University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 7

1 More on Chinese Remaindering

Let $N = pq$, where p, q are distinct primes. We saw last time the notion of *Chinese remaindering*, whereby we can view $x \in \mathbb{Z}_N^*$ as $(x_p, x_q) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$. We also saw how this representation might speed up multiplication in $\mathbb{Z}_N^*$. But it can also speed up exponentiation. For completeness, we state the following results:

Fact 1 *Let N, p, q as above. Let $\leftrightarrow$ denote the "Chinese remaindering" representation of an element in $\mathbb{Z}_N^*$ as discussed above. Then:*

- *If $x \leftrightarrow (x_p, x_q)$ and $y \leftrightarrow (y_p, y_q)$ then $xy \leftrightarrow (x_p y_p \bmod p, x_q y_q \bmod q)$ (Note that computation in the left half of the tuple is always done in $\mathbb{Z}_p^*$ and computation in the right half of the tuple in always done in $\mathbb{Z}_q^*$, so the notation "$\bmod p$", "$\bmod q$" is redundant. From now on, we omit it.)*
- *If $x \leftrightarrow (x_p, x_q)$ then $x^{-1} \leftrightarrow (x_p^{-1}, x_q^{-1})$.*
- *If $x \leftrightarrow (x_p, x_q)$ and k is an integer, then $x^k \leftrightarrow (x_p^k, x_q^k)$.*

These facts can speed up computations. As an example, consider computing $4^{1056} \bmod 15$. Since $15 = 3 \cdot 5$, we can represent 4 as $(1, 4)$. Then $4^{1056} = (1^{1056}, 4^{1056}) = (1, (-1)^{1056}) = (1, 1)$. To get our final answer, we now just need to convert $(1, 1)$ back to an element of $\mathbb{Z}_{15}^*$. We gave a technique for doing this last time, but here we can observe that $1 \in \mathbb{Z}_{15}^*$ has the property that $1 = 1 \bmod 3$ and $1 = 1 \bmod 5$! So our final answer is 1.

We will see below that Chinese remaindering is also a powerful theoretical tool, enabling us to easily prove many useful theorems.

2 Quadratic Residues

The notion of *quadratic residues* pops up very often in cryptography. An element $a \in \mathbb{Z}_k^*$ is a quadratic residue if and only if it is a square; i.e., if there is an element $x \in \mathbb{Z}_k^*$ such that $x^2 = a \bmod k$. We begin by looking at the case $k = p$, where p is an odd prime. It is a fact that every element in $\mathbb{Z}_p^*$ has either no square roots (i.e., is not a quadratic residue) or has exactly two, distinct square roots, and we now state and prove this formally.

Lemma 1 *For $p \geq 3$ an odd prime, every element $a \in \mathbb{Z}_p^*$ has either no square roots or two distinct square roots in $\mathbb{Z}_p^*$.*

Proof Let $a \in \mathbb{Z}_p^*$. If a has no square roots, we are done. Otherwise, let x be a square root of a. Note that $-x$ is also a square root of a (why?). On the other hand, x and $-x$ are distinct modulo p (this is why we require that $p \neq 2$), so a has at least two square roots. Can there be more? Well, let y be another square root of a. Then $x^2 = y^2$ and thus $x^2 - y^2 = 0$. Algebra gives: $(x-y)(x+y) = 0$. But this has the two solutions $y = \pm x$ (important note: this makes use of the fact that the equation $wz = 0 \bmod p$ has solutions only if $w = 0$ or $z = 0$, or both. This is true when p is prime but is *not* true if p is composite, as we will see below). ■

This lemma also gives us a count of how many quadratic residues there are in $\mathbb{Z}_p^*$. Since every square maps to two, distinct elements of the group, exactly half of the elements of $\mathbb{Z}_p^*$ must be squares (i.e., there are $(p-1)/2$ squares).

We now consider the case $k = N$, where $N = pq$ is a product of two, distinct (odd) primes. How many square roots can elements $a \in \mathbb{Z}_N^*$ have now? We show that each element has either no square roots or exactly *four* distinct square roots.

Theorem 1 *Let $N = pq$ as above. Then an element $a \in \mathbb{Z}_N^*$ has either no square roots or four distinct square roots in $\mathbb{Z}_N^*$.*

Proof If $a \in \mathbb{Z}_N^*$ has no square roots, we are done. So, assume a has at least one square root x. Using Chinese remaindering, let $a \leftrightarrow (a_p, a_q)$ and $x \leftrightarrow (x_p, x_q)$. Since $x^2 = a$, it must be the case that $x_p^2 = a_p \bmod p$ and $x_q^2 = a_q \bmod q$ (by Fact 1). But then a has three more square roots: $(-x_p, x_q)$, $(x_p, -x_q)$, and $(-x_p, -x_q)$ (and these are all distinct, as argued above for the case p prime). Finally, if a had another square root (y_p, y_q) then $y_p^2 = a_p \bmod p$ and $y_q^2 = a_q \bmod q$ so that $y_p = \pm x_p$ and $y_q = \pm x_q$ (as argued above for the case p prime). So these four square roots are the *only* square roots of a. ■

Define $\mathcal{QR}_N$ as the set of quadratic residues in $\mathbb{Z}_N^*$. Note that the theorem above implies that exactly 1/4 of the elements in $\mathbb{Z}_N^*$ are quadratic residues; or $|\mathcal{QR}_N| = |\mathbb{Z}_N^*|/4$.

(As an aside, note why the proof that there are only two square roots given in the case of $\mathbb{Z}_p^*$, p prime, fails here. In particular, it is not the case that if $xy = 0 \bmod N$ then either $x = 0$ or $y = 0$. As an easy counterexample, note that, for any a, b we have [using representations]: $(a, 0) \cdot (0, b) = (0, 0) = 0$. Also, $pq = 0 \bmod N$ although $p, q \neq 0 \bmod N$.)

It is the case that square roots modulo a prime p can be computed in polynomial-time (we may discuss how to do this later in the semester). This allows efficient calculation of square roots modulo N *if* the factors of N are known (by application of the Chinese remainder theorem and Fact 1). We will see below that square roots *cannot* be computed in polynomial time modulo N when the factorization of N is not known, unless factoring can be done in polynomial time.

2.1 Legendre and Jacobi Symbols

Notation has developed for dealing with quadratic residues in modular groups. For elements in $\mathbb{Z}_p^*$ (p prime), define the Legendre symbol as follows:

$$\mathcal{L}_p(y) = \begin{cases} +1 & \text{if } y \text{ is a quadratic residue modulo } p \\ -1 & \text{otherwise.} \end{cases}$$

We can extend this definition to the case $N = pq$ (p, q distinct primes), and define the Jacobi symbol as follows:

$$\mathcal{J}_N(y) = \mathcal{L}_p(y) \cdot \mathcal{L}_q(y).$$

Note that if y is a square in $\mathbb{Z}_N^*$, then we must have $\mathcal{J}_N(y) = +1$. This is so because if y is a square modulo N, and $y \leftrightarrow (y_p, y_q)$, then y_p must be a square modulo p and y_q must be a square modulo q (and hence y is a square modulo both p and q). On the other hand, there are elements y with $\mathcal{J}_N(y) = +1$ which are *not* quadratic residues; these are precisely those elements $y \leftrightarrow (y_p, y_q)$ where neither y_p nor y_q are squares (and hence $\mathcal{L}_p(y_p) = \mathcal{L}_q(y_q) = -1$).

It can be proved that exactly half the elements in $\mathbb{Z}_N^*$ have Jacobi symbol $+1$, and exactly half have Jacobi symbol -1. Furthermore, of those elements with Jacobi symbol $+1$, exactly half of those are quadratic residues.

An important result is that the Jacobi symbol of y modulo N can be computed efficiently (i.e., in polynomial time) *even if the factorization of N is not known.* Thus, if we compute $\mathcal{J}_N(x) = -1$ we know that x cannot be a quadratic residue. On the other hand, given an element x for which $\mathcal{J}_N(x) = +1$ and without the factorization of N, no efficient algorithm is known to determine whether x is a quadratic residue or not. We will use this to build an encryption scheme later in the course.

3 One-Way Functions from Number Theory

The reason we introduced all this number theory was to present some nice constructions of (conjectured) one-way functions and permutations. In fact, all known one-way functions that are used in practice arise from number theory. The reason for this is the nice algebraic properties that these functions have. This is why we prefer *not* to use "multiplication" as our one-way function (which is one-way, as discussed in the previous lecture) based on hardness of factoring: this function is not a permutation, and does not have "nice" algebraic properties.

The first function we will introduce is squaring. For a fixed modulus $N = pq$, where p, q are distinct primes, define the function $f_N : \mathbb{Z}_N^* \to \mathcal{QR}_N$ by $f_N(x) = x^2 \bmod N$.[1] Our aim is to prove the following:

Theorem 2 *If factoring is hard, then the function f_N given above is one-way. (Note that if factoring is not hard, then f_N is decidedly* not *one-way, since we mentioned previously that square roots can be efficiently computed if the factorization of N is known).*

This will be a cool result (once we prove it...)! We get the benefits of the factoring assumption (namely, that factoring is one of the problems most widely-believed to be hard) but also get the benefits of the "nice" algebraic structure of our function.

Since this is our first proof of this sort, we will give some more detail about what it is that we will actually prove. Our aim is to show that if we have an efficient algorithm A that can invert f_N (that is, can compute square roots in $\mathcal{QR}_N$), then we can build another

[1] If you are bothered by the fact that a fixed N is "hardwired" into f_N, you can consider the function $f(x, N) = (x^2 \bmod N, N)$. If you are not bothered, you can ignore this footnote for the purposes of this class.

efficient algorithm A' that can factor numbers. Note there are two components to the proof: we need to construct an algorithm A' that factors numbers (given an arbitrary algorithm A that inverts f_N) and we also need to ensure that our construction is efficient; that is, that A' runs in polynomial time (assuming that A runs in polynomial time).

A little more formally (we will see the full proof next time), assume we have an efficient algorithm A such that:

$$\Pr[N \leftarrow \mathsf{CompositeGen}(1^k); x \leftarrow \mathbb{Z}_N^*; z = x^2; y \leftarrow A(z, N) : y^2 = z] > \epsilon(k), \tag{1}$$

for some (arbitrary) function $\epsilon(k)$. A bit about the notation: recall that "$\leftarrow$" refers to the output of a random process, while "$=$" either means (on the left side of the colon) to assignment or (on the right side of the colon) to equality (i.e., are these two things equal?). In the above notation, $\mathsf{CompositeGen}$ refers to some algorithm which, on input 1^k, outputs a k-bit composite number which is a product of two distinct primes. (Typically, it will be the product of two $k/2$-bit primes, but the workings of the algorithm are actually irrelevant here.) We stress that the nature of $\mathsf{CompositeGen}$ is irrelevant to the proof of security, so let's for now assume we have such an algorithm as a black box.

We want to show how to use A to construct an efficient algorithm A' for which:

$$\Pr[N \leftarrow \mathsf{CompositeGen}(1^k); (p, q) \leftarrow A'(N) : pq = N] > \epsilon'(k), \tag{2}$$

where $\epsilon'(k)$ will be related in some way to $\epsilon(k)$. In words: A' will be given a random value N output by $\mathsf{CompositeGen}$, A' will output p, q, and the probability that $pq = N$ (where this probability is taken over the entire experiment) is at least $\epsilon'(k)$. For our proof, we will want it to be the case that if $\epsilon(k)$ is *not* negligible, then $\epsilon'(k)$ will not be negligible either. Hence, if there exists an efficient algorithm A satisfying (1) with $\epsilon(k)$ not negligible, then there exists an efficient algorithm A' satisfying (2) — but this violates the factoring assumption! (Note: whether or not this actually violates the factoring assumption depends on our definition of $\mathsf{CompositeGen}$. So what we are really assuming is that there is some algorithm $\mathsf{CompositeGen}$ for which factoring the output of $\mathsf{CompositeGen}$ is hard.)

We will see the proof next time…

3.1 Making f a Permutation

As defined above, the domain of f is all of $\mathbb{Z}_N^*$, and the range of f is (of course) $\mathcal{QR}_N$. It would be nice if we could somehow restrict the domain of f to $\mathcal{QR}_N$ and thereby make f a permutation. But how can we guarantee that f is one-to-one when we restrict the domain in this way? Equivalently, how can we guarantee that for any $y \in \mathcal{QR}_N$, exactly one of the four square roots of y is itself in $\mathcal{QR}_N$?

We can do so by restricting N to be a product of two primes p, q where $p = q = 3 \bmod 4$. We now prove that the above condition holds for such N.

Lemma 2 *Let p be a prime such that $p = 3 \bmod 4$. Then $\mathcal{L}_p(-1) = -1$ (i.e., -1 is not a quadratic residue in $\mathbb{Z}_p^*$).*

Using this lemma, we now prove:

Theorem 3 *Let $N = pq$ where p, q are primes and $p = q = 3 \bmod 4$. Then for all $y \in \mathcal{QR}_N$, exactly one of the square roots of y is also in $\mathcal{QR}_N$.*

Proof Recall that the four square roots of y can be represented as (x_p, x_q), $(-x_p, x_q)$, $(x_p, -x_q)$, and $(-x_p, -x_q)$ for some $x_p \in \mathbb{Z}_p^*$ and $x_q \in \mathbb{Z}_q^*$. If x_p is a quadratic residue in $\mathbb{Z}_p^*$ and x_q is a quadratic residue in $\mathbb{Z}_q^*$, then (x_p, x_q) is a quadratic residue in $\mathbb{Z}_N^*$. Furthermore, since -1 is not a quadratic residue in either $\mathbb{Z}_p^*$ or in $\mathbb{Z}_q^*$, it must be the case that $-x_p$ is not a quadratic residue in $\mathbb{Z}_p^*$ and simillarly $-x_q$ is not a quadratic residue in $\mathbb{Z}_q^*$. So none of $(-x_p, x_q)$, $(x_p, -x_q)$, or $(-x_p, -x_q)$ can be quadratic residues in $\mathbb{Z}_N^*$.

We leave the case when (x_p, x_q) is *not* a quadratic residue in $\mathbb{Z}_N^*$ as an exercise for the reader. ■