APPENDIX A

Background in Computational Number Theory

The material presented in this appendix is merely the minimum needed for the few examples of specific constructions presented in this book. What we cover here are a few structural and algorithmic facts concerning prime and composite numbers. For a more comprehensive treatment, consult any standard textbook (e.g., [10]).

A.1. Prime Numbers

A *prime* is a natural number that is not divisible by any natural number other than itself and 1. For simplicity, say that 1 is NOT a prime.

For a prime P, the *additive group modulo* P, denoted $\mathbb{Z}_P$, consists of the set $\{0, \ldots, P-1\}$ and the operation of *addition* mod P. All elements except the identity (i.e., 0) have order P(in this group). The *multiplicative group modulo* P, denoted $\mathbb{Z}_P^*$, consists of the set $\{1, \ldots, P-1\}$ and the operation of *multiplication* mod P. This group is cyclic too. In fact, at least $1/\log_2 P$ of the elements of the group have order $P-1$ and are called *primitive*.[1]

A.1.1. Quadratic Residues Modulo a Prime

A *quadratic residue modulo a prime* P is an integer s such that there exists an $r \in \mathbb{Z}_P^*$ satisfying $s \equiv r^2 \pmod{P}$. Thus, in particular, s has to be relatively prime to P. Clearly, if r is a square root of s modulo P, then so is $-r$ (since $(-r)^2 \equiv r^2$). Furthermore, if $x^2 \equiv s \pmod{P}$ has a solution modulo P, then it has exactly two such solutions (as otherwise $r_1 \not\equiv \pm r_2 \pmod{P}$ are both solutions, and $0 \equiv r_1^2 - r_2^2 \equiv (r_1 - r_2)(r_1 + r_2) \pmod{P}$ follows, in contradiction to the primality of P).

The quadratic residues modulo P form a subgroup of the multiplicative group modulo P. The former subgroup contains exactly half of the members of the group.

[1] The exact number of primitive elements modulo P depends on the prime factorization of $P-1 = \prod_{i=1}^{t} P_i^{e_i}$ (see Section A.2): It equals $\prod_{i=1}^{t}((P_i - 1) \cdot P_i^{e_i - 1})$.

Furthermore, squaring modulo P is a 2-to-1 mapping of the group to the subgroup. In case $P \equiv 3 \pmod 4$, each image of this mapping has one pre-image in the subgroup (i.e., a quadratic residue) and one pre-image that is not in the subgroup (i.e., a non-quadratic residue).[2]

A.1.2. Extracting Square Roots Modulo a Prime

In general, extracting square roots module a prime can be done by using Berlekamp's algorithm [28]. The latter is a randomized algorithm for factoring polynomials modulo a prime. (Note that extracting a square root of s modulo a prime P amounts to solving the equation $x^2 \equiv s \pmod P$, which can be cast as the problem of factoring the polynomial $x^2 - s$ modulo P.)

A more direct approach is possible in the special case in which the prime is congruent to 3 (mod 4), which is the case in most cryptographic applications. In this case we observe that for a quadratic residue $s \equiv x^2 \pmod P$, we have

$$\begin{aligned} s^{(P+1)/4} &\equiv x^{(P+1)/2} \pmod P \\ &\equiv x^{(P-1)/2} \cdot x \pmod P \\ &\equiv \pm x \pmod P \end{aligned}$$

where in the last equality we use Fermat's little theorem, by which $x^{(P-1)/2} \equiv \pm 1 \pmod P$ for every integer x and prime P. Thus, in this special case, we obtain a square root of s modulo P by raising s to the power $\frac{P+1}{4}$ modulo P. (Note that this square root is a quadratic residue modulo P.)

A.1.3. Primality Testers

The common approach to testing whether or not an integer is a prime is to utilize Rabin's randomized primality tester [185], which is related to a deterministic algorithm due to Miller [166].[3] The alternative of using a somewhat different randomized algorithm, discovered independently by Solovay and Strassen [202], seems less popular. Here we present a third alternative, which seems less well known (and was discovered independently by several researchers, one of them being Manuel Blum). The only number-theoretic facts that we use are as follows:

1. For every *prime* $P > 2$, each quadratic residue mod P has exactly two square roots mod P (and they sum up to P).
2. For every odd and non-integer-power *composite* number N, each quadratic residue mod N has at least four square roots mod N.

[2] This follows from the fact that -1 is a non-quadratic residue modulo such primes. In contrast, in case $P \equiv 1 \pmod 4$, it holds that -1 is a quadratic residue modulo P. Thus, in case $P \equiv 1 \pmod 4$, for each quadratic residue the two square roots either are both quadratic residues or are both non-quadratic residues.

[3] Miller's algorithm relies on the Extended Riemann Hypothesis (ERH).

Our algorithm uses as a black box an algorithm, denoted SQRT, that given a prime P and a quadratic residue s mod P, returns a square root of s mod P. There is no guarantee as to what the algorithm does in case the input is not of this form (and, in particular, in case P is not a prime).

Algorithm. On input a natural number $N > 2$, do the following:

1. If N is either even or an integer-power, then `reject`.

2. Uniformly select $r \in \{1, \ldots, N-1\}$ and set $s \leftarrow r^2 \bmod N$.

3. Let $r' \leftarrow \mathtt{SQRT}(N, s)$. If $r' \equiv \pm r \pmod{N}$, then `accept`, else `reject`.

Analysis. By Fact 1, on input a prime number N, the algorithm always accepts (since in this case $\mathtt{SQRT}(N, r^2 \bmod N) = \pm r$ for any $r \in \{1, \ldots, N-1\}$). On the other hand, suppose that N is an odd composite that is not an integer-power. Then, by Fact 2, each quadratic residue s has at least four square roots, and each is equally likely to be chosen at Step 2 (since s yields no information on the specific r). Thus, for every such s, the probability that $\pm\mathtt{SQRT}(N, s)$ is chosen in Step 2 is at most $\frac{2}{4}$. It follows that on input a composite number, the algorithm rejects with probability at least $\frac{1}{2}$.

Comment. The analysis presupposes that the algorithm SQRT is always correct when fed with a pair (P, s), where P is prime and s is a quadratic residue mod P. Such an algorithm was described for the special case where $P \equiv 3 \pmod 4$. Thus, whenever the candidate number is congruent to $3 \pmod 4$, which typically is the case in our applications, this description suffices. For the case $P \equiv 1 \pmod 4$, we employ the randomized modular square-root-extraction algorithm mentioned earlier and observe that in case SQRT has error probability $\varepsilon < \frac{1}{2}$, our algorithm still distinguishes primes from composites (since on the former it accepts with probability at least $1 - \varepsilon > \frac{1}{2}$, whereas on the latter it accepts with probability at most $\frac{1}{2}$). The statistical difference between the two cases can be amplified by invoking the algorithm several times.

We mention that error-free probabilistic polynomial-time algorithms for testing primality do exist [121, 1], but currently are much slower. (These algorithms output either the correct answer or a special `don't know` symbol, where the latter is output with probability at most $\frac{1}{2}$.)

A.1.4. On Uniform Selection of Primes

A simple method for uniformly generating a prime number in some interval, say between N and $2N$, consists of repeatedly selecting at random an integer in this interval and testing it for primality. The question, of course, is, *How many times do we need to repeat the procedure before a prime number is found?* This question is intimately related to the *density of primes*, which has been extensively studied in number theory [7]. For our purposes it suffices to assert that in case the sampling interval is sufficiently large

(when compared with the size of the integers in it), then the density of primes in it is noticeable (i.e., is a polynomial fraction). Specifically, the density of primes in the interval $[N, 2N]$ is $\Theta(1/\log N)$. Hence, on input N, we can expect to hit a prime in the interval $[N, 2N]$ within $\Theta(\log N)$ trials. Furthermore, with probability at least $1 - (1/N)^2$ we will hit a prime before conducting $\Theta((\log N)^2)$ trials. Hence, for all practical purposes, we can confine ourselves to conducting a number of trials that is polynomial (i.e., n^2) in the length of the prime we want to generate (i.e., $n = \log_2 N$). (We comment that an analogous discussion applies for primes that are congruent to 3 mod 4.)

We remark that there exists a probabilistic polynomial-time algorithm [9] that produces a uniformly selected prime P together with the factorization of $P - 1$. The prime factorization of $P - 1$ can be used to verify that a given residue is a generator of the multiplicative group modulo P: If $g^{P-1} \equiv 1 \pmod{P}$ and $g^N \not\equiv 1 \pmod{P}$ for every N that divides $P - 1$, then g is a generator of the multiplicative group modulo P. (Note that it suffices to check that $g^{P-1} \equiv 1 \pmod{P}$ and $g^{(P-1)/Q} \not\equiv 1 \pmod{P}$ for every prime Q that divides $P - 1$.) We mention that a noticeable fraction of the residues modulo P will be generators of the multiplicative group modulo P.

Finally, we comment that more randomness-efficient procedures for generating an n-bit-long prime do exist and utilize only $O(n)$ random bits.[4]

A.2. Composite Numbers

A natural number (other than 1) that is not a prime is called a *composite*. Such a number N is uniquely represented as a product of prime powers; that is, $N = \prod_{i=1}^{t} P_i^{e_i}$, where the P_i's are distinct primes, the e_i's are natural numbers, and either $t > 1$ or $e_1 > 1$. These P_i's are called the *prime factorization of* N. It is widely believed that given a composite number, it is infeasible to find its prime factorization. Specifically, it is assumed that it is infeasible to find the factorization of a composite number that is the product of two random primes. That is, it is assumed that any probabilistic polynomial-time algorithm, given the product of two uniformly chosen n-bit-long primes, can successfully recover these primes only with negligible probability. Rivest, Shamir, and Adleman [191] have suggested the use of this assumption for the construction of cryptographic schemes. Indeed, they have done so in proposing the RSA function, and their suggestion has turned out to have a vast impact (i.e., being the most popular intractability assumption in use in cryptography).

For a composite N, the *additive group modulo* N, denoted $\mathbb{Z}_N$, consists of the set $\{0, \ldots, N - 1\}$ and the operation of *addition* mod N. All elements that are relatively prime to N have order N (in this group). The *multiplicative group modulo* N, denoted $\mathbb{Z}_N^*$, consists of the set of natural numbers that are smaller than N and relatively prime to it, and the operation is *multiplication* mod N.

[4] For example, one can use a generic transformation of [177]. Loosely speaking, the latter transformation takes any polynomial-time linear-space randomized algorithm and returns a similar algorithm that has linear randomness complexity. Note that the selection process described in the preceding text satisfies the premise of the transformation.

A.2.1. Quadratic Residues Modulo a Composite

For simplicity, we focus on odd composite numbers that are not divisible by any strict prime power; that is, we consider numbers of the form $\prod_{i=1}^{t} P_i$, where the P_i's are *distinct odd* primes and $t > 1$.

Let $N = \prod_{i=1}^{t} P_i$ be such a composite number. A *quadratic residue modulo* N is an integer s such that there exists an $r \in \mathbb{Z}_N^*$ satisfying $s \equiv r^2 \pmod{N}$. Using the Chinese Remainder Theorem, one can show that s is a quadratic residue modulo N if and only if it is a quadratic residue modulo each of the P_i's. Suppose that s is a quadratic residue modulo N. Then the equation $x^2 \equiv s \pmod{N}$ has 2^t distinct (integer) solutions modulo N. Again, this can be proved by invoking the Chinese Remainder Theorem: First observe that the system

$$x^2 \equiv s \pmod{P_i} \quad \text{for } i = 1, \ldots, t \tag{A.1}$$

has a solution. Next note that each single equation has two distinct solutions $\pm r_i \pmod{P_i}$, and finally note that each of the 2^t different combinations yields a distinct-solution to Eq. (A.1) modulo N (i.e., a distinct square root of s modulo N).

The quadratic residues modulo N form a subgroup of the multiplicative group modulo N. The subgroup contains exactly a 2^{-t} fraction of the members of the group. Furthermore, for $N = \prod_{i=1}^{t} P_i$ (as before), squaring modulo N is a 2^t-to-1 mapping of the group to the subgroup. For further discussion of this mapping, in the special case where $t = 2$ and $P_1 \equiv P_2 \equiv 3 \pmod 4$, see Section A.2.4.

A.2.2. Extracting Square Roots Modulo a Composite

By the preceding discussion (and the effectiveness of the Chinese Remainder Theorem),[5] it follows that given the prime factorization of N, one can efficiently extract square roots modulo N. On the other hand, any algorithm that extracts square roots modulo a composite can be transformed into a factoring algorithm [187]: It suffices to show how an algorithm for extraction of square roots (modulo a composite N) can be used to produce non-trivial divisors of N. The argument is very similar to the one employed in Section A.1.3, the difference being that there the root-extraction algorithm was assumed to work only for extracting square roots modulo a prime (and such efficient algorithms do exist), whereas here we assume that the algorithm works for extracting square roots modulo composites (and such efficient algorithms are assumed not to exist).

Reduction of Factoring to Extracting Modular Square Roots. On input a composite number N, do the following:

1. Uniformly select $r \in \{1, \ldots, N-1\}$.
2. Compute $g \leftarrow \mathtt{GCD}(N, r)$. If $g > 1$, then *output* g and halt.[6]

[5] Specifically, the system $x \equiv a_i \pmod{P_i}$ for $i = 1, \ldots, t$ is solved by $\sum_{i=1}^{t} c_i \cdot a_i \bmod \prod_{i=1}^{t} P_i$, where $c_i \stackrel{\text{def}}{=} Q_i \cdot (Q_i^{-1} \bmod P_i)$ and $Q_i \stackrel{\text{def}}{=} \prod_{j \neq i} P_j$.

[6] This step takes place in order to allow us to invoke the root-extraction algorithm only on relatively prime pairs (s, N).

3. Set $s \leftarrow r^2 \bmod N$ and invoke the root-extraction algorithm to obtain r' such that $(r')^2 \equiv s \pmod{N}$.

4. Compute $g \leftarrow \mathtt{GCD}(N, r - r')$. If $g > 1$, then *output* g and halt.

In case the algorithm halts with some output, the output is a non-trivial divisors of N. The prime factorization of N can be obtained by invoking the algorithm recursively on each of the two non-trivial divisors of N.

Analysis. We can assume that r selected in Step 1 is relatively prime to N, or else the GCD of r and N yields the desired divisor. Invoking the root-extraction algorithm, we obtain r' such that $(r')^2 \equiv s \equiv r^2 \pmod{N}$. Because the root-extraction algorithm has no information on r (beyond $r^2 \pmod{N}$ with probability $2/2^t$, we have $r' \equiv \pm r \pmod{N}$. Otherwise, $r' \not\equiv \pm r \pmod{N}$, and still $0 \equiv (r - r')(r + r') \pmod{N}$. Therefore, $r - r'$ (resp., $r + r'$) has a non-trivial GCD with N, which is found in Step 4. Thus, with probability at least $\frac{1}{2}$, we obtain a non-trivial divisor of N.

A.2.3. The Legendre and Jacobi Symbols

The *Legendre symbol* of integer r modulo a prime P, denoted $\mathtt{LS}_P(r)$, is defined as 0 if P divides r, as $+1$ in case r is a quadratic residue modulo P, and as -1 otherwise. Thus, for r that is relatively prime to P, the Legendre symbol of r modulo P indicates whether or not r is a quadratic residue.

The Jacobi symbol of residues modulo a composite N is defined based on the prime factorization of N. Let $\prod_{i=1}^{t} P_i^{e_i}$ denote the prime factorization of N. Then the *Jacobi symbol* of r modulo N, denoted $\mathtt{JS}_N(r)$, is defined as $\prod_{i=1}^{t} \mathtt{LS}_{P_i}(r)^{e_i}$. Although the Jacobi symbol (of r modulo N) is defined in terms of the prime factorization of the modulus, the Jacobi symbol can be computed efficiently *without knowledge of the factorization of the modulus*. That is, there exists a polynomial-time algorithm that given a pair (r, N) computes $\mathtt{JS}_N(r)$. The algorithm proceeds in "GCD-like" manner[7] and utilizes the following facts regarding the Jacobi symbol:

1. $\mathtt{JS}_N(r) = \mathtt{JS}_N(r \bmod N)$

2. $\mathtt{JS}_N(a \cdot b) = \mathtt{JS}_N(a) \cdot \mathtt{JS}_N(b)$, and $\mathtt{JS}_N(1) = 1$

3. $\mathtt{JS}_N(2) = (-1)^{(N^2-1)/8}$ (i.e., $\mathtt{JS}_N(2) = -1$ iff $N \equiv 4 \pm 1 \pmod{8}$)

4. $\mathtt{JS}_N(r) = (-1)^{(N-1)(r-1)/4} \cdot \mathtt{JS}_r(N)$ for odd integers N and r

Note that a quadratic residue modulo N must have Jacobi symbol 1, but not all residues of Jacobi symbol 1 are quadratic residues modulo N. (In fact, for $N = \prod_{i=1}^{t} P_i$, as in Section A.2.1, half of the residues with non-zero Jacobi symbols have Jacobi symbol 1, but only a 2^{-t} fraction of these residues are squares modulo N.)[8] The fact that

[7] E.g., $\mathtt{JS}_{21}(10) = \mathtt{JS}_{21}(2) \cdot \mathtt{JS}_{21}(5) = (-1)^{55} \cdot (-1)^{20} \cdot \mathtt{JS}_5(21) = -\mathtt{JS}_5(1) = -1$. In general, Fact 2 is used only with $a = 2$ (i.e., $\mathtt{JS}_N(2 \cdot r) = \mathtt{JS}_N(2) \cdot \mathtt{JS}_N(r)$). Also, at the very beginning, one can use $\mathtt{JS}_{2N}(r) = \mathtt{JS}_2(r) \cdot \mathtt{JS}_N(r) = (r \bmod 2) \cdot \mathtt{JS}_N(r)$.

[8] The elements of $\mathbb{Z}_N^*$ having Jacobi symbol 1 form a subgroup of $\mathbb{Z}_N^*$. This subgroup contains exactly half of the members of the group.

the Jacobi symbol can be computed efficiently (without knowledge of the factorization of the modulus) does *not* seem to yield an *efficient* algorithm for determining whether or not a given residue is a square modulo a given composite (of unknown factorization). In fact, it is believed that determining whether or not a given integer is a quadratic residue modulo a given composite (of unknown factorization) is infeasible. Goldwasser and Micali [123] have suggested use of the conjectured intractability of this problem toward the construction of cryptographic schemes, and that suggestion has been followed in numerous works.

A.2.4. Blum Integers and Their Quadratic-Residue Structure

We call $N = P \cdot Q$, where P and Q are primes, a *Blum integer* if $P \equiv Q \equiv 3 \pmod 4$. For such P (resp., Q), the integer -1 is not a quadratic residue mod P (resp., mod Q), and it follows that -1 is not a quadratic residue modulo N and that -1 has Jacobi symbol 1 mod N.

By earlier discussion, each quadratic residue s modulo N has four square roots, denoted $\pm x$ and $\pm y$, so that $\mathtt{GCD}(N, x \pm y) \in \{P, Q\}$. The important fact about Blum Integers is that exactly one of these square roots is a quadratic residue itself.[9] Consequently, $x \mapsto x^2 \bmod N$ induces a *permutation* on the set of quadratic residues modulo N.

(We comment that some sources use a more general definition of Blum integers, but the preceding special case suffices for our purposes. The term "Blum integers" is commonly used in honor of Manuel Blum, who advocated the use of squaring modulo such numbers as a one-way *permutation*.)

We mention that in case $P \not\equiv Q \pmod 8$, the Jacobi symbol of 4 modulo $N = P \cdot Q$ is -1. In this case, obtaining a square root of 4 mod N that is a quadratic residue itself allows us to factor N (since such a residue r satisfies $r \not\equiv \pm 2 \pmod N$ and $(r-2) \cdot (r+2) \equiv 0 \pmod N$).

[9] Let a and b be such that $a^2 \equiv s \pmod P$ and $b^2 \equiv s \pmod Q$. Then, either a or $-a$ (but not both) is a quadratic residue mod P, and similarly for b. Suppose, without loss of generality, that a (resp., b) is a quadratic residue mod P (resp., mod Q). The x satisfying $x \equiv a \pmod P$ and $x \equiv b \pmod Q$ is a square root of s modulo N that is a quadratic residue itself. The other square roots of s modulo N (i.e., $-x$ and $\pm y$, such that $y \equiv a \pmod P$ and $y \equiv -b \pmod Q$ are not quadratic residues mod N.