XTR and Tori*

Martijn Stam

Simula UiB, Bergen, Norway

At the turn of the century, 80-bit security was the standard. When considering discrete-log based cryptosystems, it could be achieved using either subgroups of 1024-bit finite fields or using (hyper)elliptic curves. The latter would allow more compact and efficient arithmetic, until Lenstra and Verheul invented XTR. Here XTR stands for 'ECSTR', itself an abbreviation for *Efficient and Compact Subgroup Trace Representation*. XTR exploits algebraic properties of the cyclotomic subgroup of sixth degree extension fields, allowing representation only a third of their regular size, making finite field DLP-based systems competitive with elliptic curve ones.

Subsequent developments, such as the move to 128-bit security and improvements in finite field DLP, rendered the original XTR and closely related torus-based cryptosystems no longer competitive with elliptic curves. Yet, some of the techniques related to XTR are still relevant for certain pairing-based cryptosystems. This chapter describes the past and the present of XTR and other methods for efficient and compact subgroup arithmetic.

1 The Birth of XTR

1.1 The Rise of Subgroup Cryptography

When Diffie and Hellman introduced the concept of public key cryptography [DH76], they gave the world the core key agreement mechanism still in use today. Suppose Anna and Bob have already agreed on a cyclic group of known order with generator g. For Anna and Bob to agree on a secret key, they both select a random exponent modulo the group order, say x for Anna and y for Bob, and send each other $X = g^x$ and $Y = g^y$, respectively. Upon receiving the other party's X, resp. Y value, Anna and Bob can raise it to their own private exponent to derive the shared key $k = X^y = Y^x = g^{xy}$.

Diffie and Hellman originally suggested to use the multiplicative group $\mathbb{Z}_p^*$ of integers modulo a large prime p as their cyclic group. This group has known order $p - 1$ and two problems relevant to Diffie–Hellman key agreement are believed to be hard in it: the discrete logarithm problem (given g^x, find x) and what later became known as the computational Diffie–Hellman problem (CDH; given g^x and g^y, find g^{xy}).

At the time, in 1976, not much was known about the discrete logarithm problem in $\mathbb{Z}_p^*$; from a mathematical perspective the group isomorphism from $\mathbb{Z}_{p-1}$ to $\mathbb{Z}_p^*$ by exponentiation of a known, fixed generator g was well understood and prior to Diffie and Hellman's breakthrough paper there appeared no urgent need to invert the isomorphism efficiently for what seemed like rather large primes p. Shanks's baby-step–giant-step method achieving "birthday" complexity $O(p^{1/2})$ had been published several years prior [Sha71] but that was about it. Naturally, the concept of key-agreement put forward by Diffie and Hellman changed this perception and soon the discrete logarithm problem in $\mathbb{Z}_p^*$ was studied in more detail, but for the moment a 200-bit modulus p seemed ok.

*This material will be published in revised form in Computational Cryptography edited by Joppe W. Bos and Martijn Stam and published by Cambridge University Press. See www.cambridge.org/9781108795937.

However, at the time Hellman had already submitted a paper together with Pohlig to speed the discrete logarithm problem in groups of composite order with known factorisation [**?**]. This Pohlig–Hellman algorithm would essentially solve the discrete logarithm problem in the prime power subgroups first and then use the Chinese remainder theorem to retrieve the discrete logarithm modulo the group order. The Pohlig–Hellman algorithm necessitates that $p-1$ has at least one large prime factor N, but working in $\mathbb{Z}_p^*$ still appeared fine. Indeed, when ElGamal turned the Diffie–Hellman key agreement protocol into a public key encryption scheme, he stuck to the $\mathbb{Z}_p^*$ setting.

It wasn't until 1989, when Schnorr introduced his eponymous signature scheme [Sch90] that working in a prime order subgroup of $\mathbb{Z}_p^*$ became popular. At the time, Schnorr suggested to use primes p and N with $N|p-1$, N around 140 bits and p around 512 bits (ostensibly targeting 70-bit security). The advantage of using these "Schnorr subgroups" was primarily computational: advances in solving the discrete logarithm problem had pushed up the size of p. Staying fully in $\mathbb{Z}_p^*$ would require working with exponents the size of p. Schnorr's innovation allowed to trim back the size of the exponent back to the minimum (i.e., twice the security level to protect against Shanks's generic baby-step–giant-step attack, or by that time, Pollard's rho [Pol78]), which also led to more compact signatures. Thus for *efficiency* reasons working in subgroups is beneficial.

For ElGamal encryption [ElG85], using Schnorr subgroups is mildly annoying as the message needs to be embedded in the subgroup; thus working in $\mathbb{Z}_p^*$ remained popular. However, in 1993 Brands introduced the decisional Diffie–Hellman problem [Bra93, Bon98], which is potentially much easier for an adversary. For this DDH problem, an adversary is still given g^x and g^y, but this time instead of having to compute g^{xy}, it is given a candidate value g^z and only needs to decide whether $g^z = g^{xy}$ or not. Five years later, Tsiounis and Yung [TY98] showed that the semantic security of ElGamal encryption is essentially equivalent to the decisional Diffie–Hellman problem in the group being used. Moreover, where the Pohlig–Hellman algorithm implied the hardness of the computational Diffie–Hellman problem was linked to the hardest subgroup, for decisional Diffie–Hellman problem it actually links to the *weakest* subgroup. For $\mathbb{Z}_p^*$ we are guaranteed a subgroup of order 2 and thus DDH is easy: ElGamal encryption as originally proposed is not semantically secure. Consequently, for *security* reasons working in subgroups became beneficial.

1.2 The Search for Compactness and Efficiency

While the role and need of Schnorr subgroups became ever clearer when building cryptosystems loosely based on the discrete logarithm problem in $\mathbb{Z}_p^*$, two other developments took place. On the one hand, improved subexponential algorithms for finding those logarithms were developed, culminating in the number field sieve. At the turn of the millenium, to achieve 80-bit security one would need a 1024-bit prime p but only a 160-bit prime N. On the other hand, a new competitor arrived by using elliptic curve groups instead.

Elliptic curve cryptography was introduced by Miller [Mil86] and Koblitz [Kob87] in the mid-eighties and was becoming more and more attractive: several families of "weak" curves had been identified, efficient point counting had been solved [Sch95], resulting in a large number of curves for which the best attacks were believed to be generic "birthday bound" ones such as Van Oorschot and Wiener's version of Pollard rho [vW94]. For a prime order subgroup G_N of an elliptic curve over a prime field $E(\mathbb{F}_p)$, this meant $N \approx p$ and believed discrete log complexity $\Theta(p^{1/2}) = \Theta(N^{1/2})$. Representing a point on the elliptic curve would naively takes two elements X and Y (not to be confused with the X and Y used previously for Diffie–Hellman key agreement), both in $\mathbb{F}_p$, satisfying the curve equation, e.g. $Y^2 = X^3 + aX + b$ for shortened Weierstrass when $\mathbb{F}_p$ is a large prime field. Those two elements take $2 \lg p$ bits, but it is straightforward to compress by representing only X and a bit indicating which root to take for Y, resulting in only $\lg p \approx \lg N$ bit

representations. Thus, for 80-bit security, only 160 bits are needed rather than the 1024 bits when using $\mathbb{Z}_p^*$'s Schnorr subgroups.

The problem with Schnorr subgroups of $\mathbb{Z}_p^*$ was that there was (and is) no known way to exploit being in a prime order subgroup, other than using smaller exponents: all operations and representations still rely directly on the supergroup $\mathbb{Z}_p^*$, resulting in wastefully large representations and, notwithstanding the smaller exponent, inefficient exponentiation. To counter these problems, Lenstra suggested to work in finite fields with small extension degrees instead [Len97]. Moreover, to avoid ending up in a smaller subfield, he suggested to work in a prime order subgroup of the cyclotomic subgroup (Definition 1).

Definition 1 (Cyclotomic Subgroup)**.** Let p be a prime power and let $n \in \mathbb{Z}_{\geq 1}$ be an extension degree. Then $|\mathbb{F}_{p^n}^*| = p^n - 1 = \prod_{d|n} \Phi_d(p)$ where Φ_d is the dth cyclotomic polynomial; the unique $\mathbb{F}_{p^n}^*$-subgroup of order $\Phi_n(p)$ is called the cyclotomic subgroup of $\mathbb{F}_{p^n}^*$.

If a prime N divides $\Phi_d(p)$ for $d|n, d \neq n$, then the subgroup G_N of order N can be embedded in a proper subfield $\mathbb{F}_{p^d}$ of the larger $\mathbb{F}_{p^n}$ and the discrete logarithm problem in G_N can be solved in that smaller subfield instead. The true hardness of the discrete logarithm problem in $\mathbb{F}_{p^n}^*$ ought to reside in those prime order subgroups that cannot be embedded into proper subfields. To support this claim, Lenstra also showed a bound on the greatest common prime divisor of $\Phi_d(p)$ and $\Phi_n(p)$.

Lemma 1 (Lenstra)**.** *Let $N > n$ be a prime factor of $\Phi_n(p)$. Then N does not divide any $\Phi_d(p)$ for divisors d of n with $d < n$.*

Lenstra demonstrated how exponentiation can be sped up moderately in cyclotomic subgroups. The cyclotomic subgroup of $\mathbb{F}_{p^n}$ has size $\Phi_n(p) \approx p^{\varphi(n)}$, which raises the prospect of utilizing its structure to represent elements of that subgroup more compactly as well, using $\varphi(n) \lg p$ bits instead of the naive $n \lg p$. If we assume that the discrete logarithm problem (and related Diffie–Hellman problems) are roughly as hard in a finite field $\mathbb{F}_{p^n}$ as they are in a prime field $\mathbb{Z}_{p'}$ with $p' \approx p^n$, such a compact representation would yield a compression by a factor $n/\varphi(n)$.

2 The Magic of XTR

The XTR cryptosystem was the first cryptosystem based on the finite field discrete logarithm problem that combined good compression, namely by a factor 3, with efficient exponentiation in the compressed form. XTR is shorthand for ECSTR, which itself stands for Efficient and Compact Subgroup Trace Representation and was developed by Arjen Lenstra and Eric Verheul [LV00b]. Sometimes XTR is also considered a homophone in Dutch of "ekster", or magpie.

The XTR cryptosystem works in the cyclotomic subgroup of $\mathbb{F}_{p^6}^*$, for prime p, although the generalization to extensions fields $\mathbb{F}_{q^6}^*$ with $q = p^m$ is mostly straightforward. XTR operates in a prime order N subgroup $\mathsf{G}_N \subset \mathsf{G}_{p^2-p+1}$ where N divides $\Phi_6(p) = p^2 - p + 1$. Elements in G_{p^2-p+1} can be compactly represented by their trace over $\mathbb{F}_{p^2}$, which is defined by

$$\mathrm{Tr} : x \to x + x^{p^2} + x^{p^4}\,.$$

Lenstra and Verheul showed that if $g \in \mathsf{G}_{p^2-p+1}$ and $c = \mathrm{Tr}(g)$, then g is a root of the polynomial

$$X^3 - cX^2 + c^pX - 1\,, \tag{1}$$

thus given c it is possible to recover g up to conjugacy, as g^{p^2} and g^{p^4} will also be roots of the equation above. Moreover, if $c_x = \mathrm{Tr}(g^x)$, they derived the recurrence relation

$$c_{x+y} = c_x c_y - c_x^p c_{x-y} + c_{x-2y}\,. \tag{2}$$

This recurrence relation allows fast "exponentiation" of c_x given a compressed base c and an exponent x without first having to decompress to obtain g (or one of its conjugates). We will discuss efficiency of XTR in Section 2.1.

Notwithstanding the title of the original paper "The XTR Public Key System", XTR is not really a cryptosystem, but rather a method to work efficiently and compactly in the "DLP-hard" part of $\mathbb{F}_{p^6}$. The main limitation of the XTR method is the kind of exponentiations that can be performed easily. Whereas a single exponentiation $\mathrm{Tr}(g^x)$ is easy, a double exponentiation $\mathrm{Tr}(g^x h^y)$ is already more challenging, whereas triple exponentiations and beyond are not really feasible in compressed format. As a consequence, not all discrete-log cryptosystems can be ported to the XTR setting.

One interpretation is to consider the exponent group $\mathbb{Z}_N$ acting on the set of trace-representations (cf. hard homogeneous spaces for isogeny-based cryptography [Cou06]). Cryptosystems that can be phrased using this abstraction, should be suitable for direct application of XTR. They notably include Diffie–Hellman key exchange and a variety of ElGamal-based KEMs. However, the ability to perform double exponentiations widens the scope to include for instance the Nyberg–Roeppel signature scheme. We give some concrete examples of XTRified schemes in Section 2.2.

The security of an XTR-based cryptosystem is tightly linked to the relevant underlying hard problem in $\mathsf{G}_N \subseteq \mathsf{G}_{p^2-p+1} \subseteq \mathbb{F}_{p^6}^*$, be it the discrete logarithm problem, the computational Diffie–Hellman problem, or its decisional version. Note that XTR predates the Diffie–Hellman alphabet soup assumption explosion triggered by the constructive use of elliptic curve pairings [Boy08]; although some of these newer assumptions do make sense in pairing-free groups like $\mathsf{G}_N \subseteq \mathbb{F}_{p^6}^*$, the cryptosystems based on those assumptions typically involve operations beyond the XTR-friendly single and double exponentiations.

Precursors. Some early cryptosystems predating XTR realized partial benefits from working in the cyclotomic subgroup of finite fields, without necessarily realizing the structure being exploited. LUC works in the cyclotomic subgroup of $\mathbb{F}_{p^2}$, achieving a compression factor 2, although historically it was not presented thus. Already in 1981, Müller and Nöbauer [MN81] suggested to replace the exponentiation in RSA by the evaluation of Dickson polynomials $g_x(1, h)$ modulo an RSA modulus, where x takes the place of the exponent and h that of the base. Twelve year laters, Smith and Lennon [SL93] suggested to use the Lucas function $V_x(h, 1)$ instead as an alternative to RSA, again still modulo an RSA modulus. They called this new cryptosystem LUC, yet, as $g_x(1, h) = V_x(h, 1)$ this Lucas cryptosystem was, and is, equivalent to the Dickson scheme.

Then, in 1994, Smith and Skinner [SS95] suggested the used of the Lucas function modulo a prime to be used for discrete-log based cryptosystems such as Diffie–Hellman key exchange or ElGamal encryption. They hoped that this 'prime' LUC cryptosystem would not allow any subexponential attack, oblivious of the mathematically much cleaner interpretation of LUC as being based on the cyclotomic subgroup of $\mathbb{F}_{p^2}$ using traces to compress and speed up calculations. The observation that LUC was in fact just $\mathbb{F}_{p^2}$ in disguise was first made by Lenstra, together with Bleichenbacher and Bosma [BBL95].

The first scheme achieving compression by a factor 3, using properties of the cyclotomic subgroup of $\mathbb{F}_{p^6}$, was proposed by Brouwer, Pellikaan, and Verheul [BPV99] in 1999. It formed the inspiration for XTR, but unlike XTR it did not offer computation in the compressed domain. A slightly different compression method was developed around the same time by Gong and Harn [GH99, GHW01], though it only offered a factor 1.5 compression.

For a slightly more expanded discussion of the history, see also [Sta03, Sections 4.4.3 and 4.6.4].

2.1 Efficient Implementation

When implementing XTR, essentially three efficient routines are needed: fast parameter generation of p, N and a base element c on the one hand, efficient membership tests, and efficient double exponentaion. It turns out that efficient single exponentiation is best regarded as a special, sped up version of double exponentiation.

Parameter generation and membership tests. Generating the primes p and N where N divides $p^2 - p + 1$ is relatively easy provided that N is considerably smaller than p. Select N first, find a root of r of $X^2 - X + 1$ modulo N and try $p = r + N\ell$ for integer ℓ until a suitable prime p is found.

To find c, the naive approach would be to sample g at random from $\mathbb{F}_{p^6}^*$, raise it to $(p^6 - 1)/(p^2 - p + 1) = (p^3 - 1)(p + 1)$ and verify whether the result has the desired order $p^2 - p + 1$. Subsequently take the trace and raise to the power $(p^2 - p + 1)/N$. Finding a generator of G_{p^2-p+1} this way is quite expensive and can in fact be done far more efficiently by exploiting fast irreducibility testing of the polynomial 1 on page 3, as explored by Lenstra and Verheul [LV00a, LV01].

For a membership test, the goal is to test whether a purported compressed element $c \neq 3$ is indeed the trace of an element in G_N. One could check this by evaluating c_N using a single exponentiaton routine (that does not first reduce the exponent modulo N) and checking whether the result equals $3 = \mathrm{Tr}(1)$. The techniques to generate c faster can also be used to speed up membership tests [LV01].

Single exponentiation. Normally, a single exponentiation refers to the problem of calculating g^x in the group G_N for given generator g of G_N and exponent $x \in \mathbb{Z}_N$. For XTR, the problem translates to calculating $c_x = \mathrm{Tr}(g^x)$ given compressed generator $c = \mathrm{Tr}(g)$ and exponent x, based on the recurrence relation for c_{x+y} given in Eq. 2 on page 3.

In general, evaluating $c_{x+y} = c_x c_y - c_x^p c_{x-y} + c_{x-2y}$ given all required elements on the right hand side, would take four $\mathbb{F}_p$-multiplications. However, if $x = y$, then the relation simplifies to $c_{2x} = c_x^2 - 2c_x^p$, which effectively only costs half as much, namely two $\mathbb{F}_p$-multiplications.

Armed with this knowledge, Lenstra and Verheul devised an elegant single exponentiation routine based on a left-to-right binary expansion of the exponent x by keeping track of the triplet $(c_{2k}, c_{2k+1}, c_{2k+2})$, where k is the exponent processed so far (corresponding to the most significant bits of x). A useful property of the triplet is that it always contains two even "exponents", namely $2k$ and $2k + 2$. As a result, processing the next bit of x can always be done using to invocations of the c_{2x} rule and one invocation of the general recursion rule. Overall, a single exponentiation would cost around $8 \lg N$ $\mathbb{F}_p$-multiplications. At the time, they believed this to be roughly three times faster than a direct exponentiation in G_N.

Double exponentiation. In the context of XTR, a double exponentiation consists of the problem of, given basis c_κ, c_λ, their "quotients" $c_{\kappa-\lambda}$ and $c_{\kappa-2\lambda}$ and two exponents x and y, compute $c_{\kappa x+\lambda y}$. The original double exponentiation by Lenstra and Verheul [LV00b] was somewhat cumbersome and slow. A far more efficient routine was developed by Stam and Lenstra [SL01] based on Montgomery's PRAC algorithm [Mon83]. The original PRAC algorithm was developed by Montgomery for the efficient calculation of Lucas sequences, that is, recurrence relations of the form $L_{x+y} = f(L_x, L_y, L_{x-y})$. These occur for instance when implementing scalar multiplication on elliptic curves using Montgomery representation.

At the core of the PRAC algorithm, including its XTR variant, is the extended Euclidean algorithm. Recall that to calculate the greatest common divisor of two positive integers x

and y, the extended Euclidean algorithm eventually outputs not just the gcd d, but also the Bézout coefficients a and b such that $ax+by = d$. To create a Euclidean exponentiation routine for exponents x and y given bases g^κ and g^λ, introduce random variables $\tilde{x}, \tilde{y}$ as well as α, β and keep as invariant $\tilde{x}\alpha + \tilde{y}\beta = x\kappa + y\lambda$ and $\gcd(\tilde{x}, \tilde{y}) = \gcd(x, y)$. It's easy to initialize by setting $\tilde{x} \leftarrow x$ and $\tilde{y} \leftarrow y$. The Greek lettered variables aren't typically known or efficiently computable, but we can keep track of g^α and g^β instead. Just as in the extended Euclidean algorithm, in each step $(\tilde{x}, \tilde{y})$ can be reduced while maintaining the invariant,e.g. by setting $(\tilde{x}, \tilde{y}) \leftarrow (\tilde{y}, \tilde{x} - \tilde{y})$ and updating the other variables accordingly. Eventually $y = 0$, at which point $g^{x\kappa+y\lambda} = (g^\alpha)^d$, where $d = \gcd(x, y)$. Adapting the algorithm to Lucas sequences, one also needs to keep track of $g^{\alpha-\beta}$; for XTR one additionally needs $g^{\alpha-2\beta}$ as well.

There are many different steps possible to reduce $(\tilde{x}, \tilde{y})$. We already mentioned $(\tilde{y}, \tilde{x}-\tilde{y})$, but if $\tilde{y}$ is even, say, one can also try $(\tilde{x}, \tilde{y}/2)$. Which steps to use when to reduce $(\tilde{x}, \tilde{y})$ are governed by rules that have been determined heuristically. We refer to [Sta03, Tables 3.4 and 4.2] for possible collections of rules (and their precedence). These rules result in an XTR double exponentiation on average costing roughly $6 \lg N$ $\mathbb{F}_p$-multiplications [Sta03, Corollary 4.13.ii].

Single exponentiation, revisited. The PRAC-based double exponentiation appears 25% faster than the binary single exponentiation. Unsurprisingly, it's possible to leverage this speedup of the double exponentiation for single exponentiation as well, by casting the latter as a case of the former by writing $g^x = g^r g^{x-r}$ for some arbitrary r. The choice $r = 1$ and a slight trimming of PRAC's Euclidean reduction rules will lead back to a (costly) binary method, but Montgomery already suggested the use of the golden ratio ϕ by setting r to $\lfloor r/\phi \rfloor$ to ensure that the PRAC algorithm will initially use its most advantageous reduction rule (i.e., delivering the largest reduction in the size of the exponent per $\mathbb{F}_p$-multiplication). With this speed up, calculating c_x costs an average $5.2 \lg N$ $\mathbb{F}_p$-multiplications [Sta03, Corollary 4.13.i].

Further speedups are possible when allowing precomputation, as it allows to split the exponent evenly: if the triplet $(c_{\tau-1}, c_\tau, c_{\tau+1})$ has been precomputed with $\tau = \lfloor \sqrt{N} \rfloor$, then one can write $x = x_1 + x_2\tau$ with both x_1 and x_2 of length roughly $\frac{1}{2} \lg N$. The resulting double exponentiation brings the costs of a single exponentiation with precomputation down to an average of $3 \lg N$ $\mathbb{F}_p$-multiplications [Sta03, Corollary 4.13.v].

Stam and Lenstra [SL01] introduced another neat trick to speed up and compress the precomputation by exploiting the Frobenius endomorphism. The idea is to rewrite the exponent as $x = x_1 + x_2 p \bmod N$ with x_1 and x_2 both as short as possible, so ideally roughly $\frac{1}{2} \lg N$ bits (as above). The difference between using Frobenius and $\tau = \lfloor N \rfloor$ is that for Frobenius, the precomputation is a lot simpler as $c_p = c^p$ and $c_{p-1} = c$ can be computed for free and, re-using an earlier observation by Lenstra and Verheul [LV01, Proposition 5.7], c_{p-2} can be computed at the cost of a square root computation (the only caveat here is that an additional bit is needed to resolve which root to use). Another difference is that, in case of Frobenius, it is not a priori clear whether x_1 and x_2 can computed that small. It turns out that the extended Euclidean algorithm suffices.

Coincidentally, concurrently and independently of the work on XTR, Gallant, Lambert, and Vanstone (GLV) suggested a very similar method to use efficient non-trivial automorphisms to speed up elliptic curve scalar point multiplication [GLV01]. The GLV method is more general than the method for XTR just described and, depending on the automorphism, the exponent might be split in more than two parts. For the special XTR setting, Stam and Lenstra exploited that $N | p^2 - p + 1$ to prove the split was guaranteed to result in short exponents x_1 and x_2 [Sta03, Lemma 2.29] (essentially by framing the problem of finding a short vector in a two-dimensional lattice). For the GLV method, Sica, Ciet and Quisquater [SCQ03] provided a more general analysis.

One potential disadvantage of the PRAC-based single exponentiation compared to the Lenstra–Verheul binary single exponentiation is the increased variation in not just the runtime of the algorithm, but also the sequence of underlying operations. From a side-channel perspective, such exponent-dependent variation should be considered leakage that can likely be exploited. Page and Stam [PS04] analysed idealized single power analysis (SPA) against XTR in more detail and although their attack was still computationally expensive, the binary routine appears intrinsically safer. In fact, a simpler version of the binary routine proposed by Montgomery for the efficient calculation of Lucas sequences [Mon83] has been adapted for ordinary scalar multiplication on elliptic curves with the express purpose of boosting side-channel resistance [JY03] and Han, Lim, and Sakurai showed that Lenstra and Verheul's binary XTR exponentiation routine is SPA secure, although it is still susceptible to differential power attacks [HLS04].

Twofold exponentiation. Knuth and Papadimitriou [KP81] established a beautiful result linking the complexity of a double exponentiation $g^x h^y$ with that of a twofold exponentiation (g^x, g^y), and vice versa. The result is that any improvement for double exponentiation can also be used for a twofold exponentiation, meaning that computing g^x and g^y jointly is a lot cheaper than computing both separately. For recurrence relations as used by XTR, the duality concept is not quite as clean, but jointly calculating c_x and c_y can still be done quite efficiently with a modification of the PRAC double exponentiation routine, leading to an overall cost of $6 \lg N$ $\mathbb{F}_p$-multiplications on average [Sta03, Corollary 4.13.iii].

2.2 Examples of XTRified Primitives

Diffie–Hellman key agreement. At the beginning of this chapter, we mentioned Diffie–Hellman key agreement. In a cyclic group G_N of order N and with generator g, Anna selects ephemeral exponent $x \in \mathbb{Z}_N$, calculates $X \leftarrow g^x$ and sends X to Bob. Bob had selected exponent $y \in \mathbb{Z}_N$ and sent $Y = g^y$ to Anna. They both want to compute the shared key g^{xy}.

Moving to XTR, we take traces of all the G_N group elements, so for instance the new shared key will be $\mathrm{Tr}(g^{xy})$. Moreover, the communication will be compressed as well: Anna selects ephemeral exponent $x \in \mathbb{Z}_N$ but this time calculates $c_x = \mathrm{Tr}(g^x)$. Upon receipt, Bob can then use c_x as its base XTR exponent 'd' and calculate $K \leftarrow d_y = \mathrm{Tr}(g^{xy})$ as desired.

The benefits of using XTR here over direct calculation in G_N are immediate: the public representation of the group parameters is compressed, the communication overhead is reduced by a factor of three and the trace-exponentiations are faster.

ElGamal-style key encapsulation mechanisms. Bare-bones ElGamal key encapsulation is almost the same as Diffie–Hellman key agreement, with the only notable difference that Bob's ephemeral value $c_y = \mathrm{Tr}(g^y)$ is now declared to be his public key, with the exponent y promoted to his long-term private key. If Anna wants to send a message to Bob, select ephemeral exponent $x \in \mathbb{Z}_q$, create ciphertext $c_x = \mathrm{Tr}(g^x)$ and calculate $K \leftarrow \mathrm{Tr}(g^{xy})$ as the encapsulated key.

It's immediate that Hashed ElGamal, where K is calculated as $\mathcal{H}(\mathrm{Tr}(g^{xy}))$ with $\mathcal{H}$ a hash function, works fine with XTR as well, giving rise to an efficient IND-CCA2 secure public key cryptosytem in the random oracle model. With a little more effort, Damgård's ElGamal with explicit ciphertext rejection can be seen to work as well, yet a closely related variant with implicit ciphertext rejection is more troublesome (cf. [KPSY09]).

Signature schemes. Lenstra and Verheul [LV00b] described how to create XTR-based Nyberg–Rueppel signatures [NR93], supporting message recovery. Below we look at Schnorr

signatures instead; these look similar to Nyberg–Rueppel signatures but without offering message recovery. They are created by applying the Fiat–Shamir transform to Schnorr's sigma protocol for proving knowledge of a discrete logarithm.

We assume that the group description consisting of p, N and $c_1 = \mathrm{Tr}(g)$ are public parameters. Then a user's private key is an exponent $x \in \mathbb{Z}_q$ and the corresponding public key is the element $c_x = \mathrm{Tr}(X)$ where $X = g^x$, plus the auxiliary elements $c_{x-1} = \mathrm{Tr}(X/g)$ and $c_{x+1} = \mathrm{Tr}(Xg)$.

The auxiliary elements enable the calculation of double exponentations of the type $\mathrm{Tr}(g^y X^z)$. If the public key is represented as the triplet $(c_{x-1}, c_x, c_{x+1}) \in (\mathbb{F}_{p^2})^3$ no compression is taking place and one might as well send $y \in \mathbb{F}_{p^6}$ instead. However, the public parameters are still being compressed (namely the generator of the group); moreover, Lenstra and Verheul also showed that a few additional bits allow unique and efficient recovery of c_{x-1} and c_{x+1} based on c_x and those public parameters.

To sign a message $\tilde{m}$, the signer generates a random exponent $w \in \mathbb{Z}_N$ and evaluates $a \leftarrow c_w = \mathrm{Tr}(g^w)$. Then calculate $s \leftarrow \mathcal{H}(c_x, a, \tilde{m})$ and $r \leftarrow w + x \cdot s \bmod N$, where $\mathcal{H} : \mathbb{F}_{p^2} \times \mathbb{F}_{p^2} \times \{0,1\}^* \to \mathbb{Z}_N$ is a hash function. The signature consists of the pair $(r, s) \in (\mathbb{Z}_N)^2$. To verify a signature, re-calculate $a \leftarrow \mathrm{Tr}(g^r \cdot X^{-s})$ using XTR double exponentiation and accept the signature iff $\mathcal{H}(c_x, a, \tilde{m})$ equals s.

As the original Schnorr signature only ever sent across elements of the exponent group $\mathbb{Z}_N$ to begin with, there is no compression gain to be had here; however both the public parameters and the public key *can* be compressed and both signature generation and verification are considerably sped up compared to naive operations directly in G_N.

3 The Conservative Use of Tori

3.1 Direct Compression using Tori

One downside of using traces as XTR does, is that the compression is not lossless: conjugates are mapped to the same compressed element. Another downside is that the exponentiation in compressed form relies on a third order recurrence relation and does not support arbitrary multiplications. As a consequence, some more complicated discrete-logarithm based cryptosystems cannot easily be implemented using XTR. Although arguably one could compute directly in G_N and only use the trace-plus-a-trit to compress and decompress elements (where the trit is used to indicate which conjugate has been compressed), a much neater lossless compression method is based on algebraic tori, as proposed by Rubin and Silverberg in 2003 [RS03].

They presented two new systems: $\mathbb{T}_2$ as an alternative to LUC (based on quadratic extension fields) and CEILIDH, as an alternative to XTR. CEILIDH, pronounced Cayley, was presented as an acronym for "Compact, Efficient, Improves on LUC, Improves on Diffie–Hellman", but was really named after Silberg's deceased cat [RS08]. It's unclear whether naming CEILIDH after an undoubtedly adorable cat was at all inspired by XTR being a homophone for a bird.

In any case, like XTR, CEILIDH allows to represent elements of $\mathsf{G}_N \subset \mathsf{G}_{p^2-p+1} \subset \mathbb{F}_{q^6}^*$ using only two elements of $\mathbb{F}_q$. Note that, whereas XTR is usually presented as defined over a prime field, for CEILIDH it is customary to allow any underlying finite field $\mathbb{F}_q$, including characteristic-p fields $q = p^m$ with extension degree $m > 1$. Lenstra and Verheul already observed the same generalization works in principle for XTR, the real advantage of CEILIDH is that the compression is injective, so given a compressed element it's always possible to uniquely recover the original element.

We refer to the original papers by Rubin and Silverberg [RS03, RS08] for a precise mathematical definition of algebraic tori (see also [Gal12]), but given a finite field $\mathbb{F}_q$ and its n-degree extension $\mathbb{F}_{q^n}$, one way of characterizing the algebraic torus $\mathbb{T}_n(\mathbb{F}_q)$ is as

the intersection of the kernels of the norm maps from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^d}$ for $d|n$ (Definition **??**, [RS08, Theorem 5.7.(ii)]). Moreover, the algebraic tori neatly coincide with the cyclotomic subgroups [RS08, Proposition 5.8], so for instance $\mathbb{T}_6(\mathbb{F}_p) = \mathsf{G}_{p^2-p+1}$.

Loosely speaking, an algebraic torus $\mathbb{T}_n$ is rational over $\mathbb{F}_q$ if there exist a map from $\mathbb{T}_n(\mathbb{F}_q)$ to $(\mathbb{F}_q)^{\varphi(n)}$ that is defined almost everywhere as quotients of polynomials and the same holds true for its inverse. The rationality of an algebraic torus can be exploited to compress its elements with compression factor $n/\varphi(n)$; moreover $\mathbb{T}_2$ and $\mathbb{T}_6$ are known to be rational, leading to compression factors 2 and 3, respectively. Moreover, Rubin and Silverberg provided concrete rational maps (in both directions) for $\mathbb{T}_2$ and $\mathbb{T}_6$, coining the latter system CEILIDH.

As mentioned already, CEILIDH stands for "Compact, Efficient, Improves on LUC, Improves on Diffie–Hellman". In a way, this acronym can be misleading as Diffie–Hellman is generally understood to be a key agreement protocol that can be phrased independent of the underlying group, LUC turned out to be an trace-based factor-2 compression method for "hard" subgroups of quadratic extension fields that additionally allowed efficient arithmetic on compressed form (just like XTR). In contrast, CEILIDH is really just a factor-3 compression/decompression method for "hard" subgroups of sextic extension fields without any efficient method for exponentiations. However, the perspective offered by algebraic tori is useful and, as we will discuss in Section 3.2 on the following page, one can complement CEILIDH to arrive at efficient torus-based cryptography.

Amortized compression. If an efficient compression mechanism exists for $\mathbb{T}_n(\mathbb{F}_q)$ with compression factor $n/\varphi(n)$ whenever the torus $\mathbb{T}_n$ is rational, a natural question is for which n the tori are rational and which compression factors can be achieved. From a practical perspective, it is advantageous to look at the smallest n achieving a certain compression factor, which boils down to looking at the products of the successive smallest primes. So first 2, then $6 = 2 \cdot 3$, and next up would be $30 = 2 \cdot 3 \cdot 5$ providing slightly better compression than XTR.

Pre-CEILIDH, Brouwer, Pellikaan, and Verheul [BPV99] conjectured that extension degree 30 would allow for improved compression, yet Verheul later seemingly changed his mind and, with Bosma and Hutton, argued that such a system is unlikely to exist [BHV02]. The introduction of algebraic tori by Rubin and Silverberg brought to bear a rich field of mathematics to draw upon to settle whether better compression for degree 30 is possible or not.

Rationality of algebraic tori has been well-studied and Voskresenskii had conjectured that $\mathbb{T}_n$ is rational for all n (actually, that's a consequence of the conjecture; the original statement is more general and does not restrict to finite fields). For n the product of at most two primes, the conjecture has been proven, thus the torus $\mathbb{T}_n$ is known to be rational (enabling CEILIDH). On the other hand, for n the product of three primes the conjecture is still open, in particular the case $n = 30$ is still open.

Yet it's still possible to obtain almost optimal compression, but with a small caveat. Van Dijk and Woodruff [vW04] point out that the tori $\mathbb{T}_n$ are known to be *stably* rational for all n, meaning that there exist rational maps from $\mathbb{T}_n(\mathbb{F}_q) \times \mathbb{F}_q^d \to \mathbb{F}_q^{\varphi(n)+d}$ for some $d \geq 0$. In many scenarios, this stable rationality allows compression for $\mathbb{T}_{30}(\mathbb{F}_q)$ that beats the factor 3 provided by XTR or CEILIDH.

Consider a hybrid encryption scheme where the ephemeral key is encapsulated using an element of $\mathbb{T}_{30}(\mathbb{F}_q)$ and the data is encrypted symmetrically leading to some ciphertext bitstring. One could peel of an appropriate amount of bits of said ciphertext bitstring, embed them into $\mathbb{F}_q^d$, and then compress that part of the ciphertext together with the $\mathbb{T}_{30}(\mathbb{F}_q)$ element. Thus the overhead of the key encapsulation is reduced by a factor $30/\varphi(30)$.

In the original paper, van Dijk and Woodruff showed a stably rational map for $\mathbb{T}_{30}(\mathbb{F}_q)$

with $d = 32$, but less than a year later the idea was considerably refined and d was reduced from 32 to 2 with as side-benefit much faster rational maps [vGP+05].

3.2 Efficient Arithmetic

When Lenstra and Verheul introduced XTR, they compared its efficiency relative to direct operation in uncompressed form, i.e., in $\mathbb{F}_{p^6}$. As a rough efficiency measure, one can count the number of $\mathbb{F}_p$ squarings and multiplications, as these tend to be the most costly (see also Chapter **??**). Cohen and Lenstra [CL87] had previously worked out that a $\mathbb{F}_{p^6}$-multiplication could be done in 18 $\mathbb{F}_p$ multiplications, whereas a $\mathbb{F}_{p^6}$-squaring only required 12 $\mathbb{F}_p$-multiplications. Using a standard square-and-multiply exponentiation routine, a single resp. double exponentiation directly in $\mathbb{F}_{p^6}$ would then cost $21 \lg N$ resp. $25.5 \lg N$ $\mathbb{F}_p$-multiplications (This is sligthly lower then the $23.4 \lg N$ resp. $27.9 \lg N$ when using 18 $\mathbb{F}_p$-squarings for a single $\mathbb{F}_{p^6}$ squaring [LV00b, Lemma 2.12]).

Speeding up the cyclotomic subgroup G_{p^2-p+1}. However, it turned out that also in non-compressed form, working in $\mathsf{G}_N \subset \mathsf{G}_{p^2-p+1}$ can be used to speed up calculations considerably. Stam and Lenstra [SL03] introduced a number of useful techniques in the case that $p \equiv 2 \bmod 9$ or $p \equiv 5 \bmod 9$. The congruence ensures that p generates $\mathbb{Z}_9^*$, facilitating the use of a normal base, which means that the Frobenius endomorphims can be evaluated essentially for free.

The first observation is that $p^2 - p + 1$ divides $p^3 + 1$, thus for an element $g \in \mathsf{G}_{p^2-p+1}$ it holds that $g^{p^3+1} = 1$, or equivalently $g^{-1} = g^{p^3}$. In other words, the Frobenius endomorphism can be used to invert essentially for free in the group G_{p^2-p+1}. As for elliptic curves, free inversions can be exploited for faster exponentiation based on signed representations of the exponent(s), such as the non-adjacent form (NAF) for single exponentation or the joint sparse form (JSF) for double exponentiation. The resulting speedup leads to an average cost of $18 \lg N$ resp. $21 \lg N$ $\mathbb{F}_p$-multiplications for a single, resp. double exponentiation.

The second observation is that, as for improved XTR precomputation, the Frobenius endomorphis can be used to split the exponent, similar to the GLV method for elliptic curves. Specifically, and exponent $x \in \mathbb{Z}_N$ can be rewritten as $x_1 p + x_0$ with $\lg x_1 \approx \lg x_0 \approx \frac{1}{2} \lg N$, thus replacing a single $\lg N$-bit exponentiation with a much faster double $\frac{1}{2} \lg N$-bit exponentiations. Combining this observation with using the Solinas's joint sparse form, the cost of a single exponentiation goes down to $10.5 \lg N$ $\mathbb{F}_p$-multiplications, i.e., already twice as fast as the benchmark originally used by Lenstra and Verheul.

Yet, the most surprising observation made by Stam and Lenstra is that for elements in G_{p^2-p+1}, the squaring operation can be simplified and sped up considerably, so it only takes 6 $\mathbb{F}_p$-multiplications, Combining with the previous two observations, the average cost of a single resp. double exponentiation in G_N can be brought down to $6 \lg N$ resp. $9 \lg N$ $\mathbb{F}_p$-multiplications [SL03, Theorem 4.31], which slightly faster than the original XTR exponentiation routines, yet slightly slower than the improved XTR routines.

Speeding up the algebraic torus $\mathbb{T}_6(\mathbb{F}_q)$. As CEILIDH is only a compression/decompression method for $\mathbb{T}_6(\mathbb{F}_q)$, a natural question is how the techniques to speed up calculations in G_{p^2-p+1} can be best combined with techniques to compress elements in $\mathbb{T}_6(\mathbb{F}_q)$. A straightforward approach would be to restrict CEILIDH to the case $q = p \equiv 2, 5 \bmod 9$, compress and decompress using CEILIDH, and perform all operations directly in G_{p^2-p+1}, possibly incurring some additional overhead when changing basis representations.

Granger, Page, and Stam [GPS04] provide a more detailed analysis. For instance, by treating $\mathbb{T}_6(\mathbb{F}_p)$ as part of $\mathbb{T}_2(\mathbb{F}_{p^3})$, it is possible to look at partial compression and

decompression and for some operations to work in $\mathbb{T}_2(\mathbb{F}_{p^3})$ on partially compressed elements (based on the work by Rubin and Silverberg on $\mathbb{T}_2$). Essentially, for exponentiations, working in G_{p^2-p+1} is alwasy optimal, but there is considerable scope in optimizing intermediate operations by changing representation as required. In that sense, these optimizations are reminiscent of mixed coordinates as used for fast scalar multiplication over elliptic curves.

One obvious downside of using the Stam–Lenstra techniques for G_{p^2-p+1} is the requirement that $q = p \equiv 2, 5 \bmod 9$ to achieve fast squarings in $\mathbb{T}_6(\mathbb{F}_q)$. Granger and Scott [GS10] partly resolved this restriction by showing a more general method that allowed squaring in $\mathbb{T}_6(\mathbb{F}_q)$ for only 6 $\mathbb{F}_q$-multiplications for any $q \equiv 1 \bmod 6$. Finally, Karabina [Kar13] suggested a slightly different partially compressed format using four $\mathbb{F}_q$ elements for $\mathbb{T}_6(\mathbb{F}_q)$ with, again, $q \equiv 1 \bmod 6$. Squarings with that representation cost as little as 4 $\mathbb{F}_q$-multiplications, but the representation does not allow for direct multiplication, necessitating decompression at an amortized cost of 3 $\mathbb{F}_q$-multiplications per decompression using Montgomery's simultaneous inversion trick. Thus, if for an exponentiation in $\mathbb{T}_6(\mathbb{F}_q)$ the number of $\mathbb{T}_6(\mathbb{F}_q)$-multiplications is less than two-thirds of the number of $\mathbb{T}_6(\mathbb{F}_q)$-squarings, the Karabina-representation might be fastest of all. However, optimal use of Montgomery's simultaneous inversion trick does require that all $\mathbb{T}_6(\mathbb{F}_q)$-multiplications are done in parallel. This restriction limits the kind of exponentation routines possible, plus it requires considerable storage.

4 Pairings with Elliptic Curves

4.1 An (In)Equivalence

XTR was first presented at Crypto 2000, moreover, XTR was presented first at Crypto 2000. This prime slot in the program allowed other cryptographers to respond with their thoughts on XTR during the very same conference at the rump session. Menezes and Vanstone [MV00] cheekily rebranded ECSTR as "Elliptic Curve Singular Trace Representation" suggesting that XTR can be considered as an elliptic curve in disguise. Of course, recurrence-based "prime" LUC turned out to be $\mathbb{F}_{p^2}$ exponentiation in disguise and elsewhere polynomial-based NTRU turned out to be a lattice cryptosystem in disguise, so it was perspective worth entertaining. Let's investigate a little further the reasons behind Menezes and Vanstone's observation.

First, recall that an elliptic curve $E_{a,b}$ over $\mathbb{F}_q$ consists of the points $(X, Y) \in {\mathbb{F}_q}^2$ satisfying the curve equation $Y^2 = X^3 + aX + b$ for shortened Weierstrass. Together with the point at infinity, these $\mathbb{F}_q$-rational points form an additive group. If $q = p^2$, then the order of this group lies in the Hasse interval $p^2 - 2p + 1, p^2 + 2p + 1]$, indeed one often writes the order as $p^2 - t + 1$ where t is the Frobenius trace number. If $t = p$, then we have that $p^2 - t + 1 = \Phi_6(p)$, so the order of the elliptic curve matches the order of cyclotomic subgroup G_{p^2-p+1}.

Menezes [Men93] characterized those curves as "Class Three" supersingular elliptic curves over $\mathbb{F}p^2$ with positive parameter t (as $t = p$ rather than $-p$). For those curves, the Menezes–Okamoto–Vanstone (MOV) embedding [MVO91] provides an efficient group isomorphism from the curves to the cyclotomic subgroup G_{p^2-p+1}. Thus any problem like DLP, CDH, or DDH that is easy in G_{p^2-p+1} will also be easy for the elliptic curve. During the Crypto 2000 rump session, Menezes and Vanstone suggested that the MOV embedding would be efficiently invertible, which would immediately imply that the DLP, CDH, and DDH problems are equally hard in G_{p^2-p+1} and in those supersingular curves. That would be bad news for XTR and CEILIDH, as the MOV embedding also implies that the DDH problem is in fact *easy* for those curves.

Verheul [Ver01, Ver04] promptly took up the challenge and he showed a rather remark-

able result, namely that if the MOV embedding is efficiently invertible, then the CDH problem is easy in both G_{p^2-p+1} and for those supersingular curves. Slightly more precise, those supersingular curves are known to be of the form $E_a(\mathbb{F}_{p^2}) : Y^2 = X^3 + a$ where $a \in \mathbb{F}_{p^2}$ is a square but not a cube in $\mathbb{F}_{p^2}$. Let's call such and a suitable and, as before, let $N|p^2 - p + 1$. Furthermore, denote with $\mathbb{G}_a[N]$ the subgroup of order N of the curve $E_a(\mathbb{F}_{p^2})$ (including the point at infinity). Then Verheul showed that if for some a there exists an efficiently computable injective homomorphism from $\mathsf{G}_N \subset \mathsf{G}_{p^2-p+1}$ to $\mathbb{G}_a[N]$, then the CDH problem in both G_N and $\mathbb{G}_a[N]$ is easy. As the CDH problem in both is still believed to be hard, the MOV embedding is unlikely to be efficiently invertible and thus the cyclotomic subgroup G_{p^2-p+1} is not an elliptic curve in disguise.

The original MOV embedding was based on the Weil pairing, which is an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathsf{G}_{p^2-p+1}$ where $\mathbb{G}$ denotes the elliptic curve group and Verheul's result could be rephrased in terms of the search of an efficient isomorphism $\psi : \mathsf{G}_{p^2-p+1} \to \mathbb{G}$. In modern parlance, the Weil pairing is symmetric, or of type 1 [GPS06a]. Asymmetric pairings have become more popular in practice: in those cases $\mathbb{G}_1$ and $\mathbb{G}_2$ are two distinct groups related to the elliptic curve (defined over $\mathbb{F}_q$) and the bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathsf{G}_{\Phi_n(q)} \subset \mathbb{F}^*_{q^n}$. Here n is known as the *embedding degree*, which is the smallest n such that N divides $\Phi_n(q)$. Note that, in the wider literature, the embedding degree is more commonly referred to as k, we use n instead to highlight the connection with the degree of the algebraic torus. Verheul's results were subsequently generalized and refined to this setting as well [GHV08, KKM13].

4.2 Improved Pairings

Once one realizes that elliptic curve pairings map into cyclotomic subgroups, or algebraic tori, one immediately obtains that the compression and efficiency techniques developed for those groups are applicable to the outputs of a pairing, leading to compressed pairings [SB04]. However, often the interaction between techniques for cyclotomic subgroups and pairings is a little bit more intricate [GPS06c].

A typical pairing evaluation consists of two phases: a Miller loop that returns a finite field element and a final exponentiation to the power $(q^n - 1)/N$ that ensures membership of the group $\mathsf{G}_N \subset \mathbb{T}_n(q)$. For the final exponentiation it is often possible to deploy ideas similar to those described in this chapter. Especially the improved squaring routines by Granger and Scott and the compressed squaring by Karabina are beneficial as they work for $p \equiv 1 \bmod 6$, as commonly used for pairing-friendly curves. See also the overview by Beuchat *et al.* [BPFCRH17] for details.

Similarly, exponentiations in pairing groups can benefit. When the embedding degree is some multiple of 6, both XTR and CEILIDH, as well as fast squarings in the cyclotomic subgroup, apply. Which method is fastest depends very much on the situation at hand. While XTR allows the fastest squarings, its hard to exploit the Frobenius endomorphism beyond a split of the exponent in two, moreover the output will be compressed and can be hard to process further. Trace-less computations can benefit from a higher dimensional 'GLV' split of the exponent, but the squarings are more expensive. See also the overview by Bos *et al.* [BCN17, Section 6.5] for some concrete comparisons.

Characteristic three. Galbraith [Gal01] advocated the use of supersingular curves in characteristic 2 or 3 for pairing-based cryptography. Here the binary case only has embedding degree $n = 4$, whereas the ternary case gives embedding degree $n = 6$. The higher embedding degree offsets working with the slightly awkward characteristic [GPS05]. Moreover, Duursma and Lee [DL03] proposed an efficient algorithm to compute pairings for supersingular curves over $\mathbb{F}_{3^m}$, adding to their popularity. As the embedding degree is $n = 6$, the target group of the pairing lies in $\mathbb{T}_6(\mathbb{F}_{3^m})$, which makes both XTR and CEILIDH relevant for the computations involved. Granger *et al.* [GPS06b] worked out

the details, including how to slightly improve the Duursma–Lee algorithm for this case by incorporating trace- or torus-based speedups as part of the pairing.

Surprisingly, Shirase *et al.* [SHH+07, SHH+08] showed that in some cases, even better compression is possible than the $\varphi(n)/n$ that was targeted by LUC, XTR, CEILIDH and van Dijk *et al.* The key observation for $\mathbb{T}_6(\mathbb{F}_q)$ is that, if $q = 3^{2k-1}$, then $\Phi_6(q) = q^2 - q + 1$ factors as $(q + 3^k + 1)(q - 3^k + 1)$. Elements in the $\mathbb{F}^*_{q^6}$ subgroup of order $(q - 3^k + 1)$ can then be represented by their trace over $\mathbb{F}_q$, rather than by their trace over $\mathbb{F}_{q^2}$ as would be the case for XTR. Thus a factor 6 compression is achieved. Karabina [Kar13] later extended the factor 6 compression using a torus-based approach, while also showing factor 4 for the binary case.

Unfortunately, we now know that the discrete logarithm problem in finite fields of small characteristic (e.g. 3) is essentially broken (see Section **??** on page ??), rendering these improvements moot.

Parameter generation. XTR, as originally proposed, focused on using a prime field $\mathbb{F}_p$, but Lenstra and Verheul already mentioned the use of an extension field $\mathbb{F}_q$ instead. One challenge in that case is parameter selection: if $q = p^m$, then one requires prime p and N such that $N|\Phi_{6m}(p)$. First selecting N and then ensuring that p is a root of Φ_{6m} modulo N only works if p is at least a few bits longer than N. However, larger m are especially interesting if they allow smaller p, potentially quite a bit smaller than N.

An alternative approach then is to simply select prime p first and factor $\Phi_{6m}(p)$ in the hope to find a prime divisor N of the desired. This approach was suggested by Lenstra [Len97] and adopted by van Dijk and Woodruff [vW04] for $\mathbb{T}_{30}$.

There is however a third approach, as pointed out by Galbraith and Scott [GS08, Section 9]. Pairing-friendly curves give suitable parameters for XTR (or CEILIDH), so the polynomial families of parameters used to generate those curves, can also be used to generate XTR parameters. Such an approach is comparable to the fast generation suggested by Lenstra and Verheul [LV00b, Algorithm 3.1.1], namely searching for r such that $N = r^2 - r + 1$ and $p = r^2 + 1$ are simultaneously prime (and $p \equiv 2 \bmod 3$). However, as Lenstra and Verheul already warn "such 'nice' p may be undesirable from a security point of view because they may make application of the Discrete Logarithm variant of the Number Field Sieve easier." As explained in Chapter **??** and also Section **??**, the NFS *has* improved for parameters for pairing-friendly curves, necessitating larger parameters for a given security level. Although the precise recommendations are still somewhat in flux [DMH+17, BD19, Gui20, SKSW20], for XTR and CEILIDH it seems prudent to restrict to the prime variants G_{p^2-p+1} and $\mathbb{T}_6(\mathbb{F}_p)$ and ignore "pairing-friendly" parameter sets.

5 Over the Edge: Cyclotomic Subgroups Recycled

When XTR was proposed in 2000, the standard security level was still 80 bits, elliptic curves were not all that widely deployed, and people hardly worried about postquantum security. RSA moduli were 1024 bits long and, similarly, 160-bit Schnorr subgroups were based on 1024-bit prime fields. In that historical context, for XTR over a finite field $\mathbb{F}_{p^6}$ with prime p, this meant that $6 \lg p \approx 1024$ and $\lg q \approx 160$. Thus the '160-bit' group elements in G_N could be represented using the trace with two elements in $\mathbb{F}_{p^2}$, for only about 340 bits in total.

Compared to compressed elliptic curve cryptography of the same strength, that is just over a doubling in the number of bits for a single group element, moreover at the time, an XTR exponentiation was faster than an elliptic curve scalar multiplication. For suitably chosen elliptic curves, the best known algorithms to solve CDH/DDH still entails solving an elliptic curve discrete logarithm problem (ECDLP); moreover solving the ECDLP appears

as hard as it is generically, thus its assumed hardness is governed by the birthday bound and, to achieve k-bit security, it suffices to work with a curve defined over a $2k$-bit field. In other words, barring the emergence of new cryptanalytical techniques or the advent of quantum computers, ellptic curves scale extremely well in the targeted security level. These days, the standard security level is 128 bits for the near term and 256 bits for long term. A natural question is therefore how the comparison between XTR, or working in cyclotomic subgroups/algebraic tori in general holds up against using elliptic curves.

For the finite field discrete logarithm problem, much better algorithms are known, as described in detail in Chapter **??**. This immediately implies that qualitatively XTR and related systems scale considerably less well, but how does the comparison evolve concretely? To answer this question, we will consider both the perspective from the turn of the century, as well as a modern one based on current understanding of the hardness of the various discrete logarithm problems.

Already in 2001, Lenstra provided estimates for different security levels based on understood performance of index calculus methods at the time, as well as extrapolating the observed rate of algorithmic and computational improvements for future predictions [Len01]. For prime field systems, he essentially used estimates for the NFS for factoring integers and divided by the extension degree as appropriate. Although he did not include $\mathbb{T}_{30}$ in his comparison, we used that same methodology below. For small characteristic fields, he used slightly different formulae to take into account known speedups for that case [Len01, Table 6].

For instance, in 2001, 128-bit security would require an RSA modulus between 2644 and 3224 bits [Len01, Table 1], which after division by six gives us $441 \leq \lg p \leq 538$ for an XTR prime field [Len01, Table 5]. Similarly, his then prediction for 2020 security levels was that we would need $550 \leq \lg p \leq 660$ for an XTR prime field. Thus already by 2001 standards, the comparison skews considerably in favour of elliptic curves. Representing a single group element is roughly four times as costly (in bits) for XTR than for ECC. Lenstra also provided run-time estimates for various algorithms [Len01, Table 10] supporting the conclusion that for higher security levels, ECC would handily beat XTR.

In Table 1 on the facing page, we summarize Lenstra's findings, but we added an initial column for 80-bit security, where for p^{30} we use the recommendation by van Dijk and Woodruff [vW04] (which was a bit lower, so an $\mathbb{F}_p$ element narrowly would fit in a single 32-bit word) and, for simplicity, pretend as if a single group element could be compressed all on its own. Additionally, we appended two columns with current NIST recommendations targeting either 128 or 256 bit security [Bar20, page 55] (these recommendations match the ones suggested by ECRYPT [ES18, Table 4.6] that have been reproduced as Table **??** on page ??). Moreover, for $\mathbb{T}_{30}$ the Granger–Vercauteren attack ([GV05], see Section **??** on page ??) and later developments need to be taken into account and the limited potential for higher compression is not worth the risk. In general, for $\mathbb{T}_6(q)$ having non-prime q seems to offer no discernible benefit over prime $\mathbb{T}_6$ or $\mathbb{T}_2$, unless the group arises in the context of an elliptic curve pairing, in which case $q = p^m$ for $m \in 2, 3, 4$ may be observed. Finally, the small characteristic case has been completely broken, as explained in Section **??** on page ??).

Acknowledgement. I am grateful to Eric Verheul and Robert Granger for their inspiration and assistance while writing this chapter.

References

[Bar20] Elaine Barker. Nist special publication 800-57 part 1, revision 5, recommendation for key management: Part 1 – general. Technical report, National

Table 1: Compact representations (in bits) for cyclotomic subgroups based on various security levels and estimates.

	recommended 80-bit in 2001	estimated 128-bit in 2001	predicted 128-bit for 2020	recommended 128-bit in 2020	recommended 256-bit in 2020
p	1024	2644–3224	3296–3956	3072	15360
p^2	512	1322–1612	1648–1978	1536	7680
p^6	340	882–1076	1100–1320	1024	5120
p^{30}	256	712–864	880–1056	*discouraged*	*discouraged*
3^{6m}	n.a.	1260–1565	1603–1954	**broken**	**broken**

Institute of Standards and Technology, 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5. 14

[BBL95] Daniel Bleichenbacher, Wieb Bosma, and Arjen K. Lenstra. Some remarks on Lucas-based cryptosystems. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 386–396, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany. 4

[BCN17] Joppe W. Bos, Craig Costello, and Michael Naehrig. Scalar multiplication and exponentiation in pairing groups. In Nadia El Mrabet and Marc Joye, editors, *Guide to Pairing-Based Cryptography*, chapter 6, pages **6**–1–**6**–23. CRC Press, 2017. 12

[BD19] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, October 2019. 13

[BHV02] Wieb Bosma, James Hutton, and Eric R. Verheul. Looking beyond XTR. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 46–63, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany. 9

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, Heidelberg, Germany, 1998. Invited paper. 2

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56, Egham, UK, September 1–3, 2008. Springer, Heidelberg, Germany. 4

[BPFCRH17] Jean-Luc Beuchat, Luis J. Dominguez Perez, Laura Fuentes-Castaneda, and Francisco Rodriguez-Henriquez. Final exponentiation. In Nadia El Mrabet and Marc Joye, editors, *Guide to Pairing-Based Cryptography*, chapter 7, pages **7**–1–**7**–28. CRC Press, 2017. 12

[BPV99] Andries E. Brouwer, Ruud Pellikaan, and Eric R. Verheul. Doing more with fewer bits. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 321–332, Singapore, November 14–18, 1999. Springer, Heidelberg, Germany. 4, 9

[Bra93] S. Brands. An efficient off-line electronic cash system based on the representation problem. CWI Technical report, CS-R9323, 1993. 2

[CL87] H. Cohen and A. K. Lenstra. Supplement to implementation of a new primality test. *Mathematics of Computation*, 48(177):S1–S4, 1987. 10

[Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. http://eprint.iacr.org/2006/291. 4

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1

[DL03] Iwan M. Duursma and Hyang-Sook Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 111–123, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany. 12

[DMH+17] Sylvain Duquesne, Nadia El Mrabet, Safia Haloui, Damine Robert, and Frank Rondepierre. Choosing parameters. In Nadia El Mrabet and Marc Joye, editors, *Guide to Pairing-Based Cryptography*, chapter 10, pages **10**–1–**10**–22. CRC Press, 2017. 13

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. 2

[ES18] ECRYPT and N. P. Smart. Algorithms, key size and protocols report. https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf, 2018. 14

[Gal01] Steven D. Galbraith. Supersingular curves in cryptography. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 495–513, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. 12

[Gal12] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, Cambridge, UK, 2012. 8

[GH99] Guang Gong and Lein Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, 45(7):2601–2605, 1999. 4

[GHV08] Steven D. Galbraith, Florian Hess, and Frederik Vercauteren. Aspects of pairing inversion. *IEEE Transactions of Information Theory*, 54(12):5719–5728, 2008. 12

[GHW01] Guang Gong, Lein Harn, and Huapeng Wu. The GH public-key cryptosystem. In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001*, volume 2259 of *LNCS*, pages 284–300, Toronto, Ontario, Canada, August 16–17, 2001. Springer, Heidelberg, Germany. 4

[GLV01] Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 190–200, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 6

[GPS04] Robert Granger, Dan Page, and Martijn Stam. A comparison of CEILIDH and XTR. In Duncan A. Buell, editor, *Algorithmic Number Theory – ANTS*, volume 3076 of *LNCS*, pages 235–249. Springer, Heidelberg, Germany, 2004. 10

[GPS05] Robert Granger, Dan Page, and Martijn Stam. Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three. *IEEE Transactions on Computers*, 54(7):852–860, 2005. 12

[GPS06a] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. http://eprint.iacr.org/2006/165. 12

[GPS06b] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, 2006. 12

[GPS06c] Robert Granger, Dan Page, and Nigel P. Smart. High security pairing-based cryptography revisited. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory – ANTS*, volume 4076 of *LNCS*, pages 480–494. Springer, Heidelberg, Germany, 2006. 12

[GS08] Steven D. Galbraith and Michael Scott. Exponentiation in pairing-friendly groups using homomorphisms. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 211–224, Egham, UK, September 1–3, 2008. Springer, Heidelberg, Germany. 13

[GS10] Robert Granger and Michael Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 209–223, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. 11

[Gui20] Aurore Guillevic. Pairing-friendly curves. https://members.loria.fr/AGuillevic/pairing-friendly-curves, 2020. 13

[GV05] Robert Granger and Frederik Vercauteren. On the discrete logarithm problem on algebraic tori. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 66–85, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 14

[HLS04] Dong-Guk Han, Jongin Lim, and Kouichi Sakurai. On security of XTR public key cryptosystems against side channel attacks. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04*, volume 3108 of *LNCS*, pages 454–465, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany. 7

[JY03] Marc Joye and Sung-Ming Yen. The Montgomery powering ladder. In Burton S. Kaliski Jr, Çetin K. Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 291–302, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany. 7

[Kar13] Koray Karabina. Squaring in cyclotomic subgroups. *Mathematics of Computation*, 82(281):555–579, 2013. 11, 13

[KKM13] Koray Karabina, Edward Knapp, and Alfred Menezes. Generalizations of verheul's theorem to asymmetric pairings. *Advances in Mathematics of Communications*, 7(1):103–111, 2013. 12

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987. 2

[KP81] D. E. Knuth and C. H. Papadimitriou. Duality in addition chains. *Bulletin of the European Association for Theoretical Computer Science*, 13:2–4, 1981. 7

[KPSY09] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 590–609, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany. 7

[Len97] Arjen K. Lenstra. Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields. In Vijay Varadharajan, Josef Pieprzyk, and Yi Mu, editors, *ACISP 97*, volume 1270 of *LNCS*, pages 127–138, Sydney, NSW, Australia, July 7–9, 1997. Springer, Heidelberg, Germany. 3, 13

[Len01] Arjen K. Lenstra. Unbelievable security. Matching AES security using public key systems (invited talk). In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 67–86, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. 14

[LV00a] Arjen K. Lenstra and Eric R. Verheul. Key improvements to XTR. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 220–233, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. 5

[LV00b] Arjen K. Lenstra and Eric R. Verheul. The XTR public key system. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 1–19, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany. 3, 5, 7, 10, 13

[LV01] Arjen K. Lenstra and Eric R. Verheul. Fast irreducibility and subgroup membership testing in XTR. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 73–86, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany. 5, 6

[Men93] A. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer, Boston, MA, 1993. 11

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany. 2

[MN81] W. Müller and R. Nöbauer. Some remarks on public-key cryptosystems. *Studia Scientiarum Mathematicarum Hungarica*, 16:71–76, 1981. 4

[Mon83] P. L. Montgomery. Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains. Revised (1992) version from `ftp.cw.nl:/pub/pmontgom/Lucas.ps.gs`, 1983. 5, 7

[MV00] A. Menezes and S.A. Vanstone. Ecstr (xtr): Elliptic curve singular trace representation. Presented at the Rump Session of Crypto 2000, 2000. 11

[MVO91] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *23rd ACM STOC*, pages 80–89, New Orleans, LA, USA, May 6–8, 1991. ACM Press. 11

[NR93] Kaisa Nyberg and Rainer A. Rueppel. A new signature scheme based on the DSA giving message recovery. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 58–61, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. 7

[Pol78] John M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32:918–924, 1978. 2

[PS04] Dan Page and Martijn Stam. On XTR and side-channel analysis. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 54–68, Waterloo, Ontario, Canada, August 9–10, 2004. Springer, Heidelberg, Germany. 7

[RS03] Karl Rubin and Alice Silverberg. Torus-based cryptography. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 349–365, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 8

[RS08] Karl Rubin and Alice Silverberg. Compression in finite fields and torus-based cryptography. *SIAM Journal on Computing*, 37(5):1401–1428, 2008. 8, 9

[SB04] Michael Scott and Paulo S. L. M. Barreto. Compressed pairings. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 140–156, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 12

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 2

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995. 2

[SCQ03] Francesco Sica, Mathieu Ciet, and Jean-Jacques Quisquater. Analysis of the Gallant-Lambert-Vanstone method based on efficient endomorphisms: Elliptic and hyperelliptic curves. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 21–36, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Heidelberg, Germany. 6

[Sha71] D. Shanks. Class number, a theory of factorization, and genera. In *1969 Number Theory Institute (Proceedings of Symposia in Pure Mathematics, Vol. XX, State University New York, Stony Brook, NY, 1969)*, pages 415–440. American Mathematical Society, New York, USA, 1971. 1

[SHH+07] Masaaki Shirase, Dong-Guk Han, Yasushi Hibino, Ho Won Kim, and Tsuyoshi Takagi. Compressed XTR. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 420–431, Zhuhai, China, June 5–8, 2007. Springer, Heidelberg, Germany. 13

[SHH+08] Masaaki Shirase, Dong-Guk Han, Yasushi Hibino, Howon Kim, and Tsuyoshi Takagi. A more compact representation of XTR cryptosystem. *IEICE Fundamentals of Electronics, Communications and Computer Sciences*, 91-A(10):2843–2850, 2008. 13

[SKSW20] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad S. Wahby. Pairing-Friendly Curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-08, Internet Engineering Task Force, 2020. Work in

Progress, https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-pairing-friendly-curves-08. 13

[SL93] Peter J. Smith and Michael J. J. Lennon. Luc: A new public key system. In E. Graham Dougall, editor, *Computer Security, Proceedings of the IFIP TC11, Conference on Information Security, IFIP/Sec*, volume A-37 of *IFIP Transactions*, pages 103–117. North-Holland, 1993. 4

[SL01] Martijn Stam and Arjen K. Lenstra. Speeding up XTR. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 125–143, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. 5, 6

[SL03] Martijn Stam and Arjen K. Lenstra. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In Burton S. Kaliski Jr, Çetin K. Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 318–332, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany. 10

[SS95] Peter Smith and Christopher Skinner. A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In Josef Pieprzyk and Reihaneh Safavi-Naini, editors, *ASIACRYPT'94*, volume 917 of *LNCS*, pages 357–364, Wollongong, Australia, November 28 – December 1, 1995. Springer, Heidelberg, Germany. 4

[Sta03] Martijn Stam. *Speeding Up Subgroup Cryptosystems*. PhD thesis, Technische Universiteit Eindhoven, 2003. 4, 6, 7

[TY98] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In Hideki Imai and Yuliang Zheng, editors, *PKC'98*, volume 1431 of *LNCS*, pages 117–134, Pacifico Yokohama, Japan, February 5–6, 1998. Springer, Heidelberg, Germany. 2

[Ver01] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 195–210, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. 11

[Ver04] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *Journal of Cryptology*, 17(4):277–296, September 2004. 11

[vGP+05] Marten van Dijk, Robert Granger, Dan Page, Karl Rubin, Alice Silverberg, Martijn Stam, and David P. Woodruff. Practical cryptography in high dimensional tori. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 234–250, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 10

[vW94] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, and Ravi S. Sandhu, editors, *ACM CCS 94*, pages 210–218, Fairfax, VA, USA, November 2–4, 1994. ACM Press. 2

[vW04] Marten van Dijk and David P. Woodruff. Asymptotically optimal communication for torus-based cryptography. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 157–178, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 9, 13, 14