CPSC 467: Cryptography and Security

Michael J. Fischer

Lecture 20b
November 10, 2020

Quadratic Residues Revisited

- Euler criterion
- Distinguishing Residues from Non-Residues

Encryption Based on Quadratic Residues

- Summary

Secure Random Sequence Generators

- Pseudorandom sequence generators
- Looking random

Quadratic Residues Revisited

QR reprise

Quadratic residues play a key role in the Feige-Fiat-Shamir zero knowledge authentication protocol.

They can also be used to produce a secure probabilistic cryptosystem and a cryptographically strong pseudorandom bit generator.

Before we can proceed to these protocols, we need some more number-theoretic properties of quadratic residues.

Euler criterion

The Euler criterion gives a feasible method for testing membership in QR_p when p is an odd prime.

Theorem (Euler Criterion)

An integer a is a non-trivial[1] quadratic residue modulo an odd prime p iff

$$a^{(p-1)/2} \equiv 1 \pmod{p}.$$

[1] A non-trivial quadratic residue is one that is not equivalent to 0 (mod p).

Proof of Euler Criterion

Proof in forward direction.

Let $a \equiv b^2 \pmod{p}$ for some $b \not\equiv 0 \pmod{p}$. Then

$$a^{(p-1)/2} \equiv (b^2)^{(p-1)/2} \equiv b^{p-1} \equiv 1 \pmod{p}$$

by Euler's theorem, as desired. □

Proof of Euler Criterion (continued)

Proof in reverse direction.

Suppose $a^{(p-1)/2} \equiv 1 \pmod{p}$. Clearly $a \not\equiv 0 \pmod{p}$. We find a square root b of a modulo p.

Let g be a primitive root of p. Choose k so that $a \equiv g^k \pmod{p}$, and let $\ell = (p-1)k/2$. Then

$$g^\ell \equiv g^{(p-1)k/2} \equiv (g^k)^{(p-1)/2} \equiv a^{(p-1)/2} \equiv 1 \pmod{p}.$$

Since g is a primitive root and $g^\ell \equiv 1 \pmod{p}$, then $\phi(p) \mid \ell$. Hence, $\ell/\phi(p) = \ell/(p-1) = k/2$ is an integer.

Let $b = g^{k/2}$. Then $b^2 \equiv g^k \equiv a \pmod{p}$, so b is a non-trivial square root of a modulo p, as desired. □

A hard problem associated with quadratic residues

Let $n = pq$, where p and q are distinct odd primes.

Recall that each $a \in \mathrm{QR}_n$ has 4 square roots, and $1/4$ of the elements in Z_n^* are quadratic residues.

Some elements of Z_n^* are easily recognized as non-residues, but there is a subset of non-residues (which we denote by Q_n^{00}) that are *hard to distinguish* from quadratic residues without knowing p and q.

Quadratic residues modulo $n = pq$

Let $n = pq$, p, q distinct odd primes.

We divide the numbers in Z_n^* into four classes depending on their membership in QR_p and QR_q.[2]

- Let $Q_n^{11} = \{a \in \mathrm{Z}_n^* \mid a \in \mathrm{QR}_p \cap \mathrm{QR}_q\}$.
- Let $Q_n^{10} = \{a \in \mathrm{Z}_n^* \mid a \in \mathrm{QR}_p \cap \mathrm{QNR}_q\}$.
- Let $Q_n^{01} = \{a \in \mathrm{Z}_n^* \mid a \in \mathrm{QNR}_p \cap \mathrm{QR}_q\}$.
- Let $Q_n^{00} = \{a \in \mathrm{Z}_n^* \mid a \in \mathrm{QNR}_p \cap \mathrm{QNR}_q\}$.

Under these definitions,

$$\mathrm{QR}_n = Q_n^{11}$$

$$\mathrm{QNR}_n = Q_n^{00} \cup Q_n^{01} \cup Q_n^{10}$$

[2] To be strictly formal, we classify $a \in \mathrm{Z}_n^*$ according to whether or not $(a \bmod p) \in \mathrm{QR}_p$ and whether or not $(a \bmod q) \in \mathrm{QR}_q$.

Quadratic residuosity problem

Definition (Quadratic residuosity problem)

The *quadratic residuosity problem* is to decide, given $a \in Q_n^{00} \cup Q_n^{11}$, whether or not $a \in Q_n^{11}$.

Fact

There is no known feasible algorithm for solving the quadratic residuosity problem that gives the correct answer significantly more than 1/2 the time for uniformly distributed random $a \in Q_n^{00} \cup Q_n^{11}$, unless the factorization of n is known.

The *quadratic residuosity assumption* is that no such algorithm exists.

Encryption Based on Quadratic Residues

Securely encrypting single bits

Goldwasser and Micali devised a probabilistic public key cryptosystem based on the assumed hardness of the quadratic residuosity problem that allows one to securely encrypt single bits.

The idea is to encrypt a "0" by a random residue of QR_n and a "1" by a random non-residue in Q_n^{00}. Any ability to decrypt the bit is tantamount to solving the quadratic residuosity problem.

Goldwasser-Micali probabilistic cryptosystem

Key Generation

The public key consists of a pair $e = (n, y)$, where $n = pq$ for distinct odd primes p, q, and y is any member of Q_n^{00}.

The private key consists of the triple $d = (n, y, p)$.

The message space is $\mathcal{M} = \{0, 1\}$. (Single bits!)

The ciphertext space is $\mathcal{C} = Q_n^{00} \cup Q_n^{11}$.

Goldwasser-Micali probabilistic cryptosystem (cont.)

Encryption

To encrypt $m \in \mathcal{M}$, Alice chooses a random $r \in \mathrm{Z}_n^*$ and sets $a = r^2 \bmod n$. The result a is a random element of $\mathrm{QR}_n = Q_n^{11}$.

If $m = 0$, set $c = a$ (which is in Q_n^{11}).
If $m = 1$, set $c = ay \bmod n$ (which is in Q_n^{00}).

Decryption

Bob, knowing the private key p, can use the Euler Criterion to quickly determine whether or not $c \in \mathrm{QR}_p$ and hence whether $c \in Q_n^{11}$ or $c \in Q_n^{00}$, thereby determining m.

Goldwasser-Micali probabilistic cryptosystem (cont.)

Security

Eve's problem of finding m given c is equivalent to the problem of testing if $c \in Q_n^{11}$, given that $c \in Q_n^{00} \cup Q_n^{11}$.

This is just the quadratic residuosity problem, assuming the ciphertexts are uniformly distributed. One can show:

- Every element of Q_n^{11} is equally likely to be chosen as the ciphertext c in case $m = 0$;
- Every element of Q_n^{00} is equally likely to be chosen as the ciphertext c in case $m = 1$.

If the messages are also uniformly distributed, then any element of $Q_n^{00} \cup Q_n^{11}$ is equally likely to be the ciphertext.

Important facts about quadratic residues

1. If p is odd prime, then $|\mathrm{QR}_p| = |\mathrm{Z}_p^*|/2$, and for each $a \in \mathrm{QR}_p$, $|\sqrt{a}| = 2$.
2. If $n = pq$, $p \neq q$ odd primes, then $|\mathrm{QR}_n| = |\mathrm{Z}_n^*|/4$, and for each $a \in \mathrm{QR}_n$, $|\sqrt{a}| = 4$.
3. Euler criterion: $a \in \mathrm{QR}_p$ iff $a^{(p-1)/2} \equiv 1 \pmod{p}$, p odd prime.
4. If p is odd prime, $a \in \mathrm{QR}_p$, can feasibly find $y \in \sqrt{a}$. (See appendix.)
5. If $n = pq$, $p \neq q$ odd primes, then distinguishing Q_n^{00} from Q_n^{11} is believed to be infeasible. Hence, infeasible to find $y \in \sqrt{a}$. Why?
   If not, one could attempt to find $y \in \sqrt{a}$, check that $y^2 \equiv a \pmod{n}$, and conclude that $a \in Q^{11}$ if successful.

Secure Random Sequence Generators

Pseudorandom sequence generators

A *pseudorandom sequence generator (PRSG)* is a function that maps a short *seed* to a long "random-looking" *output sequence*.

The seed typically has length between 32 and a few thousand bits.

The output is typically much longer, ranging from thousands or millions of bits or more, but polynomially related to the seed length.

The output of a PRSG is a sequence that is supposed to "look random".

Incremental generators

In practice, a PRSG is implemented as a co-routine that outputs the next block of bits in the sequence each time it is called. For example, the linux function

```
void srandom(unsigned int seed)
```

sets the 32-bit seed. Each subsequent call on

```
long int random(void)
```

returns an integer in the range $[0, \ldots, RAND\_MAX]$.

On my machine, the return value is 31 bits long (even though `sizeof(long int)` is 64).

Limits on incremental generators

Incremental generators typically are based on state machines with a finite number of states, so the output eventually becomes periodic.

The period of `random()` is said to be approximately $16 * (2^{31} - 1)$.

The output of a PRSG becomes predictable from past outputs once the generator starts to repeat. The point of repetition defines the *maximum usable output length*, even if the implementation allows bits to continue to be produced.

What does it mean for a string to look random?

For the output of a PRSG to look random:

- It must pass common statistical tests of randomness. For example, the frequencies of 0's and 1's in the output sequence must be approximately equal.
- It must lack obvious structure, such as having all 1's occur in pairs.
- It must be difficult to find the seed given the output sequence, since otherwise the whole sequence is easily generated.
- It must be difficult to correctly predict any generated bit, even knowing all of the other bits of the output sequence.
- It must be difficult to distinguish its output from truly random bits.

Chaitin/Kolmogorov randomness

Chaitin and Kolmogorov defined a string to be "random" if its shortest description is almost as long as the string itself.

By this definition, most strings are random by a simple counting argument.

For example, 011011011011011011011011011 is easily described as the pattern 011 repeated 9 times. On the other hand, 101110100010100101001000001 has no obvious short description.

While philosophically very interesting, these notions are somewhat different than the statistical notions that most people mean by randomness and do not seem to be useful for cryptography.

Cryptographically secure PRSG

A PRSG is said to be *cryptographically secure* if its output cannot be *feasibly* distinguished from truly random bits.

In other words, no feasible probabilistic algorithm behaves significantly differently when presented with an output from the PRSG as it does when presented with a truly random string of the same length.

We argue that this definition encompasses all of the desired properties for "looking random" discussed earlier,