/2 + q^{w-1}$ group operations.

The exponentiation itself then requires at most $(l+1)/w$ additions.

11.3.20: First compute integers a_0', a_1' such that $a_0' + a_1'\pi \equiv a(\pi) \pmod{\pi^2 - t\pi + q}$ (this can be done efficiently using Lucas sequences). Then find the eigenvalue λ for π by computing $\gcd(x^2 - tx + q, x^m - 1)$. Finally, compute $a \equiv a_0' + a_1'\lambda \pmod{r}$. See Brumley and Järvinen [112].

11.3.23: Analogous to the method at the end of Section 11.3.2.

11.3.24: The first claims are immediate since $\#E(\mathbb{F}_{p^2}) = (p + 1 - t)(p + 1 + t)$. The map ψ is an endomorphism since ψ, π and ϕ^{-1} are endomorphisms (alternatively, because ψ is a morphism fixing $\mathcal{O}_{E'}$). One can directly compute that $\psi^2 = [-1]$. The formula $\psi^2 - [t]\psi + [p] = 0$ follows since π satisfies this formula and ψ is just a conjugation of π. Since $r^2 \nmid \#E'(\mathbb{F}_{p^2})$ it follows that $\psi(P) \in \langle P \rangle$ and so λ exists. The formula for λ follows from solving the simultaneous equations $\lambda^2 \equiv -1 \pmod{r}$ and $\lambda^2 - t\lambda + p \equiv 0 \pmod{r}$.

11.4.7: Let S be the set of integer k-tuples $(a_1, \ldots, a_k)$ with $0 \leq a_i < m_i$ for $1 \leq i \leq k$. Let T be the set of $(a_1, \ldots, a_k) \in S$ in such that the value of equation (11.2) is 1. Then $\prod_i g_i^{a_i} = \prod_i g_i^{b_i}$ if and only if the vector with entries $a_i - b_i \pmod{m_i}$ for $1 \leq i \leq k$ lies in T. It follows that $\#G = \#S/\#T$ and that the method samples uniformly from the group G.

11.4.9: A simple solution is to choose $a, b \in \mathbb{F}_q$ uniformly at random and reject if $a^2 - ab + b^2 = 1$. This happens with probability roughly $1/q$. One can then use the

decompress map to obtain an element of $G_{6,q}$. This process misses several points in $G_{6,q}$, such as the element 1 and the element θ^2 of order 3 (which corresponds to the point $(0,0,0)$ not appearing in the image of the map g in Example 6.4.4).

11.4.10: The expected number of trials to get a soluble quadratic equation is roughly 2 (precisely, there are at least $(q-2\sqrt{q})/2$ values for x_0 such that $y^2+H(x_0)y-F(x_0)$ is a square in $\mathbb{F}_q$ and so the probability to be successful is at least $(q-2\sqrt{q})/(2q) \approx 1/2$). One solves the quadratic equation either by computing square roots using Exercise 2.14.3 (expected complexity of $O(\log(q)^2 M(\log(q)))$ bit operations) or, in the case of characteristic 2, by Lemma 2.14.8 (complexity $O(\log(q)^3)$ bit operations). The goto statement in line 14 occurs with probability at most $\frac{1}{2}\frac{3}{q} < \frac{1}{2}$ and so the expected number of repeats is less than 2. Hence, the algorithm for finding random points on elliptic curves has expected complexity $O(\log(q)^4)$ bit operations.

11.4.12: See Section 5 of Shallue and van de Woestijne [545].

11.4.13: The first claim is checked by an elementary calculation. For the second claim equate $y - ux = v = (3A - u^4)/(6u)$ and compute the roots in $\mathbb{F}_q$ of the polynomial $6u(y - ux) - (3A - u^4)$.

11.4.17: Set $l' = l/n$ and use the method of Example 11.4.2 n times with l'-bit strings.

11.5.2: When the algorithm ends, the points P and Q generate the subgroup of $E(\mathbb{F}_q)$ of order N_0. One simply adds to P a point R of order N_1 (found using an analogue of Algorithm 5).

11.5.3: If the group is cyclic then the probability that a random element is a generator is $\varphi(l^e)/l^e = 1 - 1/l$. Hence, the probability that among two elements at least one of them is a generator is $1 - (1 - (1 - 1/l))^2 = 1 - 1/l^2$.

If the group is not cyclic, suppose its structure is $(\mathbb{Z}/l^f\mathbb{Z}) \times (\mathbb{Z}/l^{e-f}\mathbb{Z})$ where $1 \leq f \leq e - f$. Consider the projection of points $\{P, Q\}$ to elements $\{(p_1, p_2), (q_1, q_2)\}$ of this group. The points generate $E(\mathbb{F}_q)[l^e]$ if and only if the vectors $\{(p_1, p_2), (q_1, q_2)\}$ are a basis for the vector space $(\mathbb{Z}/l\mathbb{Z})^2$. Hence, one needs the first vector to be non-zero (this occurs with probability $1-1/l^2$) and the second vector to not lie on the line corresponding to the first vector (this occurs with probability $1 - 1/l$).

The probability of success overall is therefore the product for all primes l_i, which gives the stated formula. Finally, $\varphi(N_0)/N_0 > 1/(3\log(\log(N_0))) = O(1/\log(\log(q)))$ and

$$\prod_{l|N_0} \left(1 - \tfrac{1}{l^2}\right) > \prod_{l} \left(1 - \tfrac{1}{l^2}\right) = \zeta(2)^{-1} = 6/\pi^2 = O(1)$$

where the second product is over all primes l. Hence one expects $O(\log(\log(q)))$ iterations overall.

For generalisations and more detail see Section 6.1 of Miller [429].

11.6.2: Use point halving to attempt $r - 1$ halvings and then check whether the point can be halved further over $\mathbb{F}_q$ (this is only possible if the original point has odd order).

11.7.2: First, we know that $\mathrm{Tr}(x_P) = \mathrm{Tr}(a_2) = 0$. We may assume that $x_P \neq 0$ and so

$$(y_P/x_P)^2 + (y_P/x_P) = x_P + a_2 + a_6/x_P^2.$$

Hence, $\mathrm{Tr}(a_6/x_P^2) = 0$ and so $\mathrm{Tr}(\sqrt{a_6}/x_P) = 0$. It is immediate that $(0, \sqrt{a_6})$ has order 2, and since $\mathrm{Tr}(0) = 0$ it follows that it can be halved in $E(\mathbb{F}_{2^n})$. The statement about $(x_P, y_P) + (0, \sqrt{a_6})$ follows from the formulae for the group law. Since (x_P, y_P) and $(0, \sqrt{a_6})$ can both be halved, their sum can also be halved; this also follows directly since $\mathrm{Tr}(\sqrt{a_6}/x_P) = 0$.

The receiver gets an $n - 1$ bit string representing either x_P or $\sqrt{a_6}/x_P$. The receiver then computes the corresponding value $z \in \{x_P, \sqrt{a_6}/x_P\}$. One can verify that, in both

cases, $\mathrm{Tr}(z + a_2 + a_6/z^2) = 0$. Hence, one can solve $u^2 + u = z + a_2 + a_6/z^2$ for $u \in \mathbb{F}_{2^n}$ such that $\mathrm{Tr}(u) = 0$. If $z = x_P$ then u is y_P/x_P. If $z = \sqrt{a_6}/x_P$ then u is

$$\frac{y_P}{x_P} + \frac{\sqrt{a_6}}{x_P} + x_P + 1,$$

which does satisfy $\mathrm{Tr}(u) = 0$. In both cases, one determines the corresponding value y as $y = uz$.

It remains to determine the two cases. Suppose $2^i \| \#E(\mathbb{F}_{2^n})$. Then P can be halved at least i times while $(0, \sqrt{a_6})$ can only be halved $i-1$ times. It follows that $P + (0, \sqrt{a_6})$ can only be halved $i-1$ times. Hence, if one can repeatedly halve (z, y) at least i times then set $x_P = z$ and $y_P = x_P u$. Otherwise, set $x_P = \sqrt{a_6}/z$ and solve for y_P subject to $\mathrm{Tr}(y_P/x_P) = 1$. For further discussion see King [339].

11.7.3: Use action of 2-power Frobenius on x_P (represented using a normal basis) so that the least significant bits are a predictable bit pattern (for example, the longest run of ones followed by a zero). Since the expected length of the longest run of ones is roughly $\log_2(n)$ one can expect to save this many bits. One can also combine this method with the Seroussi trick of Example 11.7.1. For details see Eagle, Galbraith and Ong [188].

Chapter 12: Primality Testing and Integer Factorisation using Algebraic Groups

12.1.4: See Section 3.6 of Crandall and Pomerance [162].

12.1.5: (Gordon [262]) Let $E : y^2 = x^3 + x$ which has $N + 1$ points over $\mathbb{Z}/N\mathbb{Z}$ when N is prime. So choose a random point $P \in E(\mathbb{Z}/N\mathbb{Z})$ (by choosing a random x and computing $y = \sqrt{x^3 + x}$ using the Tonelli-Shanks algorithm) and compute $[N + 1]P$ and see if it is $\mathcal{O}_E$. If anything goes wrong then N is not prime.

12.1.6: If N is prime then the elliptic curve $E : y^2 = x^3 + a_4 x$ has $N + 1 \pm 2a$ points for some integer a such that $p = a^2 + b^2$ (there are four possible group orders); see Example 9.10.20. Hence, given N one can run the Cornacchia algorithm to compute a and b, choose a random point $P \in E(\mathbb{Z}/N\mathbb{N})$, and test whether $[N + 1 \pm 2a]P = \mathcal{O}_E$; if anything goes wrong then one deduces that N is composite.

12.1.8: Since $N - 1 = 2^4 \cdot 35$ the Miller-Rabin sequence for the base 2 is

$$263, \ \ 166, \ \ 67, \ \ 1, \ \ 1.$$

The fact that $67 \not\equiv -1 \pmod{561}$ implies that 561 is composite.

Indeed, the non-trivial solution to $x^2 \equiv 1 \pmod{N}$ leads to a partial factorisation of N via $x^2 - 1 = (x + 1)(x - 1) \equiv 0 \pmod{N}$. In this case, $\gcd(67 - 1, 561) = 33$. Unfortunately the Miller-Rabin test is not a good factoring algorithm since the condition $a^{N-1} \not\equiv 1 \pmod{N}$ is usually not satisfied.

12.2.4: Choose q, then find x_0 such that $\Phi_k(x_0) \equiv 0 \pmod{q}$, then choose $p = x_0 + lq$ for suitable l. See Section 3.1 of Lenstra and Verheul [375].

12.2.5: A solution is Gordon's algorithm; see Algorithm 4.53 of [418].

12.2.7: Theorem 4.1.1 of [162].

12.2.9: Theorem 4.1.3 of [162].

12.2.10: Generate a random prime q together with certificate (using, for example, Maurer's algorithm [403]) and determine if $p = 2q + 1$ is prime. It is easy to form a certificate for p.

12.3.6: Since there are $B/\log(B)$ primes we can save a $\log(B)$ term as long as one can find the next prime in $O(\log(B)M(\log(N)))$ bit operations; in practice this is easily done (e.g., using a sieve or Miller-Rabin).

12.3.7: List the primes $B < Q_1 < Q_2 < \cdots < Q_k \leq B$ and use the fact that $b^{Q_{i+1}} \equiv b^{Q_i} b^{Q_{i+1}-Q_i} \pmod{N}$. One precomputes all the increments $b^{Q_{i+1}-Q_i} \pmod{N}$.

12.3.8: Apply Theorem 2.16.1.

12.3.9: See Williams [634] (note that he credits Guy and Conway for the suggestion).

12.5.1: Exercise 2.2.9 showed that one can determine if N is a prime power (and factor it) in polynomial time. So suppose N has at least two distinct prime factors. If $p^2 \mid N$ then either $p < N^{1/3}$ or $N/p^2 < N^{1/3}$. Use the Pollard-Strassen method with $B = \lceil N^{1/6} \rceil$ to find all prime factors of N that are less than $N^{1/3}$ in $\tilde{O}(N^{1/6})$ bit operations. Then test whether the remaining unfactored part of N is a perfect power.

Chapter 13: Basic Discrete Logarithm Algorithms

13.0.3: Split N using Miller-Rabin style idea once know order of random g modulo N.

13.2.7: First show, using calculus, that $f(x) = x/\log_2(x)$ is monotonically increasing for $x \geq 2.7183$. Hence deduce that if $B \geq 4$ and $2 \leq x \leq B$ then $x \leq B \log_2(x)/\log_2(B)$. Finally, prove the result using induction on n.

13.3.3: Set $m = \lceil \sqrt{r/2} \rceil$ and loop giant steps up to $\lceil \sqrt{2r} \rceil$.

13.3.4: Let X be the random variable $\max\{x, y\}$ over $(x, y) \in [0, r]^2$. If $0 \leq t \leq r$ then $\Pr(X \leq t)$ is the area of the box $[0, t]^2$ divided by the area of $[0, r]^2$, i.e., t^2/r^2. Hence, $\Pr(X > t) = 1 - t^2/r^2$ and the expected value of X is $\sum_{t=0}^{r} \Pr(X > t) \approx \int_0^r (1 - t^2/r^2) dt = 2r/3$. Hence, choosing both tables to have size $\sqrt{r}$ gives an algorithm with average-case running time $2\frac{2}{3}\sqrt{r}$ group operations and which requires storing the same number of group elements. See Section 3 of Pollard [488].

13.3.6: It is convenient to replace h by hg^{-b} so that $h = g^a$ with $0 \leq a < w$. Then set $m = \lceil \sqrt{w/2} \rceil$ and run Algorithm 14 with trivial modifications (the while loop now runs to $2m$).

13.3.7: Let $m = \lceil \sqrt{w}/2 \rceil$. If $h = g^a$ then write $a = a_0 + 2ma_1$ where $-m \leq a_0 \leq m$ and $0 \leq a_1 \leq 2m$. Then one needs m group operations to produce the list of baby steps g^{a_0} and, on average, $\frac{1}{2}2m = m$ group operations for the giant steps.

13.3.8: We have $a = b + ma_1$ for some integer a_1 in the range $0 \leq a_1 < w/m$. Let $h_1 = hg^{-b}$ and $g_1 = g^m$. Then $h_1 = g_1^{a_1}$. The BSGS algorithm can be used to solve this problem in $O(\sqrt{w/m})$ group operations.

13.3.9: See van Oorshot and Wiener [472].

13.3.10: One can just run BSGS twice, but a better solution is to set $m = \lceil \sqrt{2w} \rceil$, $h_1 = hg^{-b_1}$ and $h_2 = hg^{-b_2}$. One computes m baby steps as usual and then computes $m/2$ giant steps starting from h_1 and another $m/2$ giant steps starting from h_2.

13.3.11: Solve the DLP instance $1 = g^a$.

13.3.13: Blackburn and Teske [60].

13.3.14: By Exercise 11.1.15 the number of NAFS is $\approx 2^{n+2}/3$. If a is a NAF then $a = a_0 + 2^{\lceil n/2 \rceil} a_1$ where a_0 and a_1 are NAFs of length $l = \lceil n/2 \rceil$. There is an efficient procedure to list all NAFs a_0 of length l (see Exercise 11.1.16) so one can compute and store all g^{a_0} in a sorted list. This gives a BSGS algorithm requiring $O(2^{\lceil n/2 \rceil + 2}/3)$ time and space.

13.4.6: See Maurer [404].

13.5.4: Let $m = \lfloor l/2 \rfloor$. Use the equation

$$g_1^{a_1} \cdots g_m^{a_m} = h g_{m+1}^{-a_{m+1}} \cdots g_l^{-a_l}.$$

Let $N_1 = \#\mathcal{S}_1 \cdots \#\mathcal{S}_m$ and $N_2 = \#\mathcal{S}_{m+1} \cdots \#\mathcal{S}_l$. Compute and store the left hand side of the equation in time and space $O(N_1)$ then compute the right hand side and check for

a match. The running time is $O(N_1 + N_2)$ group operations and the storage is $O(N_1)$ group elements.

13.5.5: Define $\mathcal{S}_1' = \{a_1 \cdots a_{\lfloor l/2 \rfloor} : a_j \in \mathcal{S}_j \text{ for } 1 \leq j \leq l/2\}$ and $\mathcal{S}_2' = \{a_{\lfloor l/2 \rfloor + 1} \cdots a_l : a_j \in \mathcal{S}_j\}$. We assume $\#\mathcal{S}_1' \leq \#\mathcal{S}_2'$. Compute and store (g^{a_1}, a_1), sorted according to the first component, for all $a_1 \in \mathcal{S}_1'$ then compute each $h^{a_2^{-1}}$ for $a_2 \in \mathcal{S}_2'$ and check for a match with the earlier list. The running time is $O(\#\mathcal{S}_1' + \#\mathcal{S}_2')$ group operations and the storage is $O(\#\mathcal{S}_1')$ group elements.

13.6.10: Find the largest $\alpha \in \mathbb{R}$ such that $\alpha \leq 1/2$ and $M \geq \binom{\lceil \alpha n \rfloor}{\lfloor \alpha w \rfloor}$. Compute M "baby steps" and then $\binom{n - \lceil \alpha n \rfloor}{w - \lfloor \alpha w \rfloor}$ "giant steps".

13.8.2: A solution is to sort L_1 and then, for each $x_2 \in L_2$ to check whether $x_2 \in L_1$. For the "hash-join" solution see Wagner [625].

13.8.5: See Wagner [625].

13.8.7: $\mathcal{D} = \{x \in \{0,1\}^n : \mathrm{LSB}_m(x) = 0\}$.

13.8.9: See Wagner [625].

Chapter 14: Factoring and Discrete Logarithms using Pseudorandom Walks

14.1.3: By solving $e^{-l^2/2N} < p)$ for l, for probability 0.99 the number of trials needed is $\approx 3.035\sqrt{N}$ and for probability 0.999 the number of trials is $\approx 3.717\sqrt{N}$.

14.2.4: $l_t = 20, l_h = 10$, so $x_{20} \equiv x_{40} \pmod{p}$. The walk in this example is mostly squarings, which is "not very random" and which justifies why $l_t + l_h$ is so much larger than $\sqrt{\pi r/2}$.

14.2.5: $l_t = 7, l_h = 6$, so first collision is $x_{12} = x_{24}$.

14.3.3: Solving $e^{-\alpha} = 1/r$ (the probability to simply guess the correct solution to the DLP) gives $\alpha = \log(r)$.

14.2.18: On can store some bits of $H(x_n)$, where H is a hash function. A further variant is for the clients not to compute or store the values a_i and b_i. Instead, the algorithm is structured so that the values (a_0, b_0) of the initial point $x_0 = g^{a_0} h^{b_0}$ are a function of a short random "seed"; the client then sends the end-point of the walk and the seed to the server. The server then re-calculates the walk, this time including the values a_i and b_i, once a collision has been detected. We refer to Bailey et al. [25] for details.

14.2.21: Each step of the algorithm requires computing $g^{a_0} h^{b_0}$ for random $0 < a_0, b_0 < r$, which (naively) requires $3 \log_2(r)$ group operations. The cost can be reduced to around $\log_2(r)$ (and even further) with modest precomputation and storage. In any case the total cost of the algorithm would be around $1.25\sqrt{r} \log_2(r)$ group operations on average, compared with $1.41\sqrt{r}$ for baby-step-giant-step. The storage requirements are similar.

14.4.9: $x_{i+2} = (x_i g)^{-1} g = x_i^{-1}$. To have the cycle we need $S(\hat{x}_{i+1}) = S(\hat{x}_i)$ which occurs with probability $1/n_S$ and we need $\hat{x}_{i+1} = x_{i+1}^{-1}$ which occurs with probability $1/N_C$.

14.5.6: The worst case is when $h = g^a$ with $a = 0$ or $a = w - 1$. The expected cost is then $w/(2m) + m$. To minimise this take $m = \sqrt{w/2}$. The worst-case complexity is then $(2\sqrt{2} + o(1))\sqrt{w}$ group operations.

14.5.7: Mean jump size is $m \approx 5.38$. Expected length of walk is $l \approx 8.7$. Probability of success is $1 - (1 - 1/m)^l \approx 0.83 \approx 5/6$. See Pollard [488].

14.5.9: The algorithm doesn't really change. Let d be the expected distance of the wild kangaroo from the center of the interval (for the uniform distribution on $\{0, 1, \ldots, w-1\}$ we had $d = w/4$. Then the expected running time is $d/m + 4(1+\epsilon)m/N_P^2 + 1/\theta$ group operations. The optimal value for m is $N_P\sqrt{d}/2$, giving an average-case expected running time $4(1+\epsilon)\sqrt{d}/N_P + 1/\theta$ group operations.

14.5.10: If $w \geq \pi r/8 \approx 0.4r$ then one should use the rho algorithm. If w is significantly smaller than $0.4r$ then one would use the kangaroo method. Determining the exact crossover would require careful experimentation.

14.5.12: See Galbraith, Pollard and Ruprai [226].

14.6.3: The heuristic cost is $w/(2m) + 4m/N_P^2$, which is minimised by $m = N_P\sqrt{w/8}$.

14.7.6: We always have $\#(T \cap W) = r$ where $r = \#G$. The heuristic complexity is therefore $(1+o(1))\sqrt{\pi r} \approx (1.77+o(1))\sqrt{r}$ group operations, which is slower than Pollard rho.

14.7.7: See Galbraith and Ruprai [227].

14.8.1 See van Oorchot and Wiener [473].

14.9.5: x^2+1 is fast to compute and does not have any trivial fixed points or short cycles. The other suggestions have fixed points independent of $p \mid N$.

14.9.6: The idea is that $f(x)$ is now m-to-1, so the expected length of the rho is shorter (this is the same as the reason why taking $n_S > 3$ is a good idea; see the end of Section 14.2.5). For more discussion of this specific case see Brent and Pollard [99].

14.9.7: One cannot "see" a match modulo p without computing a gcd.

Chapter 15: Factoring and Discrete Logarithms in Subexponential Time

15.1.4: We have the error term like $\exp(-u(\log(u) + c\log(\log(u))))$ for a suitable function $c \geq 1$. This splits as $u^{-u}\log(u)^{-cu}$. If the second term is $u^{-f(u)}$ for some function then taking logs gives $f(u)\log(u) = cu\log(\log(u))$ and so $f(u) = cu\log(\log(u))/\log(u) = o(u)$.

15.1.7: The first 4 properties are straightforward to verify. Property 5 follows since $\log(\log(N)^m) = m\log(\log(N)) < c\log(N)^a\log(\log(N))^{1-a} = \log(L_N(a,c))$ for sufficiently large N. Property 6 follows from property 5 and property 2. To prove property 7 let $f(N) = m(\log(\log(N))/\log(N))^a$ and note that $L_n(a, f(N)) = \exp(m\log(\log(N))) = \log(N)^m$. It is easy to check that $\lim_{N\to\infty} f(N) = 0$ and so $f(N) = o(1)$. Properties 8 and 9 are easily checked.

15.1.9: Let $B = L_N(1/2, c)$ and

$$u = \log(N)/\log(B) = \log(N)/(c\sqrt{\log(N)\log(\log(N))}) = \tfrac{1}{c}\sqrt{\log(N)/\log(\log(N))}.$$

The probability of being B-smooth is therefore $\Psi(N,B)/N = u^{-u(1+o(1))}$ as $N \to \infty$. Now $\log(u) = \log(1/c) + \frac{1}{2}(\log(N)\log(N) - \log(\log(\log(N)))) = (1/2+o(1))\log(\log(N))$. The result follows.

15.2.1: For each prime $p \mid N$ (necessarily odd) the equation $x^2 \equiv 1 \pmod{p}$ has exactly two distinct solutions modulo p. If $p^e \mid N$ for $e > 1$ then by Hensel lifting the equation $x^2 \equiv 1 \pmod{p^e}$ also has exactly 2 solutions. The result follows by the Chinese remainder theorem.

15.2.3: Some random relations are

$$\begin{aligned} 1717^2 &\equiv 2^2 \cdot 3^3 \cdot 5 \cdot 7 \pmod{N} \\ 2365^2 &\equiv 2^3 \cdot 3^2 \cdot 5 \cdot 7 \pmod{N} \\ 1757^2 &\equiv 2^7 \cdot 3^3 \pmod{N} \\ 519^2 &\equiv 2^5 \cdot 3 \cdot 5^2 \pmod{N} \end{aligned}$$

and so the relation matrix is

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 3 & 2 & 1 & 1 \\ 7 & 3 & 0 & 0 \\ 5 & 1 & 2 & 0 \end{pmatrix}.$$

Adding the first three rows modulo 2 gives the all zero vector (again, we are lucky that 5 relations are not required) and let $X = 1717 \cdot 2365 \cdot 1757 \pmod N$ and $Y = 2^6 \cdot 3^4 \cdot 5 \cdot 7 \pmod N$. Then $\gcd(X - Y, N) = 73$. One has $N = 53 \cdot 73$.

15.2.5: See Exercise 16.5 of Shoup [556].

15.2.8: First note that $(\#\mathcal{B})^2 = L_N(1/2, 1 + o(1))$ as required. For the second part, write $u = \log(N^{1/2+o(1)})/\log(B)$ and note that one expects $T_B \approx u^u$. Now,

$$u = (\tfrac{1}{2} + o(1)) \log(u)/(\tfrac{1}{2}\sqrt{\log(N)\log(\log(N))}) = (1 + o(1))\sqrt{\ln(N)/\ln(\ln(N))}.$$

Hence,

$$\log(u^u) = u \log(u) = (\tfrac{1}{2} + o(1))\sqrt{\log(N)\log\log(N)}.$$

Hence $\#\mathcal{B}T_B = O(L_N(1/2, 1 + o(1))) = L_N(1/2, 1 + o(1))$. See Section 6.1 of [162] or Section 16.4.2 of [556] for further details.

15.2.10: If $x = \sqrt{kN} + \epsilon$ then either $x^2 - kN$ or $kN - x^2$ is a positive integer $\leq 2\sqrt{kN}\epsilon$. The algorithm then proceeds the same way.

15.3.4: One expects to make $L_N(1/3, 1/(2c)+o(1))$ trials until the value x is $L_N(2/3, c)$-smooth. One can use ECM to factor the numbers (if they are smooth) in $L_p(1/2, \sqrt{2}+o(1))$ operations. Inserting $p = L_N(2/3, c)$ gives the result.

15.5.1: Let $u \in \langle g \rangle \cap G' = \{1\}$. The order of u divides r and $(p-1)/r$. Since $\gcd(r, (p-1)/r) = 1$ it follows that the order of u is 1.

15.5.2: If $r < (p-1)/r$ is large then one can efficiently generate a random element $\delta \in G'$ by choosing a random element $w \in \mathbb{F}_p^*$ and computing $\delta = w^r \pmod p$. If $r \geq (p-1)/r$ then it is better to precompute an element $g_1 \in \mathbb{F}_p^*$ of order $(p-1)/r$ and then compute $\delta = g_1^i \pmod p$ for a random value of i (one can speed this up further using a Pollard-style pseudo-random walk, which is also a good solution when r is small).

15.5.3: See Algorithm 16.1 and Lemma 16.2 of Shoup [556].

15.5.7: Let $B = L_p(1/2, c)$. By Corollary 15.1.8, $T_B = L_p(1/2, 1/(2c) + o(1))$. Hence the running time of the relation collection is $O(L_p(1/2, 2c + 1/(2c) + o(1)))$ bit operations and the running time of the linear algebra is $O(L_p(1/2, 2c + o(1)))$ bit operations. The optimal value of c is $1/2$ and the complexity is as stated.

15.5.8: Let $r^e \| (p-1)$ and let $\gamma \in F_p^*$ have order r^e. Let G' be the subgroup of order $(p-1)/r^e$. Generate relations of the form $\gamma^z \delta \pmod p$ with $1 < z < r^e$ and $\delta \in G'$. One needs a relation featuring g and a relation featuring h. The linear algebra is now modulo r^e (see Exercise 15.4 of [556]). One finds $\gamma^{Z_1} g^{Z_2} = 1$ from which it follows that $r^{e-1} \mid A$ and so $\gamma_1^{Z_1'} g^{Z_2} = 1$ with $\gamma_1 = \gamma^{r^{e-1}}$ and $Z_1' = Z_1/r^{e-1}$. One therefore compute the logarithm of g to the base γ_1 and similarly for h.

15.5.9: Once the first s relations are found only one relation is needed for each h_i, and this takes $O(u^u \#\mathcal{B} M(\log(p))) = O(L_p(1/2, c + 1/(2c) + o(1))) = O(L_p(1/2, 3/2 + o(1))$ bit operations (using trial division for the relations). With our current formulation the full linear algebra has to be repeated each time (actually, this can be avoided by using a different approach), but this only costs $O(L_p(1/2, 1 + o(1)))$ bit operations.

15.5.10: We are testing numbers of size $\sqrt{p}$ for B-smoothness so the number of trials before getting a B-smooth value for w_1 is $(u')^{u'}$ where $u' = \log(\sqrt{p})/\log(B) = \frac{1}{2}\log(p)/\log(B)$. However, since we need both w_1 and w_2 to be smooth the number of trials is $(u')^{2u'} = (u/2)^u$. For $B = L_p(1/2, c)$ it follows that $\log((u/2)^u) = u \log(u/2) \approx u \log(u)$ and so one gets the same complexity as before (but a smaller $o(1)$ term).

15.5.14: Suppose b is even. One can write the sum as $p^b/b + p^{b-1}/(b-1) + \cdots + p^{b/2}/(b/2)$ plus fewer than $2b$ terms which are all $O(p^{b/2})$. Since $p^{b-i}/(b-i) \leq p^{b-i}/(b/2) = 2p^{b-i}/b$ for $0 \leq i \leq b/2$ the first result follows.

For the approximation just note that $p^b/b + p^{b-1}/(b-1) + \cdots \approx p^b/b(1 + 1/p + 1/p^2 + \cdots) = p^b/b(1 - 1/p)$.

15.5.16: Following the discussion earlier in this section, let $b = c\sqrt{n\log(n)/\log(p)}$ and $u = n/b = \frac{1}{c}\sqrt{n\log(p)/\log(n)}$. Then $\#\mathcal{B} < p^b$ and so $\#\mathcal{B} = O(L_{p^n}(1/2, c))$. Now

$$\begin{aligned}\log(u^u) = u\log(u) &= \tfrac{1}{c}\sqrt{n\log(p)/\log(n)}\left(\log(1/c) + \tfrac{1}{2}(\log(n) + \log(\log(p)) - \log(\log(n))\right)\\ &= \left(\tfrac{1}{2c} + o(1)\right)\sqrt{n\log(n)\log(p)}.\end{aligned}$$

Hence $u^u = O(L_{p^n}(1/2, 1/(2c) + o(1)))$ as usual. The total running time is therefore $O(L_{p^n}(1/2, c + 1/(2c) + o(1)) + L_{p^n}(1/2, 2c + o(1)))$ bit operations. This is minimised when $2c^2 = 1$, i.e., $c = 1/\sqrt{2}$, which gives the stated complexity. We refer to Section 4 of Odlyzko [469] for more details, but note that Odlyzko writes the complexity as $O(\exp(c\sqrt{n\log(n)}) = O(L_{p^{n/\log(n)}}(1/2, c))$.

15.5.18: Note that $\mathbb{F}_{2^d} = \mathbb{F}_2[x]/(A(x))$ where $d = \deg(A(x))$. Let $\alpha \in \mathbb{F}_{2^d}$ be a root of $A(x)$. If the equation $x^2 + x = \alpha$ has no solutions then $A(x^2 + x)$ is irreducible. If the equation $x^2 + x = \alpha$ has two solutions (namely, α and $\alpha + 1$) in $\mathbb{F}_{2^d}$ then $A(x^2 + x)$ is the product of their minimal polynomials.

15.5.22: For example $F_1(t) = t^4 + t^2 + t + 1$ and $F_2(t) = t^4 + t^2$.

15.5.23: Note that $\phi_1(\psi_1(x)) = t$, $\phi_1(\psi_1(y)) = \phi_1(F_1(x)) = F_1(t)$, $\phi_2(\psi_2(x)) = \phi_2(F_1(y)) = F_2(F_1(t)) \equiv t \pmod{F(t)}$ and $\phi_2(\psi_2(y)) = \phi_2(y) = F_1(t)$.

15.6.2: Set $b = \lceil \log_q(L_{q^g}(1/2, c))\rceil$. Construct the factor base $\mathcal{B}$ as above, generate random group elements using the algorithm of Section 15.5.1 and use Theorem 15.6.1 to determine the expected number of trials to get a smooth relation. The group operations and polynomial factorisation are all polynomial-time and can be ignored. The algorithm therefore has the usual complexity $\#\mathcal{B}L_{q^g}(1/2, 1/(2c) + o(1)) + (\#\mathcal{B})^{2+o(1)}$ which is $L_{q^g}(1/2, c + 1/(2c) + o(1)) + L_{q^g}(1/2, 2c + o(1))$ when g is sufficiently large. This is optimised by taking $c = 1/\sqrt{2}$, giving the stated running time. For the technical details see Enge and Gaudry [195] or Section VII.6.1 of [65].

15.6.3: See page 36 of Adleman, DeMarrais and Huang [4] for the case of one point at infinity.

15.6.4: Clearly degree 1 prime divisors are points and $\#C(\mathbb{F}_q) = q + 1 + t$ where $|t| \leq 2g\sqrt{q} = o(q)$. As we know, there are approximately q^g divisor classes and $\binom{\#C(\mathbb{F}_q)+g-1}{g} = \binom{q(1+o(1))+g-1}{g} = q^g(1 + o(1))/g!$ divisors formed by taking sums of prime divisors of degree 1 (and subtracting a suitable divisor at infinity to get a degree 0 divisor).

For hyperelliptic curves with a single point at infinity, Lemma 10.3.24 shows the Mumford representation is unique and the above argument therefore proves the result. However, for general hyperelliptic curves uniqueness does not necessarily hold and so this argument needs some care. To deal with this one uses the fact that there are $\#\mathrm{Pic}^0_{\mathbb{F}_q}(C) = q^g(1 + o(1))$ divisor classes and hence only $o(q^g)$ divisor classes contain more than one reduced divisor. It follows that only $o(q^g)$ sums of g points in $C(\mathbb{F}_q)$ can yield divisor classes containing more that one reduced divisor, and so the above arguments do prove what is required.

15.6.5: See [242] or Section VII.6.2 of [65].

15.8.3: One can easily check all three properties for $n = 2$. The case $n = 3$ follows directly from the group law: if $x_1 \neq x_2$, then $x_1 + x_2 + x_3 = \lambda^2$ where $\lambda = (y_2 - y_1)/(x_2 - x_1)$. Expanding gives $2y_1y_2 = y_1^2 + y_2^2 + (x_2 - x_1)(x_1 + x_2 + x_3)$; squaring both sides and substituting $x_i^3 + a_4x_i + a_6$ for y_i^2 gives $(x_1 - x_2)^2$ times the stated equation. The case $x_1 = x_2$ follows by using $\lambda = (3x_1^2 + a_4)/(2y_1)$.

The general case follows since $P_1 + \cdots + P_n = \mathcal{O}_E$ if and only if $P_1 + \cdots + P_{n-2} = -R$ and $P_{n-1} + P_n = R$ for some point R on the curve. In other words, $P_1 + \cdots + P_{n-2} + R =$

$P_{n-1} + P_n + (-R)) = \mathcal{O}_E$. The symmetry is obvious, and the statement about degrees follows by induction and using the fact that the resultant of a quadratic and a degree 2^{n-2} polynomial is a degree 2^{n-1} polynomial.

Chapter 16: Lattices

16.1.4: A hint for last part is that the lattice contains the sublattice with basis $M\underline{e}_i$

16.1.13: $\underline{b}_1 = (1, 1)$ has volume $\sqrt{2}$.

16.2.5: (You need to have studied the proof of Theorem 16.2.3.) A radius $r = \lambda_1 + \epsilon$ hypercube around the origin has volume $(2r)^n$. Taking $\epsilon > 0$ arbitrarily small gives the result. For the second claim, consider the convex region $\{\underline{v} \in \mathbb{R}^n : \|\underline{v}\|_1 \leq r\}$, which is a hyper-tetrahedron of volume $2^n r^n/n!$. The approximation is using Stirling's formula.

16.2.6: Consider the lattice basis $\{(1, a), (0, b)\}$ of rank $n = 2$ and determinant b. Every element of the lattice is of the form $(s, as + bt)$ for some $s, t \in \mathbb{Z}$. By Minkowski, the disk of radius $u = \sqrt{b\sqrt{2}}$ has volume $\pi u^2 > 4b$ and so contains a non-zero lattice point. Hence, $0 < s^2 + r^2 < u^2 = \sqrt{2}b$.

16.3.1: Problem 1 is achieved by solving $\underline{x}B = \underline{v}$ over $\mathbb{Q}$ (or with sufficiently accurate floating point arithmetic, rounding to the nearest integer solution and then checking).

To solve problem 2, compute the HNF of the matrix B whose rows are $\underline{b}_1, \ldots, \underline{b}_n$ and discard the zero rows. For problem 3, write $A' = UA$ be the HNF of A. If the first r rows of A' are zero then the first r rows of U are a basis for $\ker(A)$ (since if $\underline{x}$ is any vector with last $m - r$ entries zero then $0 = \underline{x}A' = (\underline{x}U)A$).

To solve problem 4, concatenate the n rows of the $n \times n$ matrix MI_n to the matrix A to apply an extended matrix A'. Then use the method used to solve problem 3 on the matrix A'. To see this is correct note that $(x_1, \ldots, x_m)A \equiv \underline{0} \pmod{M}$ if and only if there are n integers $x_{n+1}, \ldots, x_{n+m}$ such that then $(x_1, \ldots, x_{n+m})A' = \underline{0}$.

Chapter 17: Lattice Basis Reduction

17.1.6: Clearly each change of basis is invertible and so the output is a basis of the lattice. Since $B_1 \in \mathbb{N}$ is strictly decreasing the algorithm must terminate after a finite number of steps.

17.1.8: $\{(-1, 0), (0, 2)\}$.

17.2.5: Yes, no, yes, no, yes.

17.2.7: An example is $\underline{b}_1 = (1, 0)$ and $\underline{b}_2 = (0.49, 0.8)$.

17.2.10: The proof closely follows the proof of Lemma 17.2.8. For part 2 one should find $\|\underline{b}_i\|^2 \leq (1 + \frac{1}{4}\sum_{k=1}^{i-1} \sqrt{2}^k)B_i \approx (0.146 + \sqrt{2}^i/1.46)B_i$ and one gets the result. Since $1/6 \leq (\sqrt{2} - 1)2^{(i-1)/2}$ for $i \geq 1$ one has $\|\underline{b}_j\|^2 \leq 2^{j/2}B_j$ and part 3 follows.

17.2.16: An example of the situation $\|\underline{v}_1\| \neq \|\underline{b}_1\|$ was seen in Exercise 17.2.5. For the second part, we have $2^{n(n-1)/4} \det(L) \geq \prod_{j=1}^{n} \|\underline{v}_j\| \geq \|\underline{v}_i\|^{n+1-i}$.

17.4.4: See [373] or Section 2.6.1 of [136].

17.5.2: Let L be any lattice of dimension n. We have proved than an LLL-reduced basis for L exists. Hence, by part 5 of Theorem 17.2.12 $\lambda_1 \leq 2^{(n-1)/4} \det(L)^{1/n}$. Also see Section 12.2 of Cassels [121].

Chapter 18: Algorithms for the Closest and Shortest Vector Problem:

18.1.9: The complexity of computing the Gram-Schmidt basis is given by Theorem 17.3.4. Using the same techniques, one can show that $\underline{w}_i$ can be represented using exact $\mathbb{Q}$-arithmetic with denominators bounded by X^{n-1} and numerator bounded by X^n.

18.1.10: The inductive process uses the fact that the orthogonal complement of $\underline{b}_n$ is the span of $\{\underline{b}_1^*, \ldots, \underline{b}_{n-1}^*\}$. If one starts at $\underline{b}_1$ instead of $\underline{b}_n$ then one needs an analogue of Lemma 18.1.1.

18.2.5: If $\underline{w} = \sum_{i=1}^n l_i \underline{b}_i$ and $\underline{u} = \sum_{i=1}^n l_i' \underline{b}_i$ then $\|\underline{w} - \underline{u}\|^2 = \sum_{i=1}^n (l_i - l_i')^2 \|\underline{b}_i\|^2$. The result follows.

18.2.7: $\underline{w} \approx 24.09\underline{b}_1 + 12.06\underline{b}_2 + 26.89\underline{b}_3$ so $\underline{v} = (99, 204, 306)$ and $\|\underline{v} - \underline{w}\| = \|(1, 1, -1)\| = \sqrt{3}$.

18.3.4: $(100, 77, 104)$.

18.4.5: See Figure 1 of Hanrot and Stehlé [275] or Algorithm 10.6 of Joux [317].

18.4.7: See Section 3 (under "Combinatorial methods") of Micciancio and Regev [423].

Chapter 19: Coppersmith's Method and Related Applications

19.1.8: The dimension is $2d$, the determinant is $M^d X^{2d(2d-1)/2}$, the condition (ignoring the constants) $(M^d X^{2d(2d-1)/2})^{1/2d} \leq M$ leads to $X \approx M^{1/(2d-1)}$. See Section 2 of Coppersmith [142].

19.1.14: Set $x_0 = px$ for $x \in \mathbb{Z}$. There is a similar example in Section 3 of Coppersmith [142].

19.4.5: See Section 3.2 of May [410] (pages 34-36).

19.4.6: Set $\epsilon = 1/\log_2(N)$ and apply Theorem 19.4.2 a constant number of times, each with a different guess for the most-significant 2 bits of $p - \tilde{p}$.

19.4.7: Guess all $N^{1/4}$ values for $\tilde{p}$ and try Coppersmith's algorithm (in polynomial-time) on each of them.

19.4.8: Just use $F(x) = (\tilde{p} + x)$ as before and note that the proof of Theorem 19.4.2 does not change. This is mentioned on page 52 of Howgrave-Graham [297].

19.4.9: Assume that $0 \leq \tilde{p} < M$. It follows that $Mx + \tilde{p}$ has a small root modulo p. Hence apply the same methods using the polynomial $F(x) = Mx + \tilde{p}$, which can be made monic as $F(x) = x + (\tilde{p}M^{-1}) \pmod{N}$.

19.4.10: Let $p = \tilde{p} + x_0$ with $0 \leq x_0 < X$ then setting

$$\tilde{q} = \lfloor N/(\tilde{p} + X) \rfloor$$

means $\tilde{q} \leq q \leq N/\tilde{p}$, which is an interval of width $NX/(\tilde{p}(\tilde{p} + X)) < qX/\tilde{p}$. Hence $q = \tilde{q} + y_0$ with $0 \leq y_0 \leq q(X/\tilde{p})$.

19.4.12: Let $\tilde{p} = 5000$ and $X = 10$. We seek a small root of $F(x) = (\tilde{p} + x)^3$ modulo a large factor of N. Reducing the matrix

$$\begin{pmatrix} N & 0 & 0 & 0 & 0 \\ 0 & NX & 0 & 0 & 0 \\ 0 & 0 & NX^2 & 0 & 0 \\ \tilde{p}^3 & 3\tilde{p}^2 X & 3\tilde{p}X^2 & X^3 & 0 \\ 0 & \tilde{p}^3 X & 3\tilde{p}^2 X^2 & 3\tilde{p}X^3 & X^4 \end{pmatrix}$$

yields the row $(1980391527, -16046759730, 20944500000, 35123963000, 35110000)$. This corresponds to the polynomial $3511x^4 + 35123963x^3 + 209445000x^2 - 1604675973x + 1980391527$, which has the root $x = 3$, giving $p = 5003$.

19.4.14: The algorithm requires $O(\binom{n}{e} \log(P)^3)$ bit operations, and at least $\binom{n}{e} > (n/e)^e$ bit operations. The input size is $O(n \log(p_n))$ bits (assuming all $0 \leq r_i < p_i$). The algorithm is not polynomial-time: a polynomial-time algorithm would need $\leq (n \log(p_n))^c$ bit operations for some constant c, and so the logarithm of the running time would be

$\leq c(\log(n) + \log\log(p_n))$; whereas if $e > \log(n)$ then the logarithm of the running time of the algorithm is $\geq \log((n/e)^e) > \log(n)^2 - \log(n)\log\log(n)$.

19.4.16: $p_n \approx n\log(n)$ so $P > n! > (n/e)^n$ (warning: the e here is 2.71828... from Stirling's formula) and so $\log(P) > n(\log(n) - 1)$. Then $\sqrt{\log(X)/\log(P)}\log(p_n) \approx \sqrt{\log(X)/(n\log(n))}\log(n) = \sqrt{\log(X)\log(n)/n}$. The claimed value for e follows. This is obviously $> \log(n)$ for large n and sufficiently small X.

19.4.17: Let $m = \lceil \log_2(U) \rceil$. Then $2^m \leq U < 2^{m+1}$ and $2^{m+1} \leq 2U \leq V$.

19.4.19: For the first case we get $x = -347641$ and for the second $x = 512$.

19.5.2: Given q let $p_i = \lfloor q\alpha_i \rceil$ so that $|q\alpha_i - p_i| \leq 1/2$.

19.5.4: The first row output by LLL on the matrix is approximately $(-0.0000225, 0.0002499, 0.0002499, 0.000750$
Now compute $q = 0.0000225Q/\epsilon = 2250$. One finds $p_1 = 3499, p_2 = 1735$, $p_3 = 750$.

19.6.1: Note that $\frac{1}{2}b^{1/3} < q_b < b^{1/3}$ and so $|y| < \frac{1}{2}q_b$ as required. Also, note that the continued fraction method finds q_a/q_b as long as $|(\tilde{a}/\tilde{b}) - (q_a/q_b)| < 1/(2q_b^2)$. First note that $b^{-1/3} < 1/q_b < 2b^{-1/3}$ so $\frac{1}{2}b^{-2/3} < 1/(2q_b^2)$. Now, the difference $(\tilde{a}/\tilde{b}) - (q_a/q_b)$ is given by equation (19.6), whose numerator is bounded by $2\frac{1}{2}b^{1/3}\frac{1}{4}b^{1/3}$. Since $1/q_b < 2b^{-1/3}$ we have the difference bounded by $\frac{1}{2}b^{1/3}/\tilde{b}$. It follows that the difference is less than $1/(2q_b^2)$ as required and so Euclid solves the problem.

19.6.2: If $\alpha < \beta$ then the target gcd is smaller than the errors – one therefore expects very many solutions.

The first condition comes from the Diophantine approximation step: Namely that the right hand side of equation (19.6) (which is $1/\tilde{b}^{1-\beta}$) is at most $1/(2q_b^2) \approx 1/\tilde{b}^{2(1-\alpha)}$. The second condition comes from the requirement that $|y| < \frac{1}{2}q_b$.

19.6.3: The process is the same as the generalisation of Diophantine approximation in Section 19.5. The lattice is $\begin{pmatrix} \tilde{b}^\beta & \tilde{b} \\ 0 & \tilde{a} \end{pmatrix}$ which has determinant roughly $\tilde{b}^{1+\beta}$ and a short vector $(q_a\tilde{b}^\beta, q_b x - q_a y)$ of length roughly $\tilde{b}^{1+\beta-\alpha}$. The result follows.

19.7.4: See [496] for the first reduction. The second is almost immediate, but needs some care.

Chapter 19a: Cryptosystems Based on Lattices

19.9.1: Given a ciphertext $\underline{c}$ one can check if it is an encryption of $\underline{m}$ by testing whether $\underline{c} - \underline{m}G'$ (respectively, $\underline{c} - \underline{m}B'$) is a valid error vector for the system.

19.9.2: Check if $\underline{c}$ is an encryption of $\underline{m}$ by testing if $\underline{c} - \underline{m}$ lies in the code (respectively, lattice) corresponding to the public key.

19.9.3: Given $\underline{c}$ add $\underline{m}'G'$ (respectively, $\underline{m}'B'$) to get a ciphertext $\underline{c}' \neq \underline{c}$ which is an encryption of $\underline{m} + \underline{m}'$. Alternatively, add an extremely small error vector $\underline{e}'$ to get $\underline{c}'$ and call the decryption oracle; hopefully this will return the message $\underline{m}$.

19.9.8: $(2, 3)$.

19.9.9: Set $\underline{w} = \underline{v}$ and, for $i = n$ downto 1 do $\underline{w} = \underline{w} - \lfloor \langle \underline{w}, \underline{b}_i^* \rangle / \langle \underline{b}_i^*, \underline{b}_i^* \rangle \rfloor$.

19.10.3: $\underline{m} = (1, 2, 0), \underline{e} = (1, 1, 1)$; $\underline{m} = (-1, 0, 1), \underline{e} = (1, -1, 1)$.

19.13.2: There is a solution $s = \sum_{i=1}^n x_i b_i$ if and only if $s' = s - x_n b_n$ is a subset sum of $\{b_1, \ldots, b_{n-1}\}$.

19.13.3: Assume n even. Compute all $2^{n/2}$ integers $\sum_{i=1}^{n/2} x_i b_i$ for $x_i \in \{0, 1\}$ and store in a sorted list, binary tree or hash table. Then compute all $2^{n/2}$ values $s - \sum_{i=n/2+1}^n x_i b_i$ and determine whether or not it appears in the list. For details see Algorithm 3.94 of [418].

19.13.4: Just divide everything by the gcd.

19.13.7: Given $s = 112$ we just subtract the largest possible element, in this case 80, which leaves $112 - 80 = 32$. We then subtract the largest element less than 32 to get

$32-20=12$. We then take $12-7=5$. Hence, we have computed that $112=5+7+20+80$, which is the solution vector $(0,1,1,1,0,1,0)$.

19.13.10: $n/(n-1)=1+1/(n-1)$.

19.13.11: $8/\log_2(430)\approx 0.9145$.

19.13.12: By Exercise 19.13.8 $b_n \geq 2^{n-1}$ and so the density is at most $n/(n-1) = 1+1/(n-1)$.

19.13.15: 0110010.

19.13.16: 0.782

19.13.17: $(154, 184, 43, 69, 125, 62)$, $c=384$.

19.13.18: Encryption is deterministic.

19.13.19: Given a ciphertext c one can call $c+a_1$ and $c-a_1$ to the decryption oracle. One of them is a valid ciphertext and corresponds to the original message with the first bit flipped.

19.13.21: Since $0<a_i<M$ we expect an average ciphertext to be a sum of around $n/2$ integers of size $M/2$. Hence $c\approx nM/4$ on average (and, in all cases, $0\leq c<nM$). Since $M\geq 2^n$ we expect c to require at least $\log_2(nM/4)>\log_2(n)+n-2$ bits.

19.13.22: Take $b_i=2^{i-1}$ for $1\leq i\leq n$, $M=2^n+1$ and any W such that $Wb_i \not\equiv 2^n \pmod M$ for all i. Then the density is >1. Of course, it is easy to compute the private key corresponding to such a public key.

19.13.24: Since the integers $a_{i,j}$ are like random integers modulo M_i we expect $\sum_{j=1}^{n} a_{i,j}\approx nM_i/2$ and so $M_{i+1}>nM_i/2$. Hence, $M_t>(n/2)^tM_1\geq(n/2)^t2^n$. The ciphertext is expected to be a sum of around $n/2$ integers of size $M_t/2$ and so around $nM_t/4$. Therefore, on average,

$$\log_2(c)=\log_2(n(n/2)^tM_1/4)>\log_2(n)+t(\log_2(n)-1)+n-2.$$

Since the $a_{t,i}$ are somewhat like randomly chosen integers modulo M_t we expect to have $\max\{a_{t,i}\}\approx M_t$. We conservatively assume in what follows that, on average, $\max\{a_{t,i}\}>M_t/2$. Hence the density is at most $n/\log_2(M_t/2)<n/(t(\log_2(n)-1)+n-1)$. For example, $n=200$ and $t=5$ gives expected density less than 0.87; in any case, the density of an iterated Merkle-Hellman knapsack is always significantly less than 1.

19.13.25: From Exercise 19.13.8 we have $b_n>2^{n-2}b_1$ and so $M>(2^{n-2}+\cdots+2+1+1)b_1=2^{n-1}b_1$. So $b_1b_2>M>2^{n-1}b_1$ implies $b_2>2^{n-1}$. Exercise 19.13.8 also shows $M>2^{n-3}b_2$ and so $M>2^{2n-4}$. Then $\log_2(nM/4)>\log_2(n)+(2n-4)-2$.

19.13.26: See if $W^{-1}a_i \pmod M$ are small and allow efficient solution to the subset sum problem using a greedy algorithm.

19.13.28: $a_1b_2-a_2b_1=7\cdot 233\cdot 37589\cdot 143197$. The only factor of size $\approx a_3$ is $M=233\cdot 37589=8758237$. This gives $W=a_1b_1^{-1} \pmod M = 5236910$. One verifies that $W^{-1}a_i \pmod M$ is a superincreasing sequence.

19.13.30: Suppose W is known and assume no permutation is used. Note that there are integers k_i for $1\leq i\leq n$ such that

$$a_i=b_iW+k_iM$$

and $k_i<b_i$. So $a_1\equiv k_1M \pmod W$ and $a_2\equiv k_2M \pmod W$. Hence, writing $c=a_2a_1^{-1} \pmod W$, we have $k_1c\equiv k_2 \pmod W$. If $k_1k_2<M$ (which is plausible since $k_1k_2<b_1b_2<M$ and W is usually about the same size as M) one can apply the same methods as used in Example 19.13.29 to find (k_1,k_2) and hence M.

19.13.32: 11100001.

19.13.33: 10111100.

Chapter 20: The Diffie-Hellman Problem and Cryptographic Applications

20.2.7: Suppose $l \mid n$ is prime and write $g_1 = g^{n/l}$. Then $g^c = g^{ab}$ implies $g^{cn/l} = g^{abn/l}$ and so $(g_1, g_1^a, g_1^b, g_1^c) = (g^{n/l}, (g^a)^{n/l}, (g^b)^{n/l}, (g^c)^{n/l})$ is a valid Diffie-Hellman tuple. If $l = O(\log(n))$ then one can solve the DLP in $\langle g_1 \rangle$ (this is just Pohlig-Hellman) and hence test DDH in $\langle g_1 \rangle$. If (g, g^a, g^b, g^c) is a random tuple in G^4 then with probability $1/l$ the resulting tuple in $\langle g_1 \rangle$ is not a valid Diffie-Hellman tuple. The algorithm therefore has a noticeable advantage in Definition 20.2.4.

20.4.1: In the first case, $\mathsf{c}_2 = \mathsf{m}h^k$ and $\mathsf{c}_2' = \mathsf{m}h^{Ak+B}$. Hence, $\mathsf{c}_2^A h^B / \mathsf{c}_2' = \mathsf{m}^{A-1}$ and so one can compute m assuming that $(A-1)$ is coprime to the order of the group G. In the second case, query the decryption oracle on $(\mathsf{c}_1, \mathsf{c}_2)$ to get m and hence h^k. One can then compute $(h^k)^A h^B$ and decrypt c_2.

20.4.3: As above we can self-correct the CDH oracle to make it reliable. Once one knows g^{ax} then one can decrypt to get M. For the second part: The problem is that given the message M corresponding to the public key (g, h) and ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ one can compute $M \oplus \mathsf{c}_2 = H(g^{ax})$ but if H is hard to invert then one cannot compute g^{ax}.

20.4.6: Given $(\mathsf{c}_1, \mathsf{c}_2)$ the user returns $\mathsf{m} = \mathsf{c}_2 \mathsf{c}_1^{-a}$ so set $\mathsf{c}_1 = h^{-1}$ and c_2 arbitrary and get $h^a = \mathsf{m}\mathsf{c}_2^{-1}$. If the group contains an element h with relatively small order l then one can easily solve the DLP to get $a \pmod{l}$. If this can be repeated for sufficiently many coprime values l then a can be determined using the Chinese remainder theorem.

20.4.9: See Boneh, Joux and Nguyen [82] for the full details of this attack. The basic idea is as follows: For suitable constant c one has $\mathsf{m} = \mathsf{m}_1 \mathsf{m}_2$ where $1 \leq \mathsf{m}_i \leq c 2^{m/2+\epsilon}$ with a certain noticeable probability (for $c = 1$ [82] states the probability is at least $\log(1 + 2\epsilon)$).

20.5.2: Eve, pretending to be Bob, sends g^y (where y is known to Eve). She makes a corrupt query to Alice and, knowing g^{xy}, can compute g^{ab}. Eve can now compute a shared key with Alice, whereas Alice believes she is sharing with Bob.

Chapter 21: The Diffie-Hellman Problem

21.1.10: For both problems let the space of instances be $(G - \{1\})^2$. Given an Inverse-DH oracle A and instance (g, g^a) one has $g^{a^{-1}} = A(g^x, (g^a)^{xy})^{yx^{-1}}$ for $1 \leq x, y < r$. Given a Square-DH oracle A and instance (g, g^a) one has $g^{a^2} = (A(g^x, (g^a)^{xy})^{(xy^2)^{-1}}$.

For the self-correction: Run the non-perfect Square-DH oracle repeatedly to produce a list L which is expected to contain g^{a^2}. Then choose $0 \leq u < r$ and repeat the process on $(g, g^a g^u)$. This givies a list L' which is expected to contain $g^{(a+u)^2}$. Finally, determine whether there is a unique pair (Z, Z') in $L \times L'$ such that $Z(g^a)^{2u} g^{u^2} = Z'$, and if so return Z. The precise details are similar to Theorem 21.3.8.

21.3.9: Suppose A is correct with noticeable probability ϵ. Since the reduction makes at least $\log_2(r)$ oracle queries, the probability that the result is correct is at most $\epsilon^{\log_2(r)}$ which is negligible. Instead, one should self-correct the oracle A to obtain an oracle A' with success probability greater than $1 - 1/(4\log_2(r))$. By Theorem 21.3.8 this requires $O(\log(\log(r))/\epsilon)$ oracle queries. One can then perform the reduction of Lemma 21.1.13 using A', with success probability $\geq 1/2$.

21.1.19: For the first part, given $(g_1, g_2, g_3) = (g, g^a, g^b)$, call $O_1(g^{r_2}, g_2^{r_2}, g_3^{r_2})$ to get g^{abr_2} and call $O_2(g^{r_1}, g_2^{r_1}, g_3^{r_1})$ to get g^{abr_1}. Finally compute $s, t \in \mathbb{Z}$ such that $r_1 s + r_2 t = 1$ and then $(g^{abr_1})^s (g^{abr_2})^t = g^{ab}$ as required.

For the next part: The problem Inverse-DH is only defined when a is coprime to the group order. If $a = r_1$, as would be required in several places, then we cannot make a meaningful query to an Inverse-DH oracle. Also, in the proof of Lemma 21.1.5 then $g \notin \langle g_1 \rangle$ so a CDH oracle can't work. Indeed, Shoup has shown (Theorem 5 of [553])

that a generic algorithm for Fixed-CDH with respect to g^{r_1}, even when given a perfect Fixed-CDH oracle with respect to g, takes $\Omega(\sqrt{r_2})$ group operations.

21.1.20: For the proof, historical references and a major generalisation of such problems see [102].

21.4.12: For example, with a perfect Fixed-CDH oracle, Pohlig-Hellman only and using projective coordinates the number of oracle queries is $O(\log(r)\log\log(r))$ oracle queries and $O((l_1^2+l_2)\log(r)^2/\log(\max\{l_1,l_2\}))$ group operations.

21.4.13: We give the results only for the case of Pohlig-Hellman combined with exhaustive search. Note that not every $a \in \mathbb{F}_r$ corresponds to an element $a+b\theta$ of $G_{2,r}$, whereas every $a \in \mathbb{A}^1(\mathbb{F}_r)$ corresponds to an element of $T_2(\mathbb{F}_r)$. Hence, embedding the DLP instance into the algebraic group $G_{2,r}$ requires an expected $O(\log(r))$ oracle queries (computing Legendre symbols and taking square roots).

The group operation using the first representation requires no inversions but the group operation for the second representation requires inversions. The number of Fixed-CDH oracle queries respectively is $O(\log(r)\log\log(r))$ and $O(\log(r)^2\log\log(r))$. Hence, the first representation is better. If one has a CDH oracle then inversions are a single oracle query and so the number of oracle queries is $O(\log(r)\log\log(r))$ in both cases.

21.5.5: See Cheon [131].

21.5.9: See Brown and Gallant [111].

21.6.4: See Kaliski [327] or Fischlin and Schnorr [203].

21.6.9: Let A be a perfect oracle which, given $h=g^x$ for $x \in \{0,1,\ldots,r-1\}$ outputs $b(x)$. It suffices to show how to use A to determine the least significant bit of x. We may assume that $h \neq g^{-1}$, since if $h=g^{-1}$ then we know x. If $A(h)=1$ then $(x_1,x_0)=(0,1)$ or $(1,0)$, so $A(gh)$ determines the LSB. Similarly, if $A(h)=0$ then $(x_1,x_0)=(0,0)$ or $(1,1)$ and so $A(gh)$ determines the LSB (since $h \neq g^{-1}$ there is no carry when computing gh).

21.6.11: The integer r in binary begins as $11000\cdots$ so the highest order 2 bits of a random integer modulo r are essentially 00, 01 or 10 with probability close to $1/3$ each. Hence, both predicates are 0 with probability close to $2/3$.

21.6.13: Let A be an oracle such that $A(g^x)=b(x)$. Let $h=g^a$. Set $j=1$ and if $A(h)=0$ then set $l=1$, and if $A(h)=1$ then set $l=2$. Given, at step j, $(l-1)r/2^j \leq a < lr/2^j$ one calls $A(h^{2^j})$. If $A(h^{2^j})=0$ then $(2l-2)r/2^{j+1} \leq a < (2l-1)r/2^{j+1}$, otherwise, $(2l-1)r/2^{j+1} \leq a < 2lr/2^{j+1}$.

We remark that Blum and Micali [73](generalised by Long and Wigderson [393]) use Legendre symbols and square roots modulo p to show this predicate is hardcore in the group $\mathbb{F}_p^*$ when g is a primitive element (their method does not work in a prime order cyclic subgroup).

21.6.15: See Section 7 of Li, Näslund and Shparlinski [387].

21.7.4: This is the same argument as Exercise 21.6.13. One bounds α as $(l-1)p/2^j \leq \alpha < lp/2^j$ and refines (l,j) by computing $A_1(2^j \pmod p) = \mathrm{MSB}_1(\alpha 2^j \pmod p)$.

21.7.6: Use the same argument as Exercise 21.6.4. See Fischlin and Schnorr [203] for details.

21.7.13: Given a DDH instance (g,g^a,g^b,g^c) one can compute $\mathrm{MSB}_{1+\epsilon}(g^c)$ and compare with the result of the oracle. If $g^c=g^{ab}$ then the results will agree. Repeating for random self-reduced versions of the original DDH instance gives the result. We refer to Blake and Garefelakis [62]and Blake, Garefelakis and Shparlinski [63]for details and generalisation to small subgroups of $\mathbb{F}_p^*$ and to elliptic curve groups.

21.7.14: Just call $A(g,(g^a)^{2^i},g^b)$ to get the i-th bit of the representation of g^{ab}.

21.7.15: Call $A(g,g^ag^z,g^b)$ about m times for uniformly random z. Each output yields a linear equation over $\mathbb{F}_2$ in the unknown bits of g^{ab}. Solving the system of linear

equations over $\mathbb{F}_2$ gives g^{ab}.

Chapter 22: Digital Signatures Based on Discrete Logarithms

22.1.2: The second and fourth are valid transcripts (i.e., the equation (22.1) is satisfied). In the third case one even has $\mathsf{s}_0 \notin \langle g \rangle$.

22.1.3: $A\mathsf{s}_2 + B - \mathsf{s}_2' \equiv a(A\mathsf{s}_1 - \mathsf{s}_1') \pmod{r}$ which can be solved for a.

22.1.6: If s_1 can be guessed then choose any $0 \leq \mathsf{s}_2 < r$ and set $\mathsf{s}_0 = g^{\mathsf{s}_2} h^{-\mathsf{s}_1}$. Send s_0 in the first stage of the protocol and respond to the challenge s_1 with s_2.

22.1.8: We need a multi-exponentiation (in equation (22.1)) and this is not well-defined in algebraic group quotients.

22.1.11: If m is a message and $(\mathsf{s}_1, \mathsf{s}_2)$ is a signature which is valid for two distinct public keys h_A and h_B then we have

$$\mathsf{s}_1 = H(\mathsf{m} \| g^{\mathsf{s}_2} h_A^{-\mathsf{s}_1}) = H(\mathsf{m} \| g^{\mathsf{s}_2} h_B^{-\mathsf{s}_1}).$$

If $\mathsf{s}_1 = 0$ then the signature is valid for all public keys. If $\mathsf{s}_1 \neq 0$ then $h_A^{-\mathsf{s}_1} \neq g^{\mathsf{s}_2} h_B^{-\mathsf{s}_1}$ and so we have a hash collision of a very special form: namely $H(\mathsf{m} \| R_1) = H(\mathsf{m} \| R_2)$ where R_1 and R_2 are distinct elements of $\langle g \rangle$. If the bit-length of the hash output is l such that $2^l < r$ then, by the pigeonhole principle, there must be distinct $R_1, R_2 \in \langle g \rangle$ such that $H(\mathsf{m} \| R_1) = H(\mathsf{m} \| R_2)$. Indeed, one expects many such pairs if 2^l is significantly smaller than r.

Even when 2^l is significantly larger than r then, by the birthday paradox, one expects there to be a collision of the form $H(\mathsf{m} \| R_1) = H(\mathsf{m} \| R_2)$. However, if 2^l is larger than r^2 then the probability of such a collision is rather low.

As for security, the existence of two keys for which the same signature is valid on the same message is not considered to lead to any practical attack in any real-world application. In any case, it appears to be impossible to construct the actual signatures, given the hash collision $H(\mathsf{m} \| R_1) = H(\mathsf{m} \| R_2)$, without solving at least two instances of the discrete logarithm problem.

22.1.12: Change s_2 to $k - a\mathsf{s}_1 \pmod{r}$. All the security arguments are unchanged by this.

22.2.2: Write the verification equation as

$$h^{F(\mathsf{s}_1)} \mathsf{s}_1^{\mathsf{s}_2} g^{r - H(\mathsf{m})} = 1.$$

If $\gcd(\mathsf{s}_2, \#G) = 1$, and h is known to lie in $\langle g \rangle$ then this equation (together with the possibly simpler check that $\mathsf{s}_1 \in G$) implies $\mathsf{s}_1 \in \langle g \rangle$.

One sees that Verify is computing a 3-dimensional multi-exponentiation with exponents all general elements of $\mathbb{Z}/r\mathbb{Z}$ and only one base fixed. Using basic methods (i.e., not windows or signed expansions) this computation will require roughly twice the cost of the computation in Example 22.1.13. Even when using more advanced methods such as windows, the fact that s_1 is a variable base in the multi-exponentiation is always a disadvantage. An additional cost in Elgamal signature verification (also for signing) is computing a hash function whose image is $\mathbb{Z}/r\mathbb{Z}$ rather than $\{0,1\}^l$.

22.2.3: Hint: Set $\mathsf{s}_1 = g^u h$ for random $0 \leq u < r$.

Set $\mathsf{s}_1 = g^u h$ for random $0 \leq u < r$. and $\mathsf{s}_2 = -F(\mathsf{s}_1) \pmod{r}$. One checks that

$$h^{F(\mathsf{s}_1)} \mathsf{s}_1^{\mathsf{s}_2} = h^{F(\mathsf{s}_1)} g^{-uF(\mathsf{s}_1)} h^{-F(\mathsf{s}_1)} = g^{-uF(\mathsf{s}_1)}.$$

Hence, taking $\mathsf{m} = -uF(\mathsf{s}_1) \pmod{r}$ gives the existential forgery, More elaborate versions of this attack are given in Section 4.2.3 of Elgamal [192].

22.2.4: Hint: Use the Chinese remainder theorem.

Given the public key h and a message with hash $H(\mathsf{m})$ the adversary chooses a random $1 \leq \mathsf{s}_1', \mathsf{s}_2 < r$ and computes $\mathsf{s}_1'' = (g^{H(\mathsf{m})} h^{-\mathsf{s}_1'})^{\mathsf{s}_2^{-1}} \pmod p$. Now, compute an integer s_1 using the Chinese remainder theorem so that

$$\mathsf{s}_1 \equiv \mathsf{s}_1'' \pmod p \quad \text{and} \quad \mathsf{s}_1 \equiv \mathsf{s}_1' \pmod r.$$

Note that, under our assumptions, $F(\mathsf{s}_1) = \mathsf{s}_1'$ and that $\mathsf{s}_1^r \equiv 1 \pmod p$. Then $h^{F(\mathsf{s}_1)} \mathsf{s}_1^{\mathsf{s}_2} = h^{\mathsf{s}_1'} g^{H(\mathsf{m})} h^{-\mathsf{s}_1'} \equiv g^{H(\mathsf{m})} \pmod p$ as required.

22.2.6: Hint: If $\mathsf{s}_1 = ur$ for any $u \in \mathbb{N}$ then $h^{F(\mathsf{s}_1)} = 1$.

Set $\mathsf{s}_1 = ur$ where $u \in \mathbb{N}$ is such that r divides the order of s_1 modulo p (this is easy). The trick is to set

$$g = \mathsf{s}_1^{(p-1)/r} \pmod p$$

which is a generator for the subgroup of order r. Use g as part of the system parameters of the scheme. The signature forgeries will all use the same value for s_1 (actually, s_1 can be varied by multiplying s_1 by any integer of order dividing $(p-1)/r$ modulo p, as long as one always treats s_1 as an integer and does not reduce modulo p). Note that, for any public key h, we have $h^{F(\mathsf{s}_1)} = 1$. Finally, set $\mathsf{s}_2 = H(\mathsf{m})(p-1)/r \in \mathbb{N}$ so that

$$h^{F(\mathsf{s}_1)} \mathsf{s}_1^{\mathsf{s}_2} \equiv \mathsf{s}_1^{\mathsf{s}_2} \equiv g^{H(\mathsf{m})} \pmod p.$$

A nice extension of the attack, which works even when the checks on s_1 and s_2 are performed, is to choose $\mathsf{s}_1 = ur$ so that $1 < \mathsf{s}_1 < p$ and so that the order of s_1 modulo p is equal to r. One can then take $g = \mathsf{s}_1$. Though it is easy to arrange that the order of s_1 equals r, it does not seem to be easy to do this while still keeping the integer ur less than p.

More details of these attacks, and variants in the case where g is a primitive root, are given in Bleichenbacher [66].

22.2.7: Try random $\mathsf{m}_1, \mathsf{m}_2$ until $r = |H(\mathsf{m}_1) - H(\mathsf{m}_2)|$ is prime. See Vaudenay [615].

22.2.8: The signatures are all valid with probability at most $1/r$. So suppose signature j is not valid. Choose all w_i for $i \neq j$ randomly. Then there is at most a $1/(r-1)$ chance that w_j is chosen so that the equation is satisfied.

When all $h_i = h$ just replace $\prod_{i=1}^t h_i^{w_i F(\mathsf{s}_{1,i})}$ by $h^{\sum_{i=1}^t w_i F(\mathsf{s}_{1,i})}$. One can now break the equation into $t/3$ 3-dimensional multi-exponentiations, so the total cost is about $1/3$ of the naive case.

For the third part see Yen and Laih [637].

It seems to be impossible to do something similar for Schnorr signatures since each $g^{\mathsf{s}_2} h^{-\mathsf{s}_1}$ needs to be computed separately.

22.2.12: First, precompute and store $g_1 = g^{\lceil \sqrt{r} \rceil}$. Then, for each signature $(\mathsf{s}_0, \mathsf{s}_2)$ to be verified run Euclid's algorithm on inputs $u_2 = F(\mathsf{s}_0)\mathsf{s}_2^{-1} \pmod r$ and r until the current remainder is approximately $\sqrt{r}$. Write v for the resulting coefficient of u_2 (in the notation of Section 2.3 this is $v = \mathsf{s}_i$ which clashes horribly with the notation of this chapter). By part 6 of Lemma 2.3.3 it follows that $v, vu_2 \pmod r \approx \sqrt{r}$. Also, write $u_1 v \equiv w_0 + w_1 \lceil \sqrt{r} \rceil \pmod r$ with $0 \leq w_0, w_1 < \sqrt{r}$. Equation 22.6 can therefore be written as

$$g^{w_0} g_1^{w_1} h^{u_2 v} \mathsf{s}_0^{-v} = 1$$

All exponents in this 4-dimensional multi-exponentiation are of size roughly $\sqrt{r}$. For further details see Antipa et al [11]. A further improvement in [11] (assuming the public key also contains a suitable power of h) leads to a 6-dimensional multi-exponentiation with exponents of size $r^{1/3}$.

22.2.13: There is no algorithm to generate signatures!

22.2.18: For example, to check that g^{a^2} is correct one can test whether $e(g_1, g_2^{a^2}) = e(\psi(g_2^a), g_2^a)$. The other elements are tested similarly. For the second part, $e\left(g_1^{(m+a)^{-1}}, g_2^a, g_2^m\right)$ should equal z.

Chapter 23: Public Key Encryption Based on Discrete Logarithms

23.1.5: The proof proceeds almost identically to the previous case, except that when a decryption query is made then only one call to the oracle is required. This variant is studied in Section 10.4 of Cramer and Shoup [161].

23.1.7: Given a Hash-DH instance (g, g^a, g^b), if one can compute g^{ab} then one can compute $\mathsf{kdf}(g^{ab})$. Given a DDH instance (g, g^a, g^b, g^c) one can compute $K = \mathsf{kdf}(g^c)$ and, using an oracle for Hash-DH, distinguish it from $\mathsf{kdf}(g^{ab})$.

23.2.2: For the first statement note that for each $0 \leq z_1 < r$ there is a unique choice for z_2. The second statement is straightforward. For the final statement write $g_2 = g_1^w$, $h = g_1^v$ and $u_2 = g^{k'}$ with $0 \leq k' < r$ and $k' \neq k$. The fact that $(z_1, z_2) \in \mathcal{X}_{g_1,g_2,h}$ imposes the linear equation $z_1 + wz_2 \equiv v \pmod{r}$. To prove the result we need to show that one can simultaneously solve $kz_1 + k'wz_2 \equiv x \pmod{r}$ for any $0 \leq x < r$. The result follows since the determinant of the matrix $\left(\begin{smallmatrix} 1 & w \\ k & k'w \end{smallmatrix}\right)$ is not zero modulo r.

23.2.7: First has $v \notin G$, second does not satisfy equation (23.1), third has message $\mathsf{m} = 1$.

23.2.11: The adversary returns $eu_1^{-z_1}u_2^{-z_2}$ just as the Decrypt algorithm does.

23.2.12: Given a challenge ciphertext (u_1, u_2, e, v) compute $u_1' = u_1 g_1, u_2' = u_2 g_2$ and $e' = eh$ (these are g_1^{k+1}, g_2^{k+1} and $\mathsf{m}h^{k+1}$ respectively). Then compute $\alpha' = H(u_1', u_2', e')$ and set $v = (u_1')^{x_1+y_1\alpha'}(u_2')^{x_2+y_2\alpha'}$. Calling the decryption oracle on (u_1', u_2', e', v') gives m.

23.2.13: Let u_1 be a random element of $\mathbb{F}_p^*$ of order l. Set $u_2 = u_1^a$ and $v = u_1^b$ for random $1 \leq a, b < l$ and choose any $e \in \mathbb{F}_p^*$. Call the decryption oracle on (u_1, u_2, e, v). With probability $1/l$ the decryption oracle does not return $\perp$, and indeed returns some message m. One therefore has

$$u_1^{r-z_1+a(r-z_2)} = \mathsf{m}e^{-1}.$$

Since it is easy to compute the discrete logarithm of $\mathsf{m}e^{-1}$ to the base u_1 when l is small one obtains a linear equation in z_1 and z_2 modulo l. Repeating the attack and solving gives $z_1 \pmod{l}$ and $z_2 \pmod{l}$.

If $p-1$ has distinct small prime factors $l_1, \ldots, l_t$ so that $\prod_{i=1}^t l_i > r$ then, by repeating the above attack, one can determine the private key uniquely using the Chinese remainder theorem.

23.3.7: Just set $\mathsf{c}_2' = \mathsf{c}_2 \oplus s$ for some non-zero string $s \in \{0,1\}^l$ and query the decryption oracle on $(\mathsf{c}_1, \mathsf{c}_2')$ to get $\mathsf{m} \oplus s$.

23.3.8: Given $Q_{\mathsf{id}} = H_1(\mathsf{id})$ set $R = \psi(g)Q_{\mathsf{id}}$. Invert the hash function to find an identity id' such that $H_1(\mathsf{id}') = R$. Then request the private key for identity id' to receive $R' = R^s = (\psi(g)Q_{\mathsf{id}})^s$. One can obtain $Q_{\mathsf{id}}' = Q_{\mathsf{id}}^s$ as $R'\psi(g')^{-1}$.

23.3.10: If one can solve CDH in G_T then compute $z = e(Q, g)$, $z_1 = e(Q, g^a) = z^a$ and $z_2 = e(Q, g^b) = z^b$. Then the solution to the CDH instance (z, z_1, z_2) is the required value. The case of CDH in G_2 is similar.

Chapter 24: The RSA and Rabin Cryptosystems

24.1.1: Repeatedly choose random primes p such that $2^{\kappa/2-1} < p < 2^{\kappa/2}$ and let $q = \lceil u2^{\kappa-l}/p \rceil + \epsilon$ (where $\epsilon \in \mathbb{Z}_{\geq 0}$ is small, possibly just always zero) until q is prime and the top l bits of pq are equal to u.

24.1.3: Generate random primes r_2 and r_3 of the required size. Generate a random prime r_1 of the form $2r_3x + 1$ for suitably sized $x \in \mathbb{N}$. Solve $p_0 \equiv 1 \pmod{r_1}$ and $p_0 \equiv -1 \pmod{r_2}$ using the Chinese remainder theorem. Try random $y \in \mathbb{N}$ of the appropriate size until $p = p_0 + r_1r_2y$ is prime. This algorithm is due to Gordon [264].

24.1.5: See Galbraith, Heneghan and McKee [220]. For parameter restrictions see [220] and [69].

24.1.7: One finds that $\mathsf{m} \equiv 385699 \pmod{p}$ and $\mathsf{m} \equiv 344504 \pmod{q}$. Solving $(385699 + px)^3 \equiv \mathsf{c} \pmod{p^2}$ gives $x = 1177$ and $\mathsf{m} \equiv 1234567890 \pmod{p^2}$. Performing another iteration of Hensel lifting gives the same value for m, as does the CRT. Hence, $\mathsf{m} = 1234567890$.

24.1.10: Instead of computing $\mathsf{c}^{d_p} \pmod{p}$ and $\mathsf{c}^{d_q} \pmod{q}$ for two primes $p, q \approx 2^{2500}$ we compute one exponentiation where p and d_p are 1/5 the size. Hence the cost is about $\frac{1}{2}(\frac{1}{5})^{2.58} \approx 0.008$ the time.

For the attack, choose an integer $\mathsf{m} > p$ (e.g., $\mathsf{m} = 2^{600}$), compute $\mathsf{c} = \mathsf{m}^e \pmod{N}$ and then ask for c to be decrypted. On receipt of the message $1 < \mathsf{m}' < p$ one knows $\mathsf{m}^e \equiv (\mathsf{m}')^e \pmod{p}$ and so $\mathsf{m}' \equiv \mathsf{m} \pmod{p}$. Hence, $\gcd(N, \mathsf{m} - \mathsf{m}')$ yields p.

24.1.11: An adversary can choose messages m_0 and m_1 such that $(\frac{\mathsf{m}_0}{N}) = -1$ and $(\frac{\mathsf{m}_1}{N}) = 1$ and then compute the Jacobi symbol of the challenge ciphertext to decide which message was encrypted.

24.1.18: Suppose A is a perfect algorithm that, on input an RSA public key (N, e), outputs the private key d. Let $N = pq$ be an integer to be factored. Choose $1 < e < N$ uniformly at random. If $A(N, e) = \perp$ then repeat for another choice of e (in this case one knows that $\gcd(e, \varphi(N)) \neq 1$; this information is potentially useful, but we ignore it in our proof). Let $d = A(N, e)$. It follows that $ed - 1$ is a multiple of $\lambda(N)$ and so the result follows from Lemma 24.1.17.

The expected number of trials to find e coprime to $\varphi(N)$ is bounded by $O(\log(N))$ (since that is the number of trials to choose a prime value $e > \sqrt{N}$). The reduction therefore requires at most polynomially many queries to the oracle A and at most polynomially many bit operations.

24.1.19: The idea is to note that $\varphi(N) = N - (p + q) + 1$ so one can compute p and q by taking the roots of the quadratic $x^2 + (\varphi(N) - N - 1)x + N$ (which can be done deterministically using numerical analysis).

24.1.20: Let $N = pq$ where p and q are odd. We use the same ideas as Lemma 24.1.17. One chooses a random $1 < g < N$ and computes $\gcd(g, N)$ (if this is not equal to 1 then we have factored N). Use A to determine the order M of g modulo N. With probability at least 3/4 one has M even (so if M is odd then repeat for a different choice of g). Now, with probability at least 1/2, $\gcd((g^{M/2} \pmod{N}) - 1, N)$ factors N (this final argument uses careful analysis of the 2-adic valuations of $p - 1$ and $q - 1$).

24.1.24: Given a small number of message-signature pairs $(\mathsf{m}_i, \mathsf{s}_i)$ one expects to find two messages $\mathsf{m}_i, \mathsf{m}_j$ such that $\gcd(H(\mathsf{m}_i), H(\mathsf{m}_j)) = 1$ (by Theorem A.14.4, the probabilty for each pair is at least 0.6). Euclid gives integers s, t such that $sH(\mathsf{m}_i) + tH(\mathsf{m}_j) = 1$. If s_i and s_j are the corresponding signatures then

$$(\mathsf{s}_i^s \mathsf{s}_j^t)^e \equiv a \pmod{N}$$

and so an e-th root of a has been computed. One can then forge signatures for any message.

24.2.5: $\mathsf{m} = 1234567890$ in all three cases.

24.2.6: One determines b_2 by computing $(\frac{\mathsf{c}}{N})$. One then computes $\mathsf{c}' = \mathsf{c}u_2^{-b_2}$ and determines b_1 by computing $(\frac{\mathsf{c}'}{p})$.

24.2.7: The advantage is that one speeds up the square roots modulo p, q and r since the primes are smaller (see Example 24.1.4). The main disadvantage is that there are now 8 square roots, in general, to choose from.

24.2.8: There are still only four square roots (two square roots modulo p, which correspond via Hensel lifting to two square roots modulo p^r, and similarly for q). Hence, one can use exactly the same type of redundancy schemes as standard Rabin. One speeds up computing square roots by using the Chinese remainder theorem and Hensel lifting as in Example 24.1.6. Hence, there is a significant advantage of using Rabin in this way.

24.2.17: Choose random $1 < x < N$, call oracle on $(x^2 \pmod N, -(\frac{x}{N}), 0)$ to get x' and compute $\gcd(x' - x, N)$.

24.2.21: Now there are nine possible roots to choose from. Redundancy in the message can be used in this case (the other two redundancy schemes are more directly related to square roots and cannot be used in this case).

For the security, choose a random integer $1 < x < N$ that does not satisfy the redundancy scheme and compute $\mathsf{c} = x^3 \pmod N$. Suppose a decryption oracle outputs a solution x' to $(x')^3 \equiv \mathsf{c} \pmod N$ so that x' satisfies the redundancy scheme. (This happens with probability at least $(1 - 1/2^l)^7 1/2^l$.)

Now $(x')^3 - x^3 = (x' - x)((x')^2 + x'x + x^2) \equiv 0 \pmod N$. If, say, $x' \equiv x \pmod p$ and $x' \not\equiv x \pmod q$ then one can split N. This situation occurs with probability 2/9. The idea of cubing is mentioned in Rabin [494].

24.2.23: Choose random x such that $(\frac{x}{N}) = -1$ and set $y = A(x^4, x^2, x^2)$. Then $\gcd(x - y, N)$ splits N with probability 1/2. This argument is due to Shmuely [552]. For precise details see Biham, Boneh and Reingold [57].

24.2.24: Compute $\varphi(N) = 2(N + 2) - M$ and then solve a quadratic equation to get p and q.

24.2.26: Use the same method as Exercise 24.2.23.

24.2.27: Since it is hard to choose random points in $E(\mathbb{Z}/N\mathbb{Z})$ one cannot apply the method of Shmuely with the first oracle. For the second oracle one can choose P and then fit the curve through it. Shmuely's method then applies.

24.3.1: Obviously, $(\mathsf{m}_1\mathsf{m}_2)^2 \equiv \mathsf{m}_1^2\mathsf{m}_2^2 \pmod N$. But we need to ensure that decryption of the product of the ciphertexts really does return the product of the messages. Since there is no guarantee that $\mathsf{m}_1\mathsf{m}_2 \pmod N$ has the correct bit pattern in the l least significant bits, redundancy in the message is not suitable for homomorphic encryption.

The "extra bits" redundancy does not work either, since the least significant bit of $\mathsf{m}_1\mathsf{m}_2 \pmod N$ may not be the product (or any other easily computable function) of the least significant bits of m_1 and m_2.

Finally, the Williams redundancy scheme is also not compatible with homomorphic encryption, since $P(\mathsf{m}_1)P(\mathsf{m}_2) \pmod N$ is almost always not equal to $P(\mathsf{m}_1\mathsf{m}_2) \pmod N$.

24.3.5: See Paillier [474].

24.3.7: One computes $\mathsf{c}^{p-1} \pmod{p^2}$ to get $1 + pq(p-1)\mathsf{m} \equiv 1 + p(-q\mathsf{m}) \pmod{p^2}$ and hence $\mathsf{m} \pmod p$. Similarly one computes $\mathsf{m} \pmod q$ and hence $\mathsf{m} \pmod N$. As with standard RSA decryption using the CRT, we replace one modular exponentiation with two modular exponentiations where the exponents and moduli are half the size. Hence, the new method is about 3 times faster than the old one.

24.3.8: The security depends on the decisional problem: Given y, is $y \equiv h^x \pmod{N^2}$ for some integer $0 \leq x < 2^k$. This problem is a variant of composite residuosity, and

also similar to the discrete logarithm problem in an interval; see Exercise 13.3.6 and Section 14.5 for algorithms to solve this problem in $O(2^{k/2})$ multiplications.

Suppose a user's public key consists of elements $h_i = u_i^N \pmod{N^2}$ for $1 \leq i \leq l$ (where the u_i are all random, or perhaps $u_i = u^{\lfloor N^{(i-1)/l} \rfloor} \pmod{N^2}$ for a randomly chosen $1 < u < N$). To encrypt to the user one could compute $\mathsf{c} = (1+N\mathsf{m})h_1^{a_1} \cdots h_l^{a_l} \pmod{N^2}$ using a sliding window multi-exponentiation method, where $0 \leq a_1, \ldots, a_l < 2^k$ for some value k (one possibility would be $2^k \approx N^{1/l}$).

24.3.11: One encrypts $0 \leq \mathsf{m} < N^k$ as

$$\mathsf{c} = (1 + \mathsf{m}N)u^{N^k} \pmod{N^{k+1}}$$

where $1 < u < N^k$ is chosen randomly. Decryption requires some care since

$$\mathsf{c}^{\lambda(N)} \equiv 1 + \lambda(N)\mathsf{m}N + \binom{\lambda(N)}{2}\mathsf{m}^2N^2 + \cdots \pmod{N^{k+1}}.$$

Hence, one first determines $\mathsf{m} \pmod{N}$ from the coefficient $\lambda(N)\mathsf{m}N$, and then $\mathsf{m} \pmod{N^2}$ from the coefficients of N and N^2, and so on. This variant is not used in practice, since messages are usually small for public key encryption.

24.3.12: Since $(u^N)^{p-1} \equiv 1 \pmod{p^2}$ for any $u \in (\mathbb{Z}/N\mathbb{Z})^*$ we have

$$g^{p-1} \equiv (1-p)^{p-1} \equiv 1 - (p-1)p \equiv 1 + p \pmod{p^2}.$$

It follows that $\mathsf{c}^{p-1} \equiv (1+p)^{\mathsf{m}} \equiv 1 + p\mathsf{m} \pmod{p^2}$. The homomorphic property follows since $g^{\mathsf{m}_1}u_1^N g^{\mathsf{m}_2}u_2^N = g^{\mathsf{m}_1+\mathsf{m}_2}(u_1u_2)^N$.

Let G be the subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ of order $(p-1)(q-1)$ containing all p-th powers in $(\mathbb{Z}/N\mathbb{Z})^*$. In other words, $g \notin G$. This subgroup is unique. Determining whether a ciphertext c encrypts a message m is precisely determining whether $\mathsf{c}g^{-\mathsf{m}} \in G$. Okamoto and Uchiyama call this the **p-subgroup problem**.

Finally, suppose one has a decryption oracle for this scheme. Choose an integer $x > N^{1/3}$, compute $\mathsf{c} = g^x \pmod{N}$, and let m be the message returned by the decryption oracle. Then $\mathsf{m} \equiv x \pmod{p}$ and so $p = \gcd(x - \mathsf{m}, N)$.

24.4.4: One uses the formula $\mathsf{c}(\mathsf{m}_1^{-1})^e \equiv \mathsf{m}_2^e \pmod{N}$ to obtain a time/memory tradeoff. We refer to [82] for discussion of the success probability of this attack.

24.4.5: 535573983004, 7873538602921, 8149260569118, 3195403782370.

24.4.6: Either set $F_1(x) = x^e - \mathsf{c}_1$ and $F_2(x) = (ax+b)^e - \mathsf{c}_2$ as before, or reduce to the previous case by multiplying c_2 by $a^{-e} \pmod{N}$.

24.4.7: To find m we construct the polynomials $F_1(x) = x^3 - \mathsf{c}_1$ and $F_2(x) = (x + 2^{10})^3 - \mathsf{c}_2$ and compute their gcd as polynomials in the ring $\mathbb{Z}_N[x]$. The result is

$$G(x) = x - 1234567890$$

from which we deduce that $\mathsf{m} = 1234567890$.

24.4.8: Write $F_1(x) = x^e - \mathsf{c}_1$ and $F_2(y) = y^e - \mathsf{c}_2$. Take the resultant of $F_1(x)$ and $P(x,y)$ with respect to x to obtain a polynomial $F(y)$ of degree de in y. Then compute the gcd of $F(y)$ and $F_2(y)$. The complexity is dominated by the computation of the resultant, which is the determinant of a $(d+e) \times (d+e)$ matrix whose entries are polynomials of degree at most d. Using naive Gaussian elimination gives the result. For further details see Coppersmith et al [144].

24.4.11: Let m be the message to forge and try to find $x, y \approx \sqrt{N}$ such that

$$(P + x) \equiv (P + \mathsf{m})(P + y) \pmod{N}.$$

Then x and y satisfy $x - y(P + \mathsf{m}) \equiv P^2 - P + P\mathsf{m} \pmod N$. In other words, we seek a small solution to $x - Ay \equiv B \pmod N$ for fixed A, B and N. We have seen how to solve such a problem in Section 11.3.2 under the name "Gallant-Lambert-Vanstone method".

24.4.12: Take $P = A^{-1}B \pmod N$.

24.5.1: One checks a guess for d by testing whether $y^d \equiv x \pmod N$ for some pre-computed pair $1 < x < N$ and $y = x^e \pmod N$. If one precomputes $y^2 \pmod N$ then one can compute the next value $y^{d+2} \pmod N$ using a single modular multiplication as $y^d y^2 \pmod N$. The total complexity is therefore $O(dM(\log(N)))$ bit operations.

24.5.5: Write $\lambda(N) = \varphi(N)/r$ where $r = \gcd(p-1, q-1)$ so that the equation $ed = 1 + k\lambda(N)$ corresponds to the equation $-edr + krN \approx kru$ (with $u \approx \sqrt{N}$). The Wiener method computes dr, as long as the condition $drkru < N$ holds or, in other words, if $d < N^{1/4}/(\sqrt{3}\, r)$. One can determine r as $\gcd(dr, N-1)$ and hence determine d.

24.5.6: One finds $\gcd(p-1, q-1) = 10$ and $d = 97$.

24.5.7: If (e, k) satisfy $ed = 1 + k\varphi(N)$ then so do $(e', k') = (e + l\varphi(N), k + ld)$. Taking l large enough one can ensure that $duk' > N$ and so the attack does not apply.

24.5.8: Choose random $1 < x < N$, set $y = x^e \pmod N$ and, for each odd integer $1 < d_p$ in turn, compute $\gcd(x - y^{d_p} \pmod N, N)$.

24.5.10: Since $0 \leq p + q < 3\sqrt{N}$ we have $0 \leq x + y \leq (p + q - 2)/r < \sqrt{3}N^{1/4}$ and so v and u give $x + y$ and xy exactly. One can therefore solve the quadratic equation $x^2 - vx + u$ to find x and hence p.

24.5.11: The first calculations are straightforward. The exhaustive search is performed by trying each value for c and testing whether $r^2c^2 + (2rv + 4)c + v^2 - 4u$ is a perfect square (for example, using the method of Exercise 2.4.9).

24.5.12: The exponent is $\mathrm{lcm}(p-1, q-1) = \mathrm{lcm}(xr, yr)$. Choose random $1 < z < N$, let $g = z^r \pmod N$ and $h = z^{ur} \pmod N$ we have $h \equiv g^c \pmod N$ where $0 \leq c < \sqrt{N}/r^2$. One therefore applies standard algorithms for the discrete logarithm problem, such as in Exercise 13.3.6 or Section 14.5. This gives c modulo the order of g. With overwhelming probability the order of g is much larger than $\sqrt{N}$ and so c is found.

24.5.13: Since we may assume one can factor $N - 1$ in under 2^{128} bit operations, it follows that a list of possible values for r is known. Assuming this list is short, one can apply the attack in Exercise 24.5.12 for each possible value for r, to compute p. The cost of the attack for a given guess for r is $O(N^{1/4}/r)$ modular multiplications. To have $N^{1/4}/r > 2^{128}$ means $r < 2^{3072/4-128} = 2^{640}$.

24.5.14: Small r can be found by factoring $N - 1$. Large r make N vulnerable to Pollard rho. with the map

$$x \mapsto x^{N-1} + 1 \pmod N.$$

If r is known then one can determine $p + q$ modulo r, and hence perform a similar attack to Exercise 24.5.12 that will split N in $O(N^{1/4}/r)$ ring operations if p and q are of similar size.

24.6.2: See Appendix A of Coron [149].

24.6.3: If $(\frac{h}{N}) = -1$ then set $f = 2$, otherwise $f = 1$. Then $(\frac{fh}{N}) = 1$. Now, if $(\frac{fh}{p}) = -1$ then set $e = -1$, otherwise $e = 1$. It follows that $(\frac{efh}{p}) = 1$ and $(\frac{efh}{q}) = 1$.

24.6.5: Use the fact that, for any $1 \leq a < p$, $(a^{(p+1)/4})^2 \equiv (\frac{a}{p})a \pmod p$.

24.6.13: If $e \mid H_2(\mathsf{m} \| \mathsf{s}_1)$ then one can compute the e-th root of $H_1(\mathsf{id})$, which is the private key of the user.

24.6.14: Generating a signature is the same as the original Shamir scheme. For the selective forgery, suppose $(\mathsf{s}_1, \mathsf{s}_2)$ is a valid signature for identity id on a message m where $\gcd(e, H_2(\mathsf{m})) = 1$. Let m' be the message to forge. There are integers $a, b \in \mathbb{Z}$ such that

$ae - bH_2(\mathsf{m}') = -H_2(\mathsf{m})$. Set $\mathsf{s}_1' = \mathsf{s}_1^b$ and $\mathsf{s}_2' = \mathsf{s}_2\mathsf{s}_1^a$ so that $(\mathsf{s}_1', \mathsf{s}_2')$ is a valid signature on m'.

24.6.15: The signature is $\mathsf{s} = s_{\mathsf{id}}^{H_2(\mathsf{m})} \pmod N$.

For the attack, suppose s_1 and s_2 are valid signatures for identity id on messages m_1 and m_2 such that $\gcd(H_2(\mathsf{m}_1), H_2(\mathsf{m}_2)) = 1$. Let $a, b \in \mathbb{Z}$ be such that $aH_2(\mathsf{m}_1)+bH_2(\mathsf{m}_2) = 1$. Then

$$\left(\mathsf{s}_1^a\mathsf{s}_2^b\right)^e \equiv H_1(\mathsf{id}) \pmod N$$

and the user's private key is obtained.

24.7.2: In the case where both top and bottom bits are 1 then let $\mathsf{c}' = \mathsf{c}2^e \pmod N$. Depending on the precise sizes of N and r, the decryption oracle on c' returns either $\mathsf{m}' = 2 + 4\mathsf{m} + 2^{258}r + 2^{3072}$ or $\mathsf{m}' - N$. Since N is odd it is easy to determine which case has arisen and, in the latter case, one simply adds N back again. Solving for m is immediate.

24.7.6: For the proof to go through it is necessary that $\kappa_1 > 2(\kappa + 2)/3$ and $n = \kappa - \kappa_0 - \kappa_1 < \kappa/9$. For full details see Boneh [74].

Chapter 25: Isogenies of Elliptic Curves

25.1.1: Since $\lambda(P) = \mathcal{O}_{\widetilde{E}}$ if and only if $P = \mathcal{O}_{\widetilde{E}}$ it follows that $\ker(\lambda \circ \phi) = \ker(\phi)$. On the other hand, $\ker(\phi \circ \lambda) = \lambda^{-1}(\ker(\phi))$. So if λ does not fix $\ker(\phi)$ then the isogenies are not equivalent.

25.1.3: Just add composition with a power of Frobenius.

25.1.4: The existence of ψ_1 and ψ_2 follows from applying Theorem 9.6.18 to $\phi_2 \circ \phi_1 : E \to E_2$.

25.1.9: The kernel of ϕ is a group of order G. Apply Vélu's formula and write the function X as a rational function in x. Once the result for $\phi_1(x)$ is proved the result for $\phi_2(x, y)$ follows from Theorem 9.7.5.

25.1.11: The calculation is essentially the same as a calculation in the proof of Theorem 9.7.5.

25.1.14: One performs d steps, each step being some arithmetic operations in $\mathbb{F}_{q^n}$. The x-coordinates of points in G are all roots of a polynomial of degree $(d-1)/2$ when d is odd or roots of $yF(x)$ where $\deg(F(x)) < d/2$. The y-coordinates are defined over quadratic extensions.

It is not quite true that $n < d$ in general. Indeed, $\mathbb{F}_{q^n}$ is a compositum of fields of degrees corresponding to the degrees of irreducible factors. Hence n can be bigger than d (e.g., $d = 5$ where the polynomial splits as quadratic times cubic). When d is prime then there are no such problems as the kernel subgroup is generated by a single x-coordinate and a quadratic extension for y.

25.1.17: Note that $t(Q) = 2F_x(Q) - a_1F_y(Q) = 2(3x_Q^2 + 2a_2x_Q + a_4 - a_1y_Q) - a_1(-2y_Q - a_1x_Q - a_3)$, and re-arranging gives the result. Similarly, $u(Q) = F_y(Q)^2 = (2y_Q + a_1x_Q + a_3)^2 = 4(y_Q^2 + y_Q(a_1x_Q + a_3)) + (a_1x_Q + a_3)^2$, and one can replace $y_Q^2 + y_Q(a_1x_Q + a_3)$ by $x_Q^3 + a_2x_Q^2 + a_4x_Q + a_6$.

25.1.18: The first statement is using $x_Q = x - (x - x_Q)$ the rest are equally easy. For example, the final statement follows from $x_Q^3 = x^3 - 3x^2x_Q + 3xx_Q^2 - (x - x_Q)^3$ together with some of the earlier cases.

25.1.20: Combine Theorem 25.1.6 with Exercise 25.1.17. Then simplify the formulae using Exercises 25.1.18 and 25.1.19. For details see pages 80-81 of [383].

25.2.2: One computes the powers of $j(E)$ in $\mathbb{F}_q$ in $O(\ell M(\log(q)))$ bit operations. In the polynomial $\Phi_\ell(x, y)$ there are $O(\ell^2)$ terms to consider and the coefficients are of size $O(\ell \log(\ell))$ and so reducing each coefficient to an element of $\mathbb{F}_q$ requires (the hardest

case being when q is prime and large) $O(\ell \log(\ell) \log(q))$ or $O(M(\ell \log(\ell)))$ bit operations. We also need to multiply each coefficient by a suitable power of $j(E)$. The total cost is therefore $O(\ell^2(\ell \log(\ell) \log(q) + M(\log(q)))$ bit operations. The resulting polynomial requires $O(\ell \log(q))$ bits. The claim about root finding is immediate from Exercise 2.12.5.

25.2.3: The first statement follows since $j(E) = j(E')$ and the $\mathbb{F}_q$-rational ℓ-isogenies are given by the roots of $\Phi_\ell(j(E), y)$. For the second statement, let $\phi : E \to E'$ be the isomorphism and consider the map $\mathrm{End}_{\overline{\mathbb{F}}_q}(E) \to \mathrm{End}_{\overline{\mathbb{F}}_q}(E')$ given by $\psi \mapsto \phi \circ \psi \circ \phi^{-1}$. One can check that this is a ring homomorphism and, since ϕ is an isomorphism, it is surjective.

25.3.15: The eigenvalues are $3, \sqrt{2}, 1, 0, -\sqrt{2}, -2$. We have $\lambda(X) = 2 < 2\sqrt{3-1}$ so the graph is Ramanujan. We have $\delta_v(\{1\}) = \{2, 4\}$, $\delta_v(\{1, 2\}) = \{3, 4, 5\}$ and $\delta_v(\{1, 3\}) = \{2, 4, 6\}$. The expander property is easy to verify. It also follows from equation (25.5) and $\lambda_1(X) = \sqrt{2}$; giving $\#\delta_v(A) \geq (3 - \sqrt{2})/6\#A > 0.26\#A$.

25.3.16: Let $n > 2/c$ be even and let X be the 2-regular "line" on n vertices (vertex $1 < i < n$ connected to vertex $i - 1$ and $i + 1$, and vertices 1 and n having single loops). Let $A = \{1, \ldots, n/2\}$ so that $\#A = n/2$ and $c\#A > 1$. But $\delta_v(A) = \{n/2 + 1\}$.

25.3.18: We have $h(-p) = 5$ and $\lfloor p/12 \rfloor + 1 = 9$, so there are nine isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{p^2}$, five of which are defined over $\mathbb{F}_p$.

Label the vertices of the graph as $24, 80, 23, 69, 34, \alpha, \overline{\alpha}, \beta$ and $\overline{\beta}$. The values α and $\overline{\alpha}$ are the roots of $t^2 + 84t + 73$, while β and $\overline{\beta}$ are roots of $t^2 + 63t + 69$. The graph is 3-regular. Vertices 24 and 80 have two loops, and an edge to 23. Vertex 23 has an edge to 69. Vertex 69 has a loop and an edge to 34. Vertex 34 has edges to α and $\overline{\alpha}$. Vertex α has edges to $\overline{\alpha}$ and β. Vertex $\overline{\alpha}$ has edges to α and $\overline{\beta}$. Finally, vertex β has two edges to $\overline{\beta}$.

25.3.19: The graph has $\lfloor p/12 \rfloor + 1 = 2$ vertices. So the vertices are $j = 0$ and $j = 1728 \equiv 1 \pmod{11}$. Using the modular polynomial $\Phi_2(x, y)$ one finds there are three edges from 0 to 1 and two edges from 1 to 0 (and a loop from 1 to itself). Hence, at least two of the three isogenies from 0 to 1 have the same dual isogeny.

25.3.20: Take $p = 131$. There are 10 supersingular j-invariants in $\mathbb{F}_p$. Taking 2-isogenies gives two components: one containing the j-invariants $\{25, 82\}$ and the other with $\{0, 10, 28, 31, 50, 62, 94, 113\}$.

25.4.9: The graph has two components. Both are trees with a root and 4 leaves. One component has root 42 and leaves $33, 51, 35, 57$. The other component has root 14 and leaves $44, 4, 18, 32$. The diameter of $X_{E,\mathbb{F}_q,\{2,3\}}$ is 2.

25.4.10: Clearly, $j = 11$ and $j = 31$ are on the floor. It follows that $\mathrm{End}(E)$ is the order of index 2 in $\mathbb{Z}[(1 + \sqrt{-7})/2]$. Hence, $\mathrm{End}(E) \cong \mathbb{Z}[\sqrt{-7}]$.

Since the isogenies from $j = 10$ to 11 and 31 are descending it follows that the ascending isogeny is to $j = 29$ and so this curve is on the surface.

25.5.2: The solution to the isogeny problem is a chain of length $O(\log(q))$ of prime-degree isogenies. Assuming that the chain is represented as a sequence of j-invariants, the cost of the Elkies and Vélu steps is $O(\ell^{2+\epsilon} \log(q)^{1+\epsilon})$ bit operations. Since $\ell = O(\log(q)^m)$ the cost for each isogeny is bounded by $O(\ell^{2m+1+\epsilon})$ bit operations. The total cost is therefore $O(\ell^{2m+2+\epsilon})$ bit operations.

25.5.3: Dijsktra's algorithm has complexity linear in the number of vertices, so it needs more than $\sqrt{q}$ operations to find the chain. The advantage is that the isogeny itself can be computed faster, but probably only by a constant factor.

Chapter 26: Pairings on Elliptic Curves

26.1.1: A function f such that $\mathrm{div}(f) = D_1$ is defined up to multiplication by an element of $\overline{\mathbb{k}}^*$. So let f be such a function and write $f(D_2) = \prod_P f(P)^{n_P}$. Then $(uf)(D_2) = u^{\sum_P n_P} f(D_2)$. The term $u^{\sum_P n_P}$ is 1 for all $u \in \overline{\mathbb{k}}^*$ if and only if $\deg(D_2) = 0$.

26.3.5: The fact that the divisors of the functions are correct is immediate. To show the functions are normalised at infinity it is necessary to show that the functions y and x are normalised at infinity. To see this note that $t_\infty^{-3} = (y/x)^3 = y^3/x^3 = y(x^3 + a_2x^2 + a_4x + a_6 - a_1xy - a_3y)/x^3 = y(1+u)$ where u is zero at $\mathcal{O}_E$. Hence, y is normalised at infinity. Similarly, $t_\infty^{-2} = (y/x)^2 = (x^3 + a_2x^2 + a_4x + a_6 - a_1xy - a_3y)/x^2 = x + u$ where $u(\mathcal{O}_E) = a_2$ and so x is normalised at infinity too. It follows that $l(x,y) = y - \lambda_x + c$ and $v(x,y) = x - c$ are normalised at infinity.

26.3.8: This follows since if $\mathrm{div}(f_{n,P}) = n(P) - n(\mathcal{O}_E)$ then $\mathrm{div}(f_{n,P}^m) = mn(P) - mn(\mathcal{O}_E)$. So take $m = N/n$.

26.3.10: Let $Q_1, Q_2 \in E[r]$ be such that $Q_1 \neq Q_2$. Suppose Q_1 and Q_2 were in the same class in $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. Then $Q_1 - Q_2 = [r]R$ for some $R \in E(\mathbb{F}_{q^k})$. It would follow that R has order r^2, but the conditions imply that no such group element exists.

26.3.12: If $v(x) = (x-a)$ is a vertical line function over $\mathbb{F}_q$ then $v(Q) = x_Q - a \in \mathbb{F}_{q^{k/2}}$. It follows that $v(Q)^{(q^k-1)/r} = 1$.

26.3.17: The first statement follows directly from the definition. To show part 2, first note that $\mathrm{div}(f_{s,x-s,Q}) = ([s]Q) - s(Q) + (s-1)(\mathcal{O}_E) = \mathrm{div}(f_{s,Q}^{-1})$. Now, $s \equiv q^m \equiv T^m \pmod r$ and so, by Exercise 26.3.14, we have a power of the ate pairing. Part 3 follows from

$$\mathrm{div}(f_{s,h(x)x,Q}) = \sum_{i=0}^{d} h_i(([s^{i+1}]Q) - (\mathcal{O}_E)) = \sum_{i=0}^{d} h_i(([s^i][s]Q) - (\mathcal{O}_E)) = \mathrm{div}(f_{s,h(x),[s]Q})$$

and the facts that

$$a_{s,h(x)}([s]Q, P) = a_{s,h(x)}(\pi_q^m(Q), P) = a_{s,h(x)}(Q,P)^{q^m} = a_{s,h(x)}(Q,P)^s$$

(this is essentially the same argument as in equation (26.5)). The additive property follows from the fact that the divisor of a product is the sum of the divisors. The multiplicative property follows from the additive property and from part 3.

26.5.3: Given $P, [a]P, [b]P$ choose a point Q such that $e(P,Q) \neq 1$ and one can invert pairings with respect to that point Q. Compute $z = e(P,Q)^{ab}$ as in Lemma 26.5.2, then call the pairing inversion oracle on (Q,z) to get $[ab]P$.