Asymptotically Optimal Communication for Torus-Based Cryptography

Marten van Dijk[1,2] and David Woodruff *[1]

[1] MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, USA
marten@mit.edu, dpwood@mit.edu
[2] Philips Research Laboratories, Eindhoven, The Netherlands

Abstract. We introduce a compact and efficient representation of elements of the algebraic torus. This allows us to design a new discrete-log based public-key system achieving the optimal communication rate, partially answering the conjecture in [4]. For n the product of distinct primes, we construct efficient ElGamal signature and encryption schemes in a subgroup of $F_{q^n}^*$ in which the number of bits exchanged is only a $\phi(n)/n$ fraction of that required in traditional schemes, while the security offered remains the same. We also present a Diffie-Hellman key exchange protocol averaging only $\phi(n)\log_2 q$ bits of communication per key. For the cryptographically important cases of $n = 30$ and $n = 210$, we transmit a 4/5 and a 24/35 fraction, respectively, of the number of bits required in XTR [14] and recent CEILIDH [24] cryptosystems.

1 Introduction

In classical Diffie-Hellman key exchange there are two fixed system parameters - a large prime q and a generator g of the multiplicative group F_q^* of the field F_q. In [10], the idea of working in finite extension fields instead of prime fields was proposed, but no computational or communication advantages were implied. In [26] Schnorr proposed working in a relatively small subgroup of F_q^* of prime order, improving the computational complexity of classical DH, but requiring the same amount of communication.

In [4] it is shown how to combine these two ideas so that the number of bits exchanged in DH key exchange is reduced by a factor of 3. Specifically, it is shown that elements of an order r subgroup G of $F_{q^6}^*$ can be efficiently represented using $2\log_2 q$ bits if r divides $q^2 - q + 1$, which is one third of the $6\log_2 q$ bits required for elements of $F_{q^6}^*$. Since the smallest field containing G is $F_{q^6}^*$, one can show [13] that with respect to attacks known today, the security of working in G is the same as that of working in $F_{q^6}^*$ for r large enough. In [14, 15] the XTR public key system was developed using the method of [4] together with an efficient arithmetic to achieve both computational and communication savings. These papers also show how to reduce communication in ElGamal encryption and signature schemes in $F_{q^6}^*$.

* Supported by an NDSEG fellowship.

In [4] it was conjectured that one can extend this technique to any n by working in the subgroup of $F_{q^n}^*$ of order $\Phi_n(q)$, where $\Phi_n(x)$ denotes the nth cyclotomic polynomial. Since the degree of $\Phi_n(x)$ is $\phi(n)$, where ϕ is the Euler function, one could transmit a $\phi(n)/n$ fraction of the number of bits needed in classical DH, while achieving the same level of security. For n the product of the first k primes, $\phi(n)/n \to 0$ as $k \to \infty$, so the savings get better and better. In [3, 24], evidence that the techniques of [4] cannot generalize to arbitrary n was presented, and in [3, 24], some specific versions of the conjecture in [4] made in [3] were shown to be false. Also in [24, 25, 23] it is shown that the group of order $\Phi_n(q)$ is isomorphic to the well-studied algebraic torus $T_n(F_q)$ [30] and that a positive answer to the conjecture in [4] is possible if one can construct an efficient rational parameterization of $T_n(F_q)$. However, such a construction is only known when n is a prime power or the product of two prime powers, although it is conjectured to exist for all n [24, 30]. In [24] a construction is given for $n = 6$, which is the basis for the CEILIDH public-key cryptosystem. CEILIDH achieves the same communication as XTR with a few computational differences.

In this paper we finally break the "$n \leq 6$ barrier" by constructing, for every n, efficient ElGamal encryption and signature schemes in $F_{q^n}^*$, which require transmitting at most a $\phi(n)/n$ fraction of the bits required in their classical counterparts. Further, we present an asymptotical variant of DH key exchange in which the average number of bits exchanged per key approaches $\phi(n)\log_2 q$. The key property that we use is the fact that $T_n(F_q)$ is *stably rational* (see [30], section 5.1). Specifically, our enabling technique is the construction of efficiently computable bijections θ and θ^{-1} with

$$\theta : T_n(F_q) \times \left(\times_{d|n,\ \mu(n/d)=-1} F_{q^d}^* \right) \to \times_{d|n,\ \mu(n/d)=1} F_{q^d}^*,$$

where $\times$ denotes direct product, and μ is the Möbius function[3]. This allows us to bypass the torus conjecture of [24], by relaxing the problem of efficiently representing a single symbol of $T_n(F_q)$, to the problem of efficiently representing a sequence of symbols in $T_n(F_q)$. Our bijections enable us to compactly represent m elements of $T_n(F_q)$ with $(m\phi(n) + \sum_{d|n,\mu(n/d)=-1} d)\log q$ bits, which for large enough m, is roughly $\phi(n)\log q$ bits per element. We stress that while our key exchange protocol achieves the optimal $n/\phi(n)$ reduction factor asymptotically, our encryption and signature schemes achieve this even for the encrypting or signing of a single message.

Note that the domain and range of θ need not be isomorphic. Indeed, letting G_d denote the cyclic group of order d, if $n = 2$ and $q = 3$, then the domain of θ is isomorphic to $G_4 \times G_2$, while the range is isomorphic to G_8. We show, however, that θ can be decomposed into isomorphisms plus a map requiring a table lookup. We show how to choose q so that constructing and querying this table is extremely efficient.

[3] For an integer n, $\mu(n) = 1$ if $n = 1$, $\mu(n) = 0$ if n has a repeated factor, and $\mu(n) = (-1)^k$ if n is a product of k distinct primes (see [11], section 16.3).

Our choice of q and r for fixed n will also affect the security of our scheme. We give an efficient heuristic for choosing q and r for the practical cases of $n = 30$ and $n = 210$, where we achieve a communication reduction by factors of $15/4$ and $35/8$, respectively. Further, for any n, we give an efficient algorithm for choosing q and r with a theoretical guarantee on its performance. This latter algorithm is primarily of theoretical interest, showing how to optimally choose q and r when n tends to infinity for a sufficiently large security requirement.

While our main focus and contribution is on the communication complexity, we also calculate the amount of computation necessary to evaluate θ and θ^{-1} for general n, and we attempt to minimize the number of modular exponentiations. We show that our representation enjoys some of the same computational advantages of CEILIDH over XTR, including the ability to multiply elements of $T_n(F_q)$ directly. This allows us to come close to the non-hybrid version of ElGamal encryption in [24]. Indeed, in addition to constructing a hybrid ElGamal encryption scheme, we construct a scheme in which to encrypt m messages, we form m ElGamal encryptions in $T_n(F_q)$ plus one additional encryption using a symmetric cipher. Unfortunately, the computational complexity of our scheme is not that practical, whereas XTR for instance, permits very efficient computations if just exponentiation is required. For $n = 30$, we hand-optimize the computation of θ and θ^{-1}. Our analysis for general n shows that all of our protocols and algorithms are (theoretically) efficient in n and the sizes of q and r.

Outline: Section 2 discusses the algebraic and number-theoretic tools we use. In section 3 we construct the bijections θ and θ^{-1}. Section 4 shows how to choose system parameters to guarantee security and efficiency, giving both a practical algorithm for $n = 30$ and $n = 210$, and a theoretical algorithm for general n. In section 5 we discuss our cryptographic applications. Section 6 treats the computational complexity of our bijections, and we conclude in section 7.

2 Preliminaries

2.1 Cyclotomic Polynomials and Algebraic Tori

We first state a few facts about the cyclotomic polynomials. See [19] for more background.

Definition 1. *Let n be a positive integer and let $\zeta_n = e^{2\pi i/n}$. The nth cyclotomic polynomial $\Phi_n(x)$ is defined by:*

$$\Phi_n(x) = \prod_{1 \leq k \leq n,\ \gcd(k,n)=1} (x - \zeta_n^k).$$

It is easy to see that the degree of $\Phi_n(x)$ is $\phi(n)$, where ϕ is the Euler-totient function. We also have:

$$x^n - 1 = \prod_{d|n} \Phi_d(x),$$

and using the Möbius function μ,

$$\Phi_n(x) = \prod_{d|n}(x^d - 1)^{\mu(n/d)}.$$

It can be shown that the cyclotomic polynomials are irreducible polynomials over $\mathbb{Q}$ with integer coefficients. For q a prime power, let F_q denote the finite field with q elements. For integers $n > 0$ we define the algebraic torus[4] $T_n(F_q)$:

$$T_n(F_q) = \{\alpha \in F_{q^n}^* \mid \alpha^{\Phi_n(q)} = 1\}.$$

2.2 Number Theory

The following is the celebrated prime number theorem (see [11], chapter 22):

Theorem 1. *For large enough n, the number of primes less than or equal to n is $\frac{n}{\ln n} + o\left(\frac{n}{\ln n}\right)$.*

We also need the fact that for any $n > 6$, $\phi(n) > n/(6 \ln \ln n)$, and for n the product of the first k distinct primes, $\phi(n) = \Theta(n/\log \log n)$. We use the following density theorem in our analysis:

Theorem 2. (Chebotarev [5, 16]) *For any integer n and any $a \in Z_n^*$, the density of primes p (among the set of all primes) with $p = a \mod n$ is $1/\phi(n)$.*

3 The Bijection

Let q be a prime power, n a positive integer, $F_{q^n}^*$ the multiplicative group of the field of order q^n, and $T_n(F_q)$ the $\phi(n)$-dimensional algebraic torus over F_q. For an integer k, let $[k] = \{1, 2, \ldots, k\}$. The goal of this section is to construct efficiently computable bijections θ and θ^{-1}, where

$$\theta : T_n(F_q) \;\times\; \left(\times_{d|n,\ \mu(n/d)=-1}\ F_{q^d}^*\right) \;\rightarrow\; \times_{d|n,\ \mu(n/d)=1}\ F_{q^d}^*.$$

Our strategy is to first find efficient bijections γ and γ^{-1}, where

$$\gamma : F_{q^n}^* \;\rightarrow\; \times_{d|n} T_d(F_q).$$

Note that in general $F_{q^n}^*$ and $\times_{d|n} T_d(F_q)$ need not be isomorphic. Let G_m denote the cyclic group of order m. We first need a few lemmas. The following is an immediate consequence of the structure theorem of abelian groups, but for completeness and to exhibit the efficient isomorphisms, we include it:

Lemma 1. *Suppose $n = r_1 \cdot r_2 \cdots r_k$ for pairwise relatively prime positive integers $r_1, \ldots, r_k$. Then there exist efficiently computable isomorphisms $\rho : G_n \rightarrow \times_{i\in[k]} G_{r_i}$ and $\sigma : \times_{i\in[k]} G_{r_i} \rightarrow G_n$.*

[4] Technically, $T_n(F_q)$ just refers to the F_q points of the algebraic torus rather than the torus itself (see [24, 30]).

Proof. For $i \in [k]$, put $d_i = n/r_i$. Since the r_i are pairwise relatively prime, $\gcd(d_1, d_2, \ldots, d_k) = 1$, so there exist integers e_i for which $\sum_{i\in[k]} e_i d_i = 1$. For $\alpha \in G_n$, define $\rho(\alpha) = (\alpha^{d_i})_{i\in[k]}$. Since $(\alpha^{d_i})^{r_i} = 1$, ρ maps elements of G_n to elements in the product group $\times_{i\in[k]} G_{r_i}$. For $(\alpha_i)_{i\in[k]} \in \times_{i\in[k]} G_{r_i}$, define $\sigma((\alpha_i)_{i\in[k]}) = \prod_{i\in[k]} \alpha_i^{e_i}$, where multiplication occurs in G_n.

The claim is that ρ and σ are inverse isomorphisms between G_n and $\times_{i\in[k]} G_{r_i}$. For $\alpha \in G_n$, we have $\sigma(\rho(\alpha)) = \sigma((\alpha^{d_i}))_{i\in[k]} = \prod_{i\in[k]} \alpha^{d_i e_i} = \alpha$. Similarly, for $(\alpha_i)_{i\in[k]} \in \times_{i\in[k]} G_{r_i}$, we have $\rho(\sigma((\alpha_i)_{i\in[k]})) = \rho(\prod_{i\in[k]} \alpha_i^{e_i}) = (\prod_{j\in[k]} \alpha_j^{e_j d_i})_{i\in[k]}$. Now, $r_j \mid d_i$ if $j \neq i$, so in this case $\alpha_j^{e_j d_i} = 1$. Also, $\alpha_i^{e_i d_i} = \alpha_i^{1-\sum_{j\neq i} e_j d_j} = \alpha_i^{1-kr_i}$ for an integer k, so $\alpha_i^{e_i d_i} = \alpha_i$. Hence, $\rho(\sigma((\alpha_i)_{i\in[k]})) = (\alpha_i)_{i\in[k]}$, which shows ρ and σ are inverses. Observe that $\rho(\alpha_1 \cdot \alpha_2) = ((\alpha_1 \cdot \alpha_2)^{d_i})_{i\in[k]} = (\alpha_1^{d_i})_{i\in[k]} \cdot (\alpha_2^{d_i})_{i\in[k]} = \rho(\alpha_1)\cdot\rho(\alpha_2)$, and similarly $\sigma((\alpha_i)_{i\in[k]} \cdot (\alpha_i')_{i\in[k]}) = \prod_{i\in[k]} (\alpha_i \cdot \alpha_i')^{e_i} = \prod_{i\in[k]} (\alpha_i)^{e_i} \prod_{i\in[k]} (\alpha_i')^{e_i} = \sigma((\alpha_i)_{i\in[k]}) \cdot \sigma((\alpha_i')_{i\in[k]})$, which shows that the maps are isomorphisms. Computing ρ and σ just requires multiplication and exponentiation, which can be made efficient by repeated squaring.

Let $U = U(n,q)$ be the smallest positive integer for which $\gcd(\Phi_d(q), \Phi_e(q), \frac{q^n-1}{U}) = 1$ for all $d \neq e$ with $d \mid n$ and $e \mid n$.

Lemma 2. *For $d \mid n$, let $y_d = \gcd(\Phi_d(q), \frac{q^n-1}{U})$. Then $F_{q^n}^* \cong G_U \times \left(\times_{d\mid n} G_{y_d}\right)$. Furthermore, the isomorphisms are efficiently computable.*

Proof. By lemma 1 it suffices to show (1) $q^n - 1 = U \prod_{d\mid n} y_d$, (2) for all d, $\gcd(U, y_d) = 1$, and (3) for all $d \neq e$, $\gcd(y_d, y_e) = 1$.

Using the fact that $q^n - 1 = \prod_{d\mid n} \Phi_d(q)$, the following establishes (1):

$$\frac{q^n-1}{U} = \gcd\left(\prod_{d\mid n} \Phi_d(q), \frac{q^n-1}{U}\right) = \prod_{d\mid n} \gcd\left(\Phi_d(q), \frac{q^n-1}{U}\right) = \prod_{d\mid n} y_d,$$

where the second equality follows from the definition of U. For (2), observe that

$$\gcd(U, y_d) = \gcd\left(U, \Phi_d(q), \frac{q^n-1}{U}\right) \mid \gcd\left(U, \frac{q^n-1}{U}\right) = 1,$$

since if prime $p \mid U$, by minimality of U there exist $d \neq e$ for which $p \mid \gcd(\Phi_d(q), \Phi_e(q))$, so if $p \mid \frac{q^n-1}{U}$, then $p \mid \gcd(\Phi_d(q), \Phi_e(q), \frac{q^n-1}{U})$, a contradiction. To see (3), note that $\gcd(y_d, y_e) = \gcd(\Phi_d(q), \Phi_e(q), \frac{q^n-1}{U}) = 1$ by the definition of U.

We use the following bijections with complexity proportional to U, which we later show to be negligible for an appropriate choice of q.

Lemma 3. *For $d \mid n$, let $z_d = \gcd(\Phi_d(q), U)$. There exist bijections between G_U and $\times_{d\mid n} G_{z_d}$ requiring $O(\log U + \log n + \log\log q)$ time to evaluate and $O(Un^{1+\epsilon} \log q)$ space for any $\epsilon > 0$.*

Proof. Using the definition of U,

$$\prod_{d|n} |G_{z_d}| = \prod_{d|n} \gcd(\Phi_d(q), U) = \gcd\left(\prod_{d|n} \Phi_d(q), U\right) = \gcd(q^n - 1, U) = U,$$

so there exists a bijection between the two groups. Choose a generator g of G_U and generators g_d of G_{z_d}. For each $i \in [U]$, make a table entry mapping g^i to a unique tuple $(g_d^{i_d})_{d|n}$. Since the sum of the divisors of n is less than $O(n^{1+\epsilon})$ for any $\epsilon > 0$ ([11], section 18.3), the table consumes $O(Un^{1+\epsilon} \log q)$ space. We sort the entries in both directions so that both bijections are efficient. Evaluations of either bijection can then be performed with a binary search in $O(\log U + \log n + \log\log q)$ time.

We need another auxiliary map:

Lemma 4. *Let y_d and z_d be as in the previous two lemmas. Then, $\times_{d|n} T_d(F_q) \cong \left(\times_{d|n} G_{y_d}\right) \times \left(\times_{d|n} G_{z_d}\right)$. Furthermore, the isomorphisms are efficiently computable.*

Proof. It suffices to show for any $d \mid n$, $T_d(F_q) \cong G_{y_d} \times G_{z_d}$, and that this isomorphism is efficiently computable. Note that $y_d z_d = \gcd(\Phi_d(q), \frac{q^n-1}{U}) \gcd(\Phi_d(q), U) = \Phi_d(q)$ since $\gcd(U, \frac{q^n-1}{U}) = 1$ by the definition of U. By the same observation, $\gcd(y_d, z_d) = 1$. Lemma 1 establishes the claim.

The following is immediate from the previous 3 lemmas:

Lemma 5. *Assuming the maps of lemma 3 are efficient, there exist efficiently computable bijections γ and γ^{-1}, where $\gamma : F_{q^n}^* \rightarrow \times_{d|n} T_d(F_q)$.*

We now have the bijection claimed at the beginning:

Theorem 3. *Assuming the maps of lemma 3 are efficient, there exist efficiently computable bijections θ and θ^{-1}, where $\theta : T_n(F_q) \times \left(\times_{d|n,\ \mu(n/d)=-1} F_{q^d}^*\right) \rightarrow \times_{d|n,\ \mu(n/d)=1} F_{q^d}^*$.*

Proof. Lemma 5 gives efficient bijections between $T_n(F_q) \times \left(\times_{d|n,\ \mu(n/d)=-1} F_{q^d}^*\right)$ and $T_n(F_q) \times \left(\times_{d|n,\ \mu(n/d)=-1} \left(\times_{e|d} T_e(F_q)\right)\right)$, and also between $\times_{d|n,\ \mu(n/d)=1} F_{q^d}^*$ and $\times_{d|n,\ \mu(n/d)=1} \left(\times_{e|d} T_e(F_q)\right)$. By permuting coordinates, the theorem will follow if we show the multiset equality

$$\{n\} \cup \bigsqcup_{d|n,\ \mu(n/d)=-1} \{e \text{ s.t. } e \mid d\} = \bigsqcup_{d|n,\ \mu(n/d)=1} \{e \text{ s.t. } e \mid d\}.$$

From section 2, $\Phi_n(x) \prod_{\mu(n/d)=-1} (x^d - 1) = \prod_{\mu(n/d)=1} (x^d - 1)$ in the polynomial ring $\mathbb{Q}[x]$. Decomposing this equation into irreducible polynomials, we have $\Phi_n(x) \prod_{\mu(n/d)=-1} \prod_{e|d} \Phi_e(x) = \prod_{\mu(n/d)=1} \prod_{e|d} \Phi_e(x)$, and since $\mathbb{Q}[x]$ is a unique factorization domain, the irreducible polynomials on the left must be the same as those on the right. This gives the desired multiset equality.

4 Parameter Selection

The two constraints on choosing q and r for fixed n are security and efficiency constraints, the latter measured by the size $U(n,q)$ of the tables needed in our bijections. We first discuss the role of security in parameter selection:

4.1 Security measures

Our schemes derive their security from the same assumptions of XTR and CEILIDH. That is, if there is a successful attack against one of our cryptographic primitives, then there is a successful attack against the corresponding primitive in the underlying group we use, which we assume is impossible. Let $\langle g\rangle \subset F_{q^n}^*$ be a multiplicative group of order r with generator g. The security of our applications relies on the hardness of both the Computational Diffie-Hellman problem (CDH) and the Decisional Diffie-Hellman problem (DDH) in $\langle g\rangle$. The former is the problem of computing g^{xy} given g^x and g^y and the latter is that of distinguishing triples of the form (g^a, g^b, g^{ab}) from (g^a, g^b, g^c) for random a, b, and c. The hardness of both of these problems implies the hardness of the discrete logarithm problem (DL) in $\langle g\rangle$: find x given g^x. Due to the Pohlig-Hellman algorithm [21], the DL problem in $\langle g\rangle$ can be reduced to the DL problem in all prime order subgroups of $\langle g\rangle$, so we might as well assume that r is prime.

There are two known approaches to solving the DL problem in $\langle g\rangle$ [1, 7, 9, 13, 20, 27, 28], one which attacks the full multiplicative group of F_{q^n} itself using the Discrete Logarithm variant of the Number Field Sieve, and one which concentrates directly on the subgroup $\langle g\rangle$ using Pollard's Birthday Paradox based rho method [22]. Let s be the smallest divisor of n for which $\langle g\rangle$ can be embedded in $F_{q^s}^*$. The heuristic expected running time of the first attack is $L[q^s, 1/3, 1.923]$, where $L[n, v, u] = \exp((u+o(1))(\ln n)^v(\ln\ln n)^{1-v})$. If q is small, e.g. $q = 2$, then the constant 1.923 can be replaced with 1.53. The second attack, due to Pollard, takes $O(\sqrt{r})$ operations in $\langle g\rangle$.

Hence we see that the difficulty of solving the DL problem in $\langle g\rangle$ depends on both the size of the minimal surrounding subfield and on the size of its prime order r. If F_{q^n} is itself the minimal surrounding subfield, as is the case if we choose $r \mid \Phi_n(q)$ with $r > n$, then for sufficiently large r the DL, CDH, and DDH problems in $\langle g\rangle$ are widely believed to be just as hard as solving their classical counterparts w.r.t. an element of prime order $\approx r$ in the prime field of cardinality $\approx q^n$ [14]. As mentioned in [14], when $n \log_2 q \approx 1024$ and $\log_2 r \approx 160$, solving the DL problem in $\langle g\rangle$ is generally believed to be harder than factoring an 1024-bit RSA modulus provided q is not too small.

4.2 Practical algorithm for $n = 30$ and $n = 210$

Based on our security discussion, it is shown in [4] that, assuming an RSA key length between 1024 and 2048 bits gives adequate security, for $n = 30$ we should choose q to be a prime between 35 and 70 bits long, and for $n = 210$ we should choose q to be a prime between 5 and 10 bits long. Note that for the next value of n for which we achieve a communication savings, $n = 2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$, the field size will have to be at least 2310 bits, so any setting of q already exceeds the 2048 bits needed for adequate security.

In [13] it is shown how to quickly find a q and an r meeting these requirements for fixed n. The algorithm is heuristic, and involves choosing random q of a certain size and checking if $\Phi_n(q)$ contains a sufficiently large prime factor r by trial division with the primes up to roughly 10^5. On a 166MHz processor, for $n = 30$ it was shown that it takes 12 seconds to find an r of size between 214 and 251 bits for q of size 32 bits. Note that for $n = 30$ we actually need r to be slightly smaller, as claimed in the previous paragraph. This way we can achieve the largest efficiency gain for a fixed security guarantee. Using the algorithm of [13], fixing the size of r to be approximately 161 bits and searching for an appropriate q took three hours instead of the 12 seconds needed previously. However, there are three reasons we do not consider this to be problematic. First, CPU speeds are easily ten times as fast these days. Second, we don't need to fix the size of r to be exactly 161 bits; we just need to find an r of approximately this size. And third, finding the system parameters is a one-time cost and can be done offline, or even by a trusted third party.

From the efficiency analysis in the next section and lemma 6, one can show that the table size $U(n, q)$ resulting from choosing q at random subject to the above constraints is likely to be small with good probability. Hence, this heuristic algorithm is likely to find a q and an r so that both security and efficiency constraints are met in a reasonable amount of time.

4.3 Theoretical algorithm for general n with probabilistic guarantees

In this section we use properties of the density of primes to design a parameter selection algorithm and rigorously analyze its performance. Unfortunately, since the factorization of $\Phi_n(q)$ for random primes q does not seem to be well-understood, we are forced to choose $q > r$, which with respect to attacks known today, doesn't allow for choosing the optimal q and r for $n = 30$ and $n = 210$ if we just want 2048 bit RSA security. A straightforward calculation shows that for $n = 30$, the following algorithm gives us the largest efficiency gain for a fixed security guarantee if and only if q is at least 558 bits. Hence, we should view the algorithm as theoretical in nature, and apply the heuristic of the previous section for small n.

Let k be a positive integer tending to infinity and let n be the product of the first k primes. We want to choose q so that:

1. $n \log q$ is sufficiently large.
2. There exists a large prime factor r of $\Phi_n(q)$.
3. $U = U(n, q)$ is small.

We say an integer is *squarefree* if it contains no repeated factors. The selection algorithm is as follows:

Parameter Selection Algorithm PSA($n = p_1 \cdots p_k, Q, R$):

1. Let S be the subset of the first k primes p for which $p-1$ is squarefree, and put $T = \{p_1, \dots, p_k\} \setminus S$.
2. Find an R-bit prime r for which $r = 1 \bmod n$, and find a $z \in Z_r^*$ of order n.
3. Find a Q-bit prime $q = z + kr > n$, for some integer k, such that:
   (a) For all $p \in S$, $q^{pO_p(q)} \neq 1 \mod p^3$, where $O_p(q)$ denotes the order of q in Z_p^*.
   (b) For all $p \in T$, $O_p(q) = p - 1$.
4. Find a generator g of the subgroup of order r of $F_{q^n}^*$. Output r, q, and g.

We first claim that if the PSA algorithm terminates, then r and q meet the aforementioned properties. By setting Q large enough, the first property holds. We have $\Phi_n(q) = \Phi_n(z + kr) = \Phi_n(z) + sr$ for some integer s, and since $O_r(z) = n$, $\Phi_n(z) + sr = 0 \mod r$. Hence by choosing R sufficiently large, the second property holds. To show $U = U(n, q)$ is small, we need the following lemma:

Lemma 6. *Let p be a prime and q an integer such that $p \nmid q$. Then $p \mid U$ if and only if $pO_p(q) \mid n$. In case of the latter, $p^i \mid U$ if and only if $p^i \mid (q^{pO_p(q)} - 1)$.*

Proof. By minimality of U, $p \mid U$ if and only if there exist divisors $d < e$ of n for which $p \mid \gcd(\Phi_d(q), \Phi_e(q))$. Fix two such divisors d and e, let $f = \gcd(d, e)$, and suppose $f < d$. Since $f < d$, $p \mid \Phi_d(q) \mid (q^d - 1)/(q^f - 1) = 1 + q^f + q^{2f} + \cdots + q^{(d/f-1)f}$. Since $p \mid \gcd(\Phi_d(q), \Phi_e(q)) \mid \gcd(q^d - 1, q^e - 1) = q^f - 1$, we have $q^f = 1 \mod p$, so $d/f = 0 \mod p$, or $p \mid d/f$. Similarly, $p \mid e/f$. But then $p \mid \gcd(d/f, e/f)$, contradicting our choice of f. Hence, $d = f$ which means $d \mid e$ and $p \mid e/d \mid n$.

Suppose there is another divisor $c < d$ of n for which $p \mid \Phi_c(q)$. Then by the above, $c \mid d$ and $p \mid (d/c)$, and since $p \mid (e/d)$, $p^2 \mid e \mid n$, contradicting the fact that n is squarefree. This means that (d, e) is the unique pair of divisors for which $p \mid \gcd(\Phi_d(q), \Phi_e(q))$. Since $p \mid q^n - 1$, $O_p(q) \mid n$, and since $\gcd(O_p(q), p) = 1$, $pO_p(q) \mid n$. Put $d = O_p(q)$ and $e = pO_p(q)$. Then d is the smallest positive integer for which $q^d = 1$, so $p \mid \Phi_d(q)$. Also, $\Phi_e(q) = (q^e - 1)/(q^d - 1) = 1 + q^d + \cdots + q^{(e/d-1)d} = e/d \mod p = 0 \mod p$. Hence if $p \mid \gcd(\Phi_d(q), \Phi_e(q))$, then $d = O_p(q)$ and $e = pO_p(q)$. Conversely, if $pO_p(q) \mid n$, then $p \mid U$ for these d, e.

We have shown $p \mid U$ if and only if $pO_p(q) \mid n$. The above shows that if $p^i \mid U$, then $p^i \mid (\Phi_{O_p(q)}(q) \cdot \Phi_{pO_p(q)}(q)) \mid (q^{pO_p(q)} - 1)$, and conversely if $p^i \mid (q^{pO_p(q)} - 1) \mid (q^n - 1)$, then $p^i \mid \gcd(\Phi_d(q), \Phi_e(q)) \mid U$.

Remark 1. Note that $p^2 \mid (q^{pO_p(q)} - 1)$, since on the one hand we have $p \mid (q^{O_p(q)} - 1)$, and on the other hand we have $(q^n - 1)/(q^{O_p(q)} - 1) = 1 + q^{O_p(q)} + q^{2O_p(q)} + \cdots + q^{(p-1)O_p(q)} = 1 + 1 + \cdots + 1 = 0 \mod p$. Hence if $p \mid U$, then $(q^{pO_p(q)} - 1) \mid (q^n - 1)$, so it follows that $p^2 \mid U$.

The following lemma provides tight asymptotic bounds on $U = U(n, q)$:

Lemma 7. *If the PSA algorithm terminates, $U = \Theta(n^{2C})$, where $C \approx .374$ is Artin's constant.*

Proof. By the previous lemma, if $p \mid U$, then $p \mid n$, so $p \in \{p_1, \ldots, p_k\}$. Now if $p \in T$, $p-1$ is not squarefree, so $O_p(q) \not| \, n$ by step 3b, so $p \not| \, U$. On the other hand, if $p \in S$, $p-1$ is a product of distinct primes in $\{p_1, \ldots, p_k\}$, so $O_p(q) \mid n$ and hence $p \mid U$. Combining this with the remark above, step 3a of the PSA algorithm, and the previous lemma, we conclude that U is exactly the square of the product of primes in S and that the PSA algorithm chooses q so that U is minimal.

To obtain the bound on U it suffices to show that the density of primes p for which $p-1$ is squarefree is C, where C is Artin's constant [8]. The bound will then hold for large enough k. For a prime p, $p-1$ is not squarefree if and only if $p = 1 \mod q^2$ for a prime q. By the inclusion-exclusion principle, the multiplicativity of $\phi(\cdot)$, and theorem 2, the density of primes p for which $p-1$ is squarefree is:

$$1 - \sum_{\text{primes } p} \frac{1}{\phi(p^2)} + \sum_{\text{primes } p,q} \frac{1}{\phi(p^2 q^2)} - \cdots = \prod_{\text{primes } p} \left(1 - \frac{1}{\phi(p^2)}\right) = C.$$

By theorem 1, for sufficiently large k, $p_k \approx k \log k$ and $k \approx \frac{\log k}{\log \log k}$, where the approximation is up to low order terms. Hence, $U \le p_k^{2Ck} \approx (k \log k)^{2Ck} \approx (\log n)^{2C \frac{\log n}{\log \log n}} \approx n^{2C}$.

Finally, we show the PSA algorithm terminates quickly in expectation:

Efficiency Analysis: By theorem 1, $k \approx \frac{\log n}{\log \log n}$ and $p_k \approx \log n$. Determining S and T in step 1 can therefore be done by trial division in $O(\log^2 n)$ time. We can perform step 2 by choosing a random R-bit number r, efficiently checking if r is prime, and checking if $r = 1 \bmod n$. This requires an expected $\phi(n)R = O\left(\frac{Rn}{\log \log n}\right)$ samples r. To find z, we choose a random $\alpha \in Z_r^*$, set $\beta = \alpha^{\frac{q-1}{n}}$, and check that $\beta^d \neq 1 \mod r$ for all proper divisors d of n. In expectation, after $O(\log R)$ trials one such α will be a generator of Z_r^*, for which setting $z = \beta = \alpha^{\frac{q-1}{n}}$ gives z with $O_r(z) = n$. Conversely, if for all proper divisors d of n we have $\beta^d \neq 1 \bmod r$, then $O_r(\beta) = n$. Since the number of proper divisors of n is $O(n^\epsilon)$ for any $\epsilon > 0$ ([11], section 18.1), the check in step 2 is efficient.

For step 3, for each $p \in T$, we can find an element $a_p \in Z_p^*$ with $O_p(a_p) = p-1$ by simply trying each of the $p - 1 = O(\log n)$ elements of Z_p^* until we succeed. We then choose a random integer k for which $q = z + kr$ is a Q-bit number and efficiently check if q is prime. If so, then for each $p \in S$, we can compute $O_p(q)$ in $O(\log n)$ time, then check if $q^{pO_p(q)} \neq 1 \bmod p^3$ by repeated squaring. For each

$p \in T$ we check if $q = a_p \bmod p$.

The claim is that the number of random samples k needed in step 3 is only $O(Qn^{1-C})$. Using the fact that the density of primes amongst integers of the form $z + kr$ is $O\left(\frac{1}{\log(z+kr)}\right)$, an integer k for which $z + kr$ is prime can be found with $O(Q)$ samples in expectation. By independence, the density of primes q which are $a_p \bmod p$ for every $p \in T$ is $\prod_{p \in T} \frac{1}{\phi(p)} = \Omega\left(\frac{\log \log n}{n^{1-C}}\right)$, where C is Artin's constant. Fix any $p \in S$. By theorem 2, for all but a negligible fraction of primes q, $q = g^i \mod p^3$ for g a generator of $Z^*_{p^3}$. Since g is a generator, $q^{pO_p(q)} = 1 \bmod p^3$ if and only if i is a multiple of $\frac{\phi(p^3)}{pO_p(q)}$, and there are only $pO_p(q) \leq p(p-1)$ such multiples. By theorem 2, it is equally likely that $q = g^i$ for any $i \in [\phi(p^3)]$, so the density of primes q for which $q^{pO_p(q)} \neq 1 \bmod p^3$ is at least $1 - 1/p$. By independence, the density of q for which $q^{pO_p(q)} \neq 1 \bmod p^3$ for all $p \in S$ is at least $\prod_{p \in S}(1 - 1/p) = \prod_{p \in S} = \Omega\left(\frac{1}{\log \log n}\right)$. Applying independence one last time, we conclude that q can be found with an expected $O(Qn^{1-C})$ samples k.

Finally, step 4 can be implemented by choosing a random $g \in F^*_{q^n}$ and making sure that $(q^n - 1)/r \neq 1$. The number of generators of $F^*_{q^n}$ is $\phi(q^n - 1)$ which is $\Omega\left(\frac{q^n}{\log n + \log Q}\right)$, so the expected number of samples g needed is $O(\log n + \log Q)$.

5 Cryptographic Applications

Let n be the product of the first k primes, and let r, q, and g be public parameters generated as in section 4. Define $\sigma^-(n) = \sum_{d|n,\ \mu(n/d)=-1} d$ and $\sigma^+(n) = \sum_{d|n,\ \mu(n/d)=1} d$, and observe that $\phi(n) + \sigma^-(n) = \sigma^+(n)$. From section 3, we have an efficiently computable bijection θ and its inverse θ^{-1}, with $\theta : T_n(F_q) \times \left(\times_{d|n,\ \mu(n/d)=-1} F^*_{q^d}\right) \rightarrow \times_{d|n,\ \mu(n/d)=1} F^*_{q^d}$.

From the proof of theorem 3, we see that there are a number of choices for θ depending on which coordinate permutation is chosen. While this choice does not affect the communication of our protocols or the size of our encryptions/signatures, it can affect the computational costs. In section 6 we choose a specific permutation and analyze the computational requirements for $n = 30$.

We will think of θ and θ^{-1} as efficiently computatble maps between $T_n(F_q) \times F_q^{\sigma^-(n)}$ and $F_q^{\sigma^+(n)}$ by fixing polynomial representations of F_{q^d} with $d \mid n$. An element of $F_q^{\sigma^-(n)}$ is then just a list of $\sigma^-(n)$ q-ary coefficients with respect to these polynomials, and can be treated as an element of $\times_{d|n,\ \mu(n/d)=-1} F^*_{q^d}$. Let $i_d, i_d + 1, \ldots, i_d + d - 1$ denote the coordinates of an element $x \in F_q^{\sigma^-(n)}$ corresponding to the coefficients of x with respect to the irreducible polynomial for F_{q^d}. Our map may not be well-defined because we may have $(x_{i_d}, x_{i_d+1}, \ldots, x_{i_d+d-1}) = 0$. However, if $y \in F_q^{\sigma^-(n)}$ is chosen randomly, the probability that some coordinate of y is zero is less than $\sigma^-(n)/q = O(n^\epsilon/q)$ for any $\epsilon > 0$, which is negligible. The same is true of a randomly chosen element of $F_q^{\sigma^+(n)}$. Hence, if we apply θ and θ^{-1} to random $(x_1, x_2) \in T_n(F_q) \times F_q^{\sigma^-(n)}$ and $y \in F_q^{\sigma^+(n)}$, $\theta(x_1, x_2)$ and $\theta^{-1}(y)$ are well-defined with overwhelming probability.

It is possible to modify θ and θ^{-1} if one wants more than a probabilistic guarantee. Define $d^-(n) = \sum_{d|n,\ \mu(n/d)=-1} 1$ and $d^+(n) = \sum_{d|n,\ \mu(n/d)=1} 1$. We can efficiently extend θ to the well-defined map $\tilde{\theta}$,

$$\tilde{\theta} : T_n(F_q) \times F_q^{\sigma^-(n)} \rightarrow \left(\times_{d|n,\ \mu(n/d)=1} F_{q^d}^*\right) \times \{0,1\}^{d^-(n)},$$

where for each $(x, y) \in T_n(F_q) \times F_q^{\sigma^-(n)}$ and for each $d \mid n$ with $\mu(n/d) = -1$, if $(y_{i_d}, \ldots, y_{i_d+d-1}) = 0$, we replace y_{i_d+d-1} with 1, obtaining a new string y', and define $\tilde{\theta}(x, y) = \theta(x, y') \circ b$, where for all $j \in [d^-(n)]$, $b_j = 1$ if and only if $(y_{i_d}, \ldots, y_{i_d+d-1}) = 0$ for the jth divisor d. Note that $\tilde{\theta}^{\leftarrow}$, the inverse of $\tilde{\theta}$ restricted to the image of $\tilde{\theta}$, is also well-defined. Similarly, letting β denote θ^{-1}, we can extend β to a well-defined map $\tilde{\beta} : F_q^{\sigma^+(n)} \rightarrow T_n(F_q) \times F_q^{\sigma^-(n)} \times \{0,1\}^{d^+(n)}$ and construct $\tilde{\beta}^{\leftarrow}$.

The next sections describe our cryptographic applications. For simplicity, in our security analyses we assume θ and θ^{-1} are actually bijections between $T_n(F_q) \times F_q^{\sigma^-(n)}$ and $F_q^{\sigma^+(n)}$, although it should be understood that our protocols can be slightly modified so that $\tilde{\theta}$ or β can be used without affecting the security. The only application where this is not immediately obvious is the non-hybrid ElGamal encryption, but step 3 of that protocol can be modified to additionally encrypt the "extra bits" from $\tilde{\beta}$ using, say, the same key used in step 3.

5.1 Diffie-Hellman Key Agreement

For Alice and Bob to agree on a sequence of m secret keys K_i, they engage in the following protocol:

1. Alice and Bob choose random S_0 and T_0 in $\times_{d|n,\ \mu(n/d)=-1} F_{q^d}^*$, respectively, and treat them as elements of $F_q^{\sigma^-(n)}$.
2. For $i = 1$ to m,
   (a) Alice selects a random integer x_i with $1 \le x_i \le r$, sets $A_i = g^{x_i}$, computes $\theta(A_i, S_{i-1}) = (a_i, S_i) \in F_q^{\phi(n)} \times F_q^{\sigma^-(n)}$ and transmits a_i to Bob.
   (b) Bob selects a random integer y_i with $1 \le y_i \le r$, sets $B_i = g^{y_i}$, computes $\theta(B_i, T_{i-1}) = (b_i, T_i) \in F_q^{\phi(n)} \times F_q^{\sigma^-(n)}$ and transmits b_i to Alice.
3. Alice sends S_m to Bob and Bob sends T_m to Alice.
4. For $i = m$ to 1,
   (a) Alice computes $\theta^{-1}(b_i, T_i) = (B_i, T_{i-1})$, and sets $K_i = B_i^{x_i} = g^{x_i y_i}$.
   (b) Bob computes $\theta^{-1}(a_i, S_i) = (A_i, S_{i-1})$, and sets $K_i = A_i^{y_i} = g^{x_i y_i}$.

The number of bits sent from Alice to Bob (and from Bob to Alice) is about $(m\phi(n)+\sigma^-(n))\log q$, so the rate approaches the optimal $\phi(n)\log q$ bits per key as m gets large. This beats all known schemes for $n \geq 30$. In particular, for $n = 30$, our scheme requires only $8\log q$ bits per shared key while generalizing the scheme in section 4.11 of [14] to $n = 30$ gives a scheme requiring $10\log q$ bits per key exchange. The scheme in [24] would also achieve our rate, but needs an unproven conjecture concerning the rationality of $T_{30}(F_q)$.

Observe that (A_1, S_0) and (B_1, T_0) are random, and since θ is a bijection, the last $\sigma^-(n)$ coordinates of $\theta(A_1, S_0)$ are of a random element in $\times_{d|n,\ \mu(n/d)=1} F_{q^d}^*$. Hence the probability that some coordinate of S_1 is zero is even less than that for a random element in $F_q^{\sigma^+(d)}$, which is negligible. One can then verify that every application of θ or θ^{-1} is on a random element. It follows from the foregoing discussion and the union bound that the probability of either Alice or Bob ever attempting to apply θ or θ^{-1} on an element outside of the domain is negligible. For deterministic guarantees, one can replace θ and θ^{-1} with $\tilde{\theta}$ and $\tilde{\theta}^{\leftarrow}$, negligibly changing the rate to $\phi(n)\log q + O(n^\epsilon)$ for any $\epsilon > 0$. Given the overwhelming probability guarantees for θ and θ^{-1}, this does not seem necessary.

Security: An eavesdropper obtains $a_1, \ldots, a_m, b_1, \ldots, b_m, S_m$, and T_m. Since θ and θ^{-1} are efficient bijections, this is equivalent to obtaining $A_1, \ldots, A_m$, $B_1, \ldots, B_m, S_0$, and T_0. Since S_0 and T_0 are random, determining a shared secret K_i is equivalent to solving the CDH problem in $\langle g \rangle$, given $A_1, \ldots, A_m, B_1, \ldots, B_m$.

5.2 ElGamal Signature Schemes

Suppose the message M to be signed is at least $\sigma^-(n)\log q - \log r$ bits long. If this is not the case, one can wait until there are $m > 1$ messages M_i to be signed for which $\sum_i |M_i| \geq \sigma^-(n)\log q - \log r$, then define M to be the concatenation $M_1 \circ \cdots \circ M_m$ and sign M. For a random a, $1 \leq a \leq r-1$, let a be Alice's private key and $A = g^a$ her public key. Let $h : \{0,1\}^* \to \mathbb{Z}_r$ be a cryptographic hash function. We have the following generalized ElGamal signature scheme (see p.458 of [18] for background):

Signature Generation (M):

1. Alice selects a random secret integer k, $1 \leq k \leq r$, and computes $d = g^k$.
2. Alice then computes $e = k^{-1}(h(M) - ah(d)) \bmod r$.
3. Alice expresses $M \circ e$ as $(R, S) \in F_q^{\sigma^-(n)} \times \{0,1\}^*$, computes $\theta(d, R) = T$, and outputs (S, T) as her signature.

Signature Verification (M, S, T):

1. Bob computes $\theta^{-1}(T) = (d, R)$ and constructs M and e from R and S.
2. Bob accepts the signature if and only if $A^{h(d)}d^e = g^{h(M)}$.

The communication of this scheme is at the optimal $|M| + \log r + \phi(n)\log q$ for ElGamal signature schemes, even for one message (as long as M is large enough).

This beats the $|M| + \log r + (n/3)\log q$ communication of the scheme in [4, 17] when $n \geq 30$, in particular for the practical values $n = 30$ and $n = 210$. Our communication is the same as that in [24], but we do not rely on any conjectures.

Note that our map θ may fail since M need not be random. One can avoid this by excluding the negligibly few M for which θ is not defined (as in RSA or the schemes of [24]), or one can replace θ with $\tilde{\theta}$, as defined above, and communicate an additional $O(n^\epsilon)$ bits of overhead. Alternatively Alice can use a pseudorandom generator to randomize M and communicate the small seed used to Bob, requiring even less communication than the already asymptotically negligible $O(n^\epsilon)$ bits.

We note that a simple modification of our protocol, making it similar in spirit to our key exchange protocol, can allow Alice to sign each M_i individually, allowing for incremental verification.

Security: In this scheme the verifier obtains (S, T), which is equivalent to obtaining M, d, and e. Thus, the security of this scheme reduces to the security of the generalized ElGamal signature scheme in $\langle g \rangle$.

5.3 ElGamal Encryption

We present two flavors of ElGamal encryption. The first is a hybrid scheme with shorter encryptions than the one in [14], while the second is essentially a non-hybrid analogue of ElGamal in $T_n(F_q)$. In the second, to encrypt a sequence of m messages, $m+1$ encryptions are created and m of them are performed directly in $T_n(F_q)$. The first scheme achieves optimal communication, while the second is asymptotically optimal.

Hybrid ElGamal For random b, $1 \leq b \leq r-1$, let b be Bob's private key and $B = g^b$ his public key. Suppose Alice wants to encrypt the message $M \in F_q^{\sigma^-(n)}$ with Bob's public key. Let E be an agreed upon symmetric encryption scheme with domain $F_q^{\sigma^-(n)}$. We have the following protocol:

Encryption (M):

1. Alice selects a random secret integer k, $1 \leq k \leq r$, and computes $d = g^k$.
2. From B Alice computes $e = B^k = g^{bk}$.
3. From e Alice derives a key Q for E and computes the encryption of M, $E(M)$, under key Q. Alice writes $E(M)$ as $(R, S) \in F_q^{\sigma^-(n)} \times \{0,1\}^*$.
4. Alice computes $\theta(d, R) = T$ and outputs her encryption (S, T).

Decryption (S, T):

1. Bob computes $\theta^{-1}(T) = (d, R)$.
2. From d and b Bob computes $e = g^{bk}$.
3. From e Bob derives Q and decrypts $E(M) = (R, S)$ to obtain and output M.

The communication of this scheme is at the optimal $|E(M)| + \phi(n)\log q$ bits for hybrid ElGamal encryption. As in our protocol for signature schemes, we achieve this rate even for a single message. This beats the $|E(M)| + (n/3)\log q$ bit scheme in [14] for $n \geq 30$.

It is unlikely that θ or θ^{-1} is applied to an element with any zero coordinates since d is random and $E(M)$ is likely to "look random" in practice, so $\theta(d, R)$ is likely to be a random element of $F_q^{\sigma^+(n)}$ for which it is extremely unlikely that any coordinates are zero. An exact analysis, though, depends on one's choice of E. As in our protocol for signature schemes, one can randomize $E(M)$ to decrease the error probability or replace θ with $\tilde{\theta}$ for a deterministic guarantee at the cost of a few bits of communication.

Security: An adversary learns (S, T), which is equivalent to learning d and $E(M)$. Assuming the CDH problem is hard in $\langle g \rangle$, the security of this scheme is just that of the symmetric scheme E, assuming the key Q to E is chosen reasonably from e. To derive Q from e, one can extract bits that are hard to compute by an eavesdropper, see [2].

Almost Non-Hybrid ElGamal In the following, Alice will encrypt a sequence of m messages $M_1, \ldots, M_m$, each in $F_q^{\phi(n)}$. She will form $m+1$ encryptions, m of which are encryptions in $T_n(F_q)$, and one requiring the use of an agreed upon symmetric encryption scheme E.

In the encryption phase of our scheme we will apply θ^{-1} to $(M_i \circ R)$ for some $R \in F_q^{\sigma^-(n)}$. For semantic security, for all i it must hold that $\theta^{-1}(M_i \circ R) \in \langle g \rangle \times F_q^{\sigma^-(n)}$, which in general may be strictly contained in $T_n(F_q) \times F_q^{\sigma^-(n)}$. For this we adopt the technique in section 3.7 of [25]. Namely, by reserving a few bits of each M_i to be "redundancy bits", if $\langle g \rangle$ has small enough index in $T_n(q)$, then for any R we need only try a few random settings of these bits until $\theta^{-1}(M_i \circ R) \in \langle g \rangle \times F_q^{\sigma^-(n)} = (c, d) \in \langle g \rangle \times F_q^{\sigma^-(n)}$, which we can test by checking if $c^r = 1$. In the following protocol description we ignore this issue and assume whenever θ^{-1} is applied, its image is in $\langle g \rangle \times F_q^{\sigma^-(n)}$.

For random b_1, b_2, $1 \leq b_1, b_2 \leq r-1$, let b_1, b_2 be Bob's private keys and $B_1 = g^{b_1}, B_2 = g^{b_2}$ be his public keys. We have the following scheme:

Encryption (M):

1. Alice chooses a random $R_0 \in F_q^{\sigma^-(n)}$.
2. For $i = 1$ to m,
   (a) Alice computes $\theta^{-1}(M_i \circ R_{i-1}) = (c_i, R_i) \in \langle g \rangle \times F_q^{\sigma^-(n)}$.
   (b) Alice chooses a random secret integer $k_i, 1 \leq k_i \leq r$, and forms the encryption $(d_i, e_i) = (g^{k_i}, c_i B_1^{k_i})$.
3. Alice uses the hybrid ElGamal encryption scheme with symmetric cipher E and public key B_2 to encrypt R_m as (T_m, S) with $T_m \in F_q^{\sigma^-(n)}$ and $S \in \{0,1\}^*$.

4. For $i = m$ to 1,
   (a) Alice computes $\theta(d_i, T_i) = (x_i, W_i) \in F_q^{\phi(n)} \times F_q^{\sigma^-(n)}$.
   (b) Alice computes $\theta(e_i, W_i) = (y_i, T_{i-1}) \in F_q^{\phi(n)} \times F_q^{\sigma^-(n)}$.
5. Alice outputs $x_1, \ldots, x_m, y_1, \ldots, y_m, T_0, S$ as her encryption of $M_1, \ldots, M_m$.

Decryption $(x_1, \ldots, x_m, y_1, \ldots, y_m, T_0, S)$:

1. For $i = 1$ to m,
   (a) Bob computes $\theta^{-1}(y_i \circ T_{i-1}) = (e_i, W_i)$.
   (b) Bob computes $\theta^{-1}(x_i \circ W_i) = (d_i, T_i)$.
   (c) Bob computes $c_i = e_i / d_i^{b_1}$.
2. Bob uses T_m and S, together with b_2, in the decryption procedure of the hybrid ElGamal scheme to recover R_m.
3. For $i = m$ to 1, Bob computes $\theta(c_i, R_i) = M_i \circ R_{i-1}$.
4. Bob outputs $M_1, \ldots, M_m$.

The communication of this scheme is $2m\phi(n) \log q + |E(R_m)| + \phi(n) \log q$ bits. Hence, as m grows, the rate of this scheme approaches $2\phi(n) \log q$, which is optimal for ElGamal type encryption.

Note that the M_i's need not be random, and consequently $\theta^{-1}(M_i, R_{i-1})$ may not be well-defined. Choosing random R_0 will increase the chances that $\theta^{-1}(M_i, R_{i-1})$ is always defined. Alternatively, one can use the ideas of section 5.2 to randomize M_i, or one can use β instead of θ^{-1}. Again, since $E(R_m) = (S, T_m)$ needn't be random even if E is semantically secure, one may want to use $\tilde{\theta}$ in place of θ. This adds a negligible amount to the communication, and as stated earlier, encrypting the extra bits of $\tilde{\beta}$ can be done in step 3.

Security: An adversary learns $x_1, \ldots, x_m, y_1, \ldots, y_m, T_0, S$, which is equivalent to learning $E'(R_m), d_1, \ldots, d_m, e_1, \ldots, e_m$, where E' is the semantically secure hybrid encryption scheme. Assuming DDH is hard in $\langle g \rangle$, (d_i, e_i) is a semantically secure encryption $E''(c_i)$ of c_i for all i. The security of the scheme then follows from the fact that the keypairs (b_1, B_1) and (b_2, B_2) of E', E'' are independent.

6 Computational Complexity

In this section we present efficient algorithms for computing θ and θ^{-1}, analyze their complexity, and suggest an alternative way of improving computational costs with slightly more communication. Each of these is described in turn.

6.1 Algorithm

Before describing θ and θ^{-1}, we need some notation:

– For $d \mid n$, let U_d be the smallest integer for which $\gcd(\Phi_e(q), \Phi_f(q), \frac{q^d-1}{U_d}) = 1$ for all $e \neq f$ with $e \mid d$ and $f \mid d$.

- For $e \mid d \mid n$, we define $y_{d,e} = \gcd(\Phi_e(q), \frac{q^d-1}{U_d})$ and $z_{d,e} = \gcd(\Phi_e(q), U_d)$. Generalizing section 3, we can find w_d and $w_{d,e}$ s.t. $\frac{q^d-1}{U_d} w_d + \sum_{e|d} \frac{q^d-1}{y_{d,e}} w_{d,e} = 1$. Further, we can find $u_{d,e}$ and $v_{d,e}$ for which $\frac{\Phi_e(q)}{y_{d,e}} u_{d,e} + \frac{\Phi_e(q)}{z_{d,e}} v_{d,e} = 1$.
- Let $\rho_e(d) : \{d : e \mid d \mid n, \mu(n/d) = -1\} \rightarrow \{d : e \mid d \mid n, \mu(n/d) = +1\}$ for $e \mid n$, $e \neq n$, be a bijective mapping and define $\rho_n(n) = n$.

A naive implementation of θ consists of the following steps:

1. We first use an isomorphism
$$T_n(F_q) \times \times_{\mu(n/d)=-1} F_{q^d} \longrightarrow T_n(F_q) \times \times_{\mu(n/d)=-1} G_{U_d} \times G_{(q^d-1)/U_d}.$$
2. By using a table lookup we map $\times_{\mu(n/d)=-1} G_{U_d} \longrightarrow \times_{\mu(n/d)=-1} \times_{e|d} G_{z_{d,e}}$ and we use an isomorphism $\times_{\mu(n/d)=-1} G_{(q^d-1)/U_d} \longrightarrow \times_{\mu(n/d)=-1} \times_{e|d} G_{y_{d,e}}$. By the structure theorem of Abelian groups there is an isomorphism $G_{z_{d,e}} \times G_{y_{d,e}} \longrightarrow T_e(F_q)$ for each $d|n$ with $\mu(n/d) = -1$ and $e \mid d$.
3. By using a permutation we obtain a mapping
$$T_n(F_q) \times \times_{\mu(n/d)=-1} \times_{e|d} T_e(F_q) \longrightarrow \times_{\mu(n/d)=+1} \times_{e|d} T_e(F_q).$$
4. By the structure theorem of Abelian groups there is, for each $d|n$ with $\mu(n/d) = +1$ and $e \mid d$, an isomorphism $T_e(F_q) \longrightarrow G_{z_{d,e}} \times G_{y_{d,e}}$. By using a table lookup we map $\times_{\mu(n/d)=+1} \times_{e|d} G_{z_{d,e}} \longrightarrow \times_{\mu(n/d)=+1} G_{U_d}$ and we use an isomorphism $\times_{\mu(n/d)=+1} \times_{e|d} G_{y_{d,e}} \longrightarrow \times_{\mu(n/d)=+1} G_{(q^d-1)/U_d}$.
5. In the last step we use an isomorphism
$$\times_{\mu(n/d)=+1} G_{U_d} \times G_{(q^d-1)/U_d} \longrightarrow \times_{\mu(n/d)=+1} F_{q^d}.$$

Each of the isomorphisms are defined by taking simultaneous exponentiations. An improved implementation combines different isomorphisms in a single simultaneous exponentiation. Each table lookup followed by an exponentiation can be implemented as a single table lookup. This reduces the number of exponentiations and multiplications.

Computation of $\theta(x, (x_d)_{d|n,\mu(n/d)=-1})$ for $(x_d)_{d|n,\mu(n/d)=-1} \in \times_{d|n,\mu(n/d)=-1} F^*_{q^d}$ and $x \in T_n(F_q)$:

1. For $d \mid n$, $\mu(n/d) = -1$,
   (a) Compute $x_d^{(q^d-1)/U_d} \in G_{U_d}$ and map it to $(Z_{d,e})_{e|d} \in \times_{e|d} G_{z_{d,e}}$ by using a table look up.
   (b) Compute $(Z_{\rho_e(d),e} = (Z_{d,e}^{v_{d,e}} x_d^{(q^d-1)u_{d,e}/y_{d,e}})^{\Phi_e(q)/z_{\rho_e(d),e}})_{e|d} \in \times_{e|d} G_{z_{\rho_e(d),e}}$.
2. Compute $Z_{n,n} = x^{\Phi_n(q)/z_{n,n}} \in G_{z_{\rho_n(n),n}}$.
3. For $d \mid n$, $\mu(n/d) = +1$,
   (a) Map $(Z_{d,e})_{\rho_e(d')=d,e|d} \in \times_{e|d} G_{z_{d,e}}$ to $Z_d \in G_{U_d}$ by using a table look up.
   (b) Compute $x_d = Z_d^{w_d} \cdot \prod_{\rho_e(d')=d,e|d,e\neq n} (Z_{d',e}^{v_{d',e}} x_{d'}^{(q^{d'}-1)u_{d',e}/y_{d',e}})^{\Phi_e(q) w_{d,e}/y_{d,e}}$ which is in $G_{U_d} \cdot G_{(q^d-1)/U_d} = F^*_{q^d}$.

4. Multiply x_n with $x^{\Phi_n(q)w_{n,n}/y_{n,n}}$.
5. $\theta(x,(x_d)_{d|n,\mu(n/d)=-1}) = (x_d)_{d|n,\mu(n/d)=+1}$.

The ideas in section 3 can be used to show the algorithm above is well-defined. The improved computation of θ^{-1} is similar, where we make sure to use the inverse of the coordinate permutation used in θ.

6.2 Complexity

For background on efficient computations in fields and subgroups, see [6, 12, 29]. Consider the algorithm for θ. In step 1, for $d \mid n$, $\mu(n/d) = -1$, we perform $1 + \sum_{e|d} 1$ exponentiations in F_{q^d}. Notice that, in step 1b we do not need to compute $Z_{d,e}^{v_{d,e}}$ since it can be combined with the table lookup in step 1a (there is an entry in the table corresponding to $Z_{d,e}^{v}$ for every v). Step 2 costs 1 exponentiation in F_{q^n}.

For $d \mid n$, $\mu(n/d) = -1$ or $d = n$, we precompute $x_d^{2^i}$, $0 \le i \le d \log q$. This costs $d \log q$ multiplications in F_{q^d}. By using the results of the precomputation, an exponentiation x_d^t, for some t, in F_{q^d} costs on average $(d \log q)/2$ multiplications in F_{q^d} (the bit length of the exponent t is $(d \log q)$ and roughly half the time a bit is equal to 1). Each multiplication in F_{q^d} costs $f(d) \le d^2$ multiplications in F_q. Summarizing, steps 1 and 2 cost about

$$C_1 = \left(3f(n)n + \sum_{d|n,\mu(n/d)=-1} (3 + \sum_{e|d} 1) f(d) d \right) \frac{\log q}{2}$$

multiplications in F_q.

In step 3, for $d \mid n$, $\mu(n/d) = +1$, we need to perform, for each $e \mid d$ with $\rho_e(d') = d$, one exponentiation in $F_{q^{d'}}$. We do not need to compute $Z_{d',e}^{v_{d',e}}$ which can be combined with the table lookup in step 1a.

The cost of step 3, measured in multiplications in the base field F_q, is on average approximately $\sum_{d|n,\mu(n/d)=+1} \sum_{e|d} f(\rho_e^{-1}(d)) \rho_e^{-1}(d) (\log q)/2$. Since ρ_e defines a permutation, this expression is equal to

$$C_2 = \left(f(n)n + \sum_{d|n,\mu(n/d)=-1} (\sum_{e|d} 1) f(d) d \right) \frac{\log q}{2}.$$

The total cost is $C_1 + C_2$ multiplications in F_q, where we neglect the cost of table lookups, addition, and multiplication modulo an integer. Since $\sum_{e|d} 1 = O(d^{\varepsilon})$, we have $\sum_{d|n,\mu(n/d)=-1} (3 + 2\sum_{e|d} 1) f(d) d = O(\sum_{d|n} d^{3+\varepsilon}) = O((\sum_{d|n} d)^{3+\varepsilon}) = O(n^{3+\varepsilon})$, since the sum of divisors of n is $O(n^{1+\epsilon})$ for any $\epsilon > 0$. This proves $C_1 + C_2 = O(n^{3+\varepsilon} \log q)$.

The same techniques show θ^{-1} requires $O(n^{3+\epsilon} \log q)$ multiplications in F_q.

6.3 Efficiency Improvements

To improve the efficiency we may use exponentiation algorithms for fixed exponents using vector addition chains. Also, we may group several exponentiations of x_d together into one exponentiation by appropriately choosing the bijections ρ_e. If n is not too large, we may use simultaneous exponentiation to speed up the computations. Full simultaneous exponentiations in every step requires a precomputation of 2^n multiplications. We may optimize by using simultaneous exponentiation to compute intermediate results which we multiply together to compute the full exponentiation. Finally, we may combine the exponentiations required in our applications with the evaluation of θ.

Notice that θ is much more efficient if, for $d \mid n$ with $\mu(n/d) = -1$, $x_d \in G_{(q^d-1)/U_d}$. Then, for $e \mid d \mid n$ with $\mu(n/d) = -1$, $Z_{d,e} = 1$ and $Z_d = 1$. Table lookups can be avoided. Therefore each x_d, for $d \mid n$ with $\mu(n/d) = +1$, can be computed by a single simultaneous exponentiation of $x, x_d \in G_{(q^d-1)/U_d}, d \mid n, \mu(n/d) = -1$, with fixed exponents in step 3. To make use of this, we define a new map τ which maps $(x, (x_d)_{d|n,\mu(n/d)=-1})$ into $\theta(x, (x_d^{U_d})_{d|n,\mu(n/d)=-1})$ and the table entries of $(x_d^{(q^d-1)/U_d})_{d|n,\mu(n/d)=-1}$. This increases the communication cost by

$$\sum_{d|n,\mu(n/d)=-1} \log_2 U_d$$

bits which in practice is much less than $\log_2 q$. So at the cost of a small increase in communication we improve the computational efficiency.

Computation of $\tau(x, (x_d)_{d|n,\mu(n/d)=-1})$ and $\tau^{\leftarrow}$:

1. For $d \mid n$, $\mu(n/d) = -1$, compute $(x'_d = x_d^{(q^d-1)/U_d})_{d|n,\mu(n/d)=-1}$.
2. Compute

$$x_d = \prod_{\rho_e(d')=d, e|d, e\neq n} (x_{d'}^{U_{d'}(q^{d'}-1)u_{d',e}/y_{d',e}})^{\Phi_e(q)w_{d,e}/y_{d,e}} \in G_{(q^d-1)/U_d} \subseteq F^*_{q^d},$$

   for $d \mid n$ with $\mu(n/d) = +1$. Multiply x_n with $x^{\Phi_n(q)w_n/z_{n,n} + \Phi_n(q)w_{n,n}/y_{n,n}}$.
3. $\tau(x, (x_d)_{d|n,\mu(n/d)=-1}) = ((x_d)_{d|n,\mu(n/d)=+1}, (x'_d)_{d|n,\mu(n/d)=-1})$.
4. Compute $x_n^{(q^n-1)v_{n,n}/U_n + (q^n-1)u_{n,n}/y_{n,n}} = x$.
5. Compute

$$x'^{a_{d'}}_{d'} \Big(\prod_{d=\rho_{e'}(d'), e'|d'} (x_d^{(q^d-1)u_{d,e'}/y_{d,e'}})^{\Phi_{e'}(q)w_{d',e'}/y_{d',e'}} \Big)^{b_{d'}} = x'^{a_{d'}}_{d'} x_{d'}^{U_{d'}b_{d'}} = x_{d'},$$

   for $d' \mid n$ with $\mu(n/d') = -1$, where $\frac{q^{d'}-1}{U_{d'}} a_{d'} + U_{d'} b_{d'} = 1$.
6. $\tau^{\leftarrow}((x_d)_{d|n,\mu(n/d)=+1}, (x'_d)_{d|n,\mu(n/d)=-1}) = (x, (x_d)_{d|n,\mu(n/d)=-1})$.

For $n = 30$, $\{d \mid n : \mu(n/d) = -1\} = \{15, 10, 6, 1\}$ and $\{d \mid n : \mu(n/d) = +1\} = \{30, 5, 3, 2\}$. We define $\rho_1(15) = 5, \rho_3(15) = 30, \rho_5(15) = 5, \rho_{15}(15) =$

$30, \rho_1(10) = 2, \rho_2(10) = 2, \rho_5(10) = 30, \rho_{10}(10) = 30, \rho_1(6) = 3, \rho_2(6) = 30, \rho_3(6) = 3, \rho_6(6) = 30, \rho_1(1) = 30, \rho_{30} = 30$. We use $f(30) = 234$, $f(15) = 78$, $f(10) = 45$, $f(6) = 18$, $f(5) = 15$, $f(3) = 6$, and $f(2) = 3$ [31]. In step 1, we compute x'_{15}, x'_{10}, x'_6, and x'_1 using single exponentiations by using the square and multiply method [18, p. 614]. This costs in total $3(78 \cdot 15 + 45 \cdot 10 + 18 \cdot 6 + 1)(\log q)/2 = 2593.5 \log q$ multiplications in F_q.

In step 2, x_{30} is computed as a simultaneous exponentiation [18, p. 618]in $x \in F_{q^{30}}, x_{15} \in F_{q^{15}}, x_{10} \in F_{q^{10}}, x_6 \in F_{q^6}, x_1 \in F_q$. In a precomputation we compute for each of the 2^5 possible sets $S \subset \{x, x_{15}, x_{10}, x_6, x_1\}$ the product $\prod_{w \in S} w$. The whole precomputation costs at most 2^5 multiplications in $F_{q^{30}}$. In the computation of x_{30} the exponents of x, x_{15}, x_{10}, etc., have bit lengths $30 \log q$, $15 \log q$, $10 \log q$, etc. This means that in the second half of the simultaneous exponentiation (the last $30 \log q - 15 \log q$ bits of the exponents) we only need to square or square-and-multiply with $x \in F_{q^{30}}$. So the average costs in the second half of the simultaneous multiplication is equal to $3(15 \log q)/2$ multiplications in $F_{q^{30}}$. The simultaneous exponentiation corresponding to the bits ranging from position $10 \log q$ to $15 \log q$ involves square or square and multiply with x, x_{15}, or $x \cdot x_{15}$. This costs on average $7(5 \log q)/4$ multiplications (5 is the difference between 15 and 10, on average we need 1 multiplication in 1 out of 4 cases and 2 multiplications in 3 out of 4 cases). Notice that we treat squaring as a single multiplication in this excersise. Continuing this argument we need in total $234(2^5 + 15(3/2) + 5(7/4) + 4(15/8) + 5(31/16) + 1(63/32))(\log q) = 19283.1 \log q$ multiplications in F_q (2^5 comes from preprocessing).

The outputs x_5, x_3 and x_2 are single multiplications in x_{15}, x_6, and x_{10}, respectively costing a total of $3(78 \cdot 15 + 18 \cdot 6 + 45 \cdot 10)(\log q)/2 = 2592 \log q$ multiplications. Concluding, the computation of τ costs approximately $24468.6 \log q$ multiplications in F_q. A single exponentiation in $F_{q^{30}}$ costs $234 \cdot 30(\log q)3/2 = 10530 \log q$ multiplications. Hence, τ costs about 2.32 exponentiations in $F_{q^{30}}$.

In the implementation of $\tau^{\leftarrow}$ we compute x as a single exponentiation in x_{30}, costing $234 \cdot 30(\log q)3/2 = 10530 \log q$ multiplications. In step 5, x_{15} is a simultaneous exponentiation in x_{30} and x_5 (and a table look up for the exponentiation in x'_{15}). This costs $78(2^2 + 25(3/2) + 5(7/4))(\log q) = 3919.5 \log q$ multiplications. Similarly, x_{10} costs $45(2^2 + 28(3/2) + 2(7/4))(\log q) = 2227.5 \log q$ and x_6 costs $18(2^2 + 27(3/2) + 3(7/4))(\log q) = 895.5 \log q$ multiplications. We compute x_1 as a single exponentiation in x_{30}, costing $234 \cdot 30(\log q)3/2 = 10530 \log q$ multiplications. Concluding, the computation of $\tau^{\leftarrow}$ costs approximately $28102.5 \log q$ multiplications, which is equivalent to 2.67 exponentiations in $F_{q^{30}}$.

7 Conclusions and Open Problems

Our fundamental contribution is a compact and efficient representation of elements of $T_n(F_q)$, namely, the construction of bijections θ and θ^{-1} of section 3. This allows us to construct ElGamal signature and encryption schemes meeting the optimal rate of communication, as well as a secret key exchange protocol meeting this rate asymptotically. If the torus conjecture of [24] is proven, the schemes in that paper will also achieve this rate, and moreover, their scheme for DH key exchange will meet the optimal rate even for a single key exchanged. Hence, resolving their conjecture is an important problem. Another important question is whether the computational cost of our schemes can be reduced to a more practical level. Finally, our representation of $T_n(F_q)$ may have other applications.

References

[1] L. M. Adelman, J. DeMarrais, *A Subexponential Algorithm for Discrete Logarithms over All Finite Fields*, in Advances in Cryptology – Crypto '93, LNCS 773, Springer-Verlag 1994, 147-158.

[2] D. Boneh and R. Venkatesan, *Rounding in lattices and its cryptographic applications*, Proc. 8-rd Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, NY, 1997, 675– 681.

[3] W. Bosma, J. Hutton, and E. R. Verheul, *Looking Beyond XTR*, in Advances in Cryptology – Asiacrypt '02, LNCS **2501**, Springer, Berlin, 2002, 46 - 63.

[4] A. E. Brouwer, R. Pellikaan, and E. R. Verheul, *Doing More with Fewer Bits*, In Advances of Cryptology – Asiacrypt '99, LNCS **1716**, Springer, 321-332.

[5] N. G. Chebotarev, Die Bestimmung der Dichtigkeit einer Menge von Primzahlen, welche zu einer gegebenen Substitutionsklasse gehören. Math. Ann. **95**, 191-228 (1926).

[6] H. Cohen and A. K. Lenstra, *Supplement to Implementation of a New Primality Test*, Mathematics of Computation, volume 48, number 177, 1987.

[7] D. Coppersmith, *Fast Evaluation of Logarithms in Fields of Characteristic Two*, IEEE Trans. Inform. Theory 30 (1984), 587-594.

[8] S. R. Finch, *Artin's Constant*, 2.4 in Mathematical Constants, Cambridge, England: Cambridge University Press (2003), 104-110.

[9] D. Gordon, *Discrete Logarithms in GF(p) Using the Number Field Sieve*, SIAM J. Discrete Math. 6 (1993), 312-323.

[10] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory 31(4), 1985, 469-472.

[11] G.H. Hardy and E.M. Wright, An Introduction to the Theory of Numbers, 5th edition, Oxford University Press, 1979.

[12] A. Karatsuba and Y. Ofman. *Multiplication of Multidigit Numbers on Automata*, Soviet Physics Doklady, volume 7, 1963, 595-596.

[13] A. K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, Proceedings of ACISP 97, LNCS 1270, Springer-Verlag 1997, 127-138.

[14] A. K. Lenstra and E. R. Verheul, *The XTR Public Key System*, In Advances of Cryptology – Crypto 2000, LNCS **1880**, Springer, 1-19.

[15] A. K. Lenstra and E. R. Verheul, *An Overview of the XTR Public Key System*, in Public-key cryptography and computational number theory (Warsaw, 2000), de Gruyter, Berlin, 2001, 151-180.

[16] H. W. Lenstra, *The Chebotarev Density Theorem*, URL: http://math.berkeley.edu/ jvoight/notes/oberwolfach/Lenstra-Chebotarev.pdf

[17] Seongan Lim, Seungjoo Kim, Ikkwon Yie, Jaemoon Kim, Hongsub Lee, *XTR Extended to $GF(p^{6m})$*, Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001, 301-312, Springer Verlag, 2001.

[18] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, 1997.

[19] T. Nagell, "The Cyclotomic Polynomials" and "The Prime Divisors of the Cyclotomic Polynomial", 46 and 48 in Introduction to Number Theory. New York: Wiley, 158-160 and 164-168, 1951.

[20] A. Odlyzko, *Discrete Logarithms: The past and the future*, Designs, Codes and Cryptography, 19 (2000), 129-145.

[21] S. C. Pohlig, M. E. Hellman, *An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance*, IEEE Trans. on IT, 24 (1978), 106-110.

[22] J. M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. Comp., 32 (1978), 918-924.

[23] K. Rubin and A. Silverberg, *Algebraic tori in cryptography*, to appear in High Primes and Misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams, Fields Institute Communications Series, American Mathematical Society, Providence, RI (2004).

[24] K. Rubin and A. Silverberg, *Torus-Based Cryptography*, In Advances of Cryptology – Crypto 2003, LNCS **2729**, Springer, 349-365.

[25] K. Rubin and A. Silverberg, *Using primitive subgroups to do more with fewer bits*, In Algorithmic Number Theory (ANTS VI), Lecture Notes in Computer Science 3076 (2004), Springer, 18-41.

[26] C. P. Schnorr, *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, 4 (1991), 161-174.

[27] O. Schirokauer, *Discrete Logarithms and Local Units*, Phil. Trans. R. Soc. Lond. A 345, 1993, 409-423.

[28] O. Schirokauer, D. Weber, Th. F. Denny, *Discrete Logarithms: the effectiveness of the index calculus method*, Proceedings ANTS II, LNCS 1122, Springer-Verlag 1996.

[29] M. Stam, *Speeding up Subgroup Cryptosystems*, PhD Thesis, Eindhoven University of Technology, 2003.

[30] V. Voskresenskii, Algebraic Groups and Their Birational Invariants, Translations of Mathematical Monographs **179**, American Mathematical Society, Providence, RI, 1998.

[31] A. Weimerskirch and C. Paar, *Generalizations of the Karatsuba Algorithm for Efficient Implementations*, URL:
http://www.crypto.ruhr-uni-bochum.de/Publikationen/, 2003.