CO 409 Cryptography Engineering
Shank's Algorithm

Michael Huth, Imperial College London

January 6, 2019

1 Note and Exercises

Definition 1 *1. Let $\mathbb{Z}_n$ be the set of numbers $\{0, 1, \dots, n-1\}$ for an integer n. We write $\mathbb{Z}_n^*$ for the subset of $\mathbb{Z}_n$ of all those numbers a that have a multiplicative inverse modulo n.*

2. *A* quadratic residue modulo n *is an integer q in $\mathbb{Z}$ for which there exists some x in $\mathbb{Z}$ with*
$$q = x^2 \,(\mathsf{mod}\, n) \tag{1}$$
In that case, we refer to x also as a square root of q modulo n*.*

3. *If q in $\mathbb{Z}$ is not a quadratic residue modulo n, we also refer to q as a* quadratic non-residue modulo n*.*

4. *We write QR_n for the set of elements of $\mathbb{Z}_n^*$ that are quadratic residues modulo n.*

5. *We write QNR_n for the set of elements of $\mathbb{Z}_n^*$ that are quadratic non-residues modulo n.*

Exercise 1 *Let $n > 0$ be an integer and $p > 2$ a prime.*

1. *Let q be in $\mathbb{Z}$.*

   (a) *Show that there exists some x in $\mathbb{Z}$ satisfying (1) iff there exists some x in $\{0, 1, \dots, n-1\}$ satisfying (1).*

   (b) *Explain why q is a quadratic residue modulo n iff $q\,(\mathsf{mod}\, n)$ is a quadratic residue modulo n.*

(c) *Show that every quadratic residue modulo n is equal, modulo n, to some element in* $\{a^2 \,(\mathsf{mod}\, n) \mid 0 \leq a \leq \lfloor n/2 \rfloor + 1\}$.

2. *Show that there are exactly $(p+1)/2$ quadratic residues modulo p in $\mathbb{Z}_p$, including 0; and that there are exactly $(p-1)/2$ quadratic non-residues modulo p in $\mathbb{Z}_p$.*

3. *Prove that $|\mathsf{QR}_p|$ equals $|\mathsf{QNR}_p|$.*

4. *Prove that QR_n is a sub-group of the multiplicative group $\mathbb{Z}_n^*$.*

5. *Prove that for all a in QNR_n and q in QR_n we have that $a \cdot q \,(\mathsf{mod}\, n)$ is in QNR_n.*

6. *Let a be in QR_n. Prove that a square root of a modulo n is also in $\mathbb{Z}_n^*$.*

Definition 2 *Let $p > 2$ be a prime. For any a in $\mathbb{Z}$ relatively prime to p, we define the* Legendre Symbol $\left(\frac{a}{p}\right)$ *by*

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \,(\mathsf{mod}\, p) \tag{2}$$

Exercise 2 *Let $p > 2$ be a prime and $a \neq 0$ in $\mathbb{Z}$ relatively prime to p.*

1. *Use Fermat's Little Theorem, that $a^{p-1} = 1 \,(\mathsf{mod}\, p)$, to infer that*

$$\left(a^{(p-1)/2} - 1\right) \cdot \left(a^{(p-1)/2} + 1\right) = 0 \,(\mathsf{mod}\, p) \tag{3}$$

2. *Use (3) to prove* Euler's Criterion, *that*

   (a) $\left(\frac{a}{p}\right)$ *equals $1 \,(\mathsf{mod}\, p)$ iff a is a quadratic residue modulo p*

   (b) $\left(\frac{a}{p}\right)$ *equals $-1 \,(\mathsf{mod}\, p)$ iff a is a quadratic non-residue modulo p.*

We therefore note that it is efficient to decide whether a given non-zero number a, relatively prime to an odd prime p, is a quadratic residue modulo p. We now want to use this decision procedure to derive an algorithm for computing a witness x with $a = x^2 \,(\mathsf{mod}\, p)$ whenever a is a quadratic residue modulo p.

It is worth noting that there is a rather simple solution for this in the case when $p \,(\mathsf{mod}\, 4)$ equals 3. For a quadratic residue a modulo p we claim that one solution for this is then

$$a^{(p+1)/4} \,(\mathsf{mod}\, p) \tag{4}$$

Moreover, using Iterated Squaring we can compute this value efficiently in the bit sizes of p and a. It is an instructive exercise to show that this does yield a square root of a modulo p:

Exercise 3 *Let p be an odd prime with $p \,(\mathsf{mod}\, 4) = 3$ and $a \neq 0$ a quadratic residue modulo p. Let x be $a^{(p+1)/4} \,(\mathsf{mod}\, p)$. Then $a = x^2 \,(\mathsf{mod}\, p)$.*

We have to invest more work in the remaining case when $p = 1 \,(\mathsf{mod}\, 4)$; the remaining cases $p = 2 \,(\mathsf{mod}\, 4)$ and $p = 0 \,(\mathsf{mod}\, 4)$ are impossible as p is odd. It is an open problem whether there is a deterministic, efficient algorithm that can compute such square roots. But there are efficient randomized algorithms for computing this, Figure 1 shows one of them.

The idea behind the algorithm is as follows. We mean to determine values r and t such that $r^2 = t \cdot a \,(\mathsf{mod}\, p)$. It is clear that this gives us a square root r of a modulo p if $t = 1 \,(\mathsf{mod}\, p)$. Otherwise, we claim that t is a 2^{s-1}-th root of 1 modulo p, meaning that $t^{2^{s-1}} = 1 \,(\mathsf{mod}\, p)$. Setting $m = s - 1$, we then require a process by which we can maintain the equation $r^2 = t \cdot a \,(\mathsf{mod}\, p)$ but modify the values of r and t so that either $t = 1 \,(\mathsf{mod}\, p)$ (and so the modified r is a square root of a modulo p) or t is a 2^{m-1}-th square root of 1 modulo p. We then iterate this process, which will strictly decrease the value of m, until $t = 1 \,(\mathsf{mod}\, p)$ holds, which will definitely happen by the time that m equals 1.

Theorem 1 *Algorithm computeSquareRoot(a, p) returns a square root r of a modulo p, whenever p is an odd prime and a is a quadratic residue of p; it returns such an r in expected polynomial time in the bit sizes of a and p.*

Proof: We first establish that *invariant* is indeed an invariant:

- This invariant holds at program point l_0:
  - We have $c^{2^{m-1}} = c^{2^{s-1}} = (z^q)^{2^{s-1}} = z^{2^{s-1} \cdot q} = z^{(p-1)/2} = -1 \,(\mathsf{mod}\, p)$ since z is in QNR_p.

```
  $computeSquareRoot(a, p)$ {
    $assert\ (p > 2)\ \&\&\ (p\ is\ prime)\ \&\&\ (a \in \mathsf{QR}_p)$;
    $repeat\ \{\ z = random(\mathbb{Z}_p^*)\ \}\ (until\ \left(\frac{z}{p}\right) = -1)$;
    $s = 0;\ q = p - 1$;
    $while\ (q\ even)$ {
      $s = s + 1$;
      $q = q/2$;
    }
    $assert\ (p - 1 = 2^s \cdot q)\ \&\&\ (q\ odd)$;
    $m = s$;
    $c = z^q \pmod{p}$;
    $t = a^q \pmod{p}$;
    $r = a^{(q+1)/2} \pmod{p}$;
$l_0$: $assert\ invariant$;
    $while\ (t \neq 1 \pmod{p})$ {
  $l_1$: $assert\ invariant$;
      $i = \min\{j \mid 0 < j < m, t^{2^j} = 1 \pmod{p}\}$;
      $b = c^{2^{m-i-1}} \pmod{p}$;
      $m = i$;
      $c = b^2 \pmod{p}$;
      $t = t \cdot b^2 \pmod{p}$;
      $r = r \cdot b \pmod{p}$;
  $l_2$: $assert\ invariant$;
    }
    $return\ r$;
  }
```

Figure 1: A version of the Tonelli-Shanks algorithm for computing a square root r of a non-zero, quadratic residue a modulo an odd pime p. The *invariant* is $(c^{2^{m-1}} = -1 \pmod{p}) \wedge (t^{2^{m-1}} = 1 \pmod{p}) \wedge (r^2 = t \cdot a \pmod{p})$. The invariant and labels l_0, l_1, and l_2 are used to prove correctness

  - We have $t^{2^{m-1}} = (a^q)^{2^{m-1}} = (a^q)^{2^{s-1}} = a^{2^{s-1}\cdot q} = a^{(p-1)/2} = 1 \pmod p$ since a is in QR_p.
  - We have $r^2 = (a^{(q+1)/2})^2 = a^{q+1} = a^q \cdot a = t \cdot a \pmod p$.

- Claim: *"This invariant also holds at program point l_2, assuming that it holds at program point l_1"*. For the proof of this claim, we write c and so forth to refer to the value of variables at program point l_1, and use variable names such as c_* to refer to their respective values at program point l_2.
  - We have $c_*^{2^{m_*-1}} = (b^2)^{2^{m_*-1}} = ((c^{2^{m-i-1}})^2)^{2^{m_*-1}} = (c^{2\cdot 2^{m-i-1}})^{2^{m_*-1}} = (c^{2^{m-i}})^{2^{m_*-1}} = c^{2^{m-i+m_*-1}} = c^{2^{m-1}} = -1 \pmod p$ since *invariant* is true at l_1 and since m_* equals i given the assignment for m in the body of the *while*-statement.
  - We have $t_*^{2^{m_*-1}} = (t \cdot b^2)^{2^{m_*-1}} = (t \cdot b^2)^{2^{i-1}} = t^{2^{i-1}} \cdot b^{2^i}$. We need to show that this equals $1 \pmod p$. But this follows if we can show that both $t^{2^{i-1}}$ and b^{2^i} are equal to $-1 \pmod p$, since $(-1)^2 = 1 \pmod p$.
    * We have $t^{2^i} = 1 \pmod p$ by definition of i as the minimal index j with $0 < j < m$ with that property. If $1 < i$, then $0 < i-1 < i < m$ implies that we have $t^{2^{i-1}} \neq 1 \mod p$, since i is minimal amongst all indexes j satisfying $0 < j < m$ and $t^{2^j} = 1 \pmod p$. But $t^{2^{i-1}}$ is a square root of 1 modulo p, since $(t^{2^{i-1}})^2 = t^{2^{i-1}\cdot 2} = t^{2^i} = 1 \pmod p$. The only two square roots of 1 modulo p are 1 and -1, and we saw that $t^{2^{i-1}}$ cannot equal 1. Therefore, $(t^{2^{i-1}})^2 = -1 \pmod p$ as claimed.
      It remains to consider the case when $1 < i$ is not true. Then we have $i = 1$. But then $m = i$ means that m equals 1. Since *invariant* holds at l_1, this implies that $t^{2^{m-1}} = t^{2^{1-1}} = t^{2^0} = t^1 = t = 1 \pmod p$. This is a contradiction, since at program point l_1 the Boolean guard of the *while* loop must be true, and this is $t \neq 1 \pmod p$. Therefore, we see that the case in which i is not larger than 1 cannot arise here.
    * We now need to show that $b^{2^i} = -1 \pmod p$. But $b^{2^i} = (c^{2^{m-i-1}})^{2^i} = c^{2^{m-i-1+i}} = c^{2^{m-1}} = -1 \pmod p$ since *invariant* holds at program point l_1.

– We have $r_*^2 = (r \cdot b)^2 = r^2 \cdot b^2 \pmod{p}$. Since *invariant* holds at program point l_1, the latter equals $(t \cdot a) \cdot b^2 \pmod{p}$, which we can rewrite as $a \cdot (t \cdot b^2) = a \cdot t_* = t_* \cdot a \pmod{p}$ as claimed.

Next, we establish the (partial) correctness of this algorithm: whenever it terminates, it does indeed return a square root of a modulo p. If it terminates, we know that the Boolean guard of the *while* loop is false, and so $t = 1 \pmod{p}$. But we also know that *invariant* is still true at program point l_2. In particular, $r^2 = t \cdot a \pmod{p}$ holds. Putting these two things together, we learn that $r^2 = 1 \cdot a = a \pmod{p}$, and so the returned value r is a square root of a, as claimed.

Next, we establish that this algorithm always terminates, provided that p is an odd prime and a a quadratic residue modulo p. We use the *variant* m for this. At program point l_0, the value of m is s, which is positive since p is odd. It suffices to show that the value of m strictly decreases with each iteration of the *while* loop but stays non-negative. The new value of m is the minimum i computed over the range $0 < j < m$. Therefore, it will be positive and at most $m - 1$. This shows termination.

Finally, we consider efficiency. The algorithm runs in expected time, since we may have to make several calls to the function *random*, which selects an element of $\mathbb{Z}_p^*$ at random. Since half of the elements of $\mathbb{Z}_p^*$ are in QNR_p, the expected number of calls to this function is polynomial in the bit size of p (and independent of the bit size of a). The implementation of the Legendre symbol $\left(\frac{z}{p}\right)$ that reduces it to Euler's criterion is efficient, if we implement exponentiation modulo p via Iterated Squaring. The latter is also used for all other applications of exponentiation modulo p in *computeSquareRoot*. Since the initial value of m is logarithmic in the size of p, there are at most linearly many iterations of the *while* loop in the bit size of p. This shows that the running time of *computeSquareRoot* is in expected polynomial time in the bit sizes of a and p. QED

It is worth reflecting where this algorithm exploited the existence of an element z of QNR_p. This is done in the invariant for $t^{2^{m-1}}$, which ensures that both t and b^2 are 2^{M_*-1}-th roots of -1 modulo p. Note that the invariant also contains the existence of further elements in QNR_p. These elements are not generated randomly, but obtained by multiplying and element of QR_p with an element of QNR_p; we showed in Exercise 1.5 that this computes an element in QNR_p.

As discussed in the above proof, the algorithm in Figure 1 can be implemented to run efficiently in expected time. The latter means that the expected number of random samples needed to find an element k in QNR_p is polynomial in the bit sizes of a and p.

We may be tempted to get rid of the randomization. At the time of writing, it is not known how to do this for general odd primes p and quadradic residues modulo p. For example, a deterministic strategy that would try $1, 2, 3, \ldots$ as candidate square roots will fail in general: using more advanced number theory, one can show that for each $n > 0$ in $\mathbb{N}$ there is some prime p such that $\{1, 2, \ldots, n\}$ is a subset of QR_p. So the above algorithm is a good example of the computational power of randomization.

A natural question to ask is whether the algorithm in Figure 1 can be generalized to work when p is not a prime but any natural number n, which would then be *composite*. One can show that any efficient algorithm for completely factoring natural numbers n can be converted into an efficient algorithm for computing square roots modulo n, and vice versa. In other words, the computational problems FACTORING and SQUAREROOT are equivalent.

Exercise 4 *Let $n = p \cdot q$ be the product of two odd primes p and q. Let a be in $\mathbb{Z}_n^*$.*

1. *Prove that a is in QR_n iff $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$*
2. *Show that anyone who knows the factorization of n can efficiently decide whether an element in $\mathbb{Z}_n^*$ is a quadratic residue modulo n.*

We can use the Chinese Remainder Theorem to compute a square root for a number a that is in QR_n where n is the product of two odd primes p and q. We can use *computeSquareRoot* to compute the square root s_p of a modulo p, and the square root s_q of a modulo q. Then we use the Chinese Remainder Theorem to solve

$$x = s_p \,(\mathsf{mod}\, p) \qquad\qquad x = s_q \,(\mathsf{mod}\, q) \tag{5}$$

Exercise 5 *Show that the unique solution x in (5) is a square root of a modulo n.*

In Security, we pay close attention to such equivalences or weaker dependencies, for example that a solution for FACTORING also solves the RSA problem. Such relationships help us to assess how secure some security mechanism may be. For example, a commitment scheme discussed in Exercise 35 of the course (see also Exercise 34) is based on the hardness of SQUAREROOT for composite numbers. Since we believe that FACTORING is hard, we therefore also have to believe that SQUAREROOT is hard, and so we also believe that commitment scheme is secure.

But such assessments may not withstand the test of time, they are not formally, eternal mathematical truths. For example, there is an efficient algorithm for factoring natural numbers that makes use of quantum computing. So if, and when, we will have quantum computers that can stably process an arbitrary number of quantum bits, the FACTORING problem will be easy to solve. Therefore, at that point in time the SQUAREROOT problem will also be easy to solve, and the commitment scheme that depends on its hardness will no longer be secure.

Post-Quantum Cryptography is an area of security research that aims to develop security mechanisms that will be resilient to the new capabilities of attackers that will have access to scalable quantum computers in their shed.

We can generalize the Legendre symbol to composite numbers using a *Compositionality Principle*: let n be in $\mathbb{N}$ with its unique prime factorization

$$n = p_1^{k_1} \cdot p_2^{k_2} \cdot p_l^{k_l} \tag{6}$$

where $l > 1$ and $k_i \geq 1$ for all $1 \leq i \leq l$. Let a be in $\mathbb{Z}_n^*$. Then the Legendre symbol $\left(\frac{a}{p_i}\right)$ is well defined for all such i, and so we can define the *Jacobi symbol* (with the same notation) as

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{l} \left(\frac{a}{p_i}\right)^{k_i} \tag{7}$$

Let us now define

$$\mathsf{J}_n = \{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\} \tag{8}$$

as the set of all a in $\mathbb{Z}_v^*$ whose Jacobi symbol for n equals 1. One can show that QR_n is a subset of J_n: all quadratric residues modulo n have Jacobi symbol 1. The converse is also true when n is a prime, then $\mathsf{QR}_n = \mathsf{J}_n$. But the converse is not true in general, a price we seem to have paid by defining the Jacobi symbol compositionally in (7). Elements in $\mathsf{J}_n \setminus \mathsf{QR}_n$ are called *pseudo-squares* of n. When n is the product of two primes p and q, e.g., then

$$|\,\mathsf{J}_n \setminus \mathsf{QR}_n\,| = (p-1)\cdot(q-1)/4 \tag{9}$$

But one can compute J_n efficiently in the bit size of n without requiring the unique prime factorization of n, as was the case in its definition in (7). If this efficient method, for input $\left(\frac{a}{n}\right)$, outputs a value $\neq 1 \pmod{n}$, then we know with certainty that a is not a quadratic residue of n. But if the output is $1 \pmod{n}$, we have no evidence for or against the claim that a is a quadratic residue modulo n.

2 Solutions to Exercises

Exercise 1:

1. (a) Since $\{0, 1, \ldots, n-1\}$ is a subset of $\mathbb{Z}$, we only have to show that any solution $q = x^2 \pmod{n}$ for q in $\mathbb{Z}$ also has such a solution where q is in $\{0, 1, \ldots, n-1\}$. So let $q = x^2 \pmod{n}$ for q in $\mathbb{Z}$. Consider the integer $a = q \pmod{n}$. Then a is in $\{0, 1, \ldots, n-1\}$ and $a = q = x^2 \pmod{n}$ since all equations here are modulo n.

   (b) This follows from the previous item: q is a quadratic residue modulo n iff there is some x that makes (1) true iff there is some x that makes (1) true for $q \pmod{n}$ instead of q.

   (c) Clearly, all elements $a^2 \pmod{n}$ are quadratic residues modulo n. We only need to consider a between 0 and $\lfloor n/2 \rfloor + 1$ because of the symmetry $a^2 = (n-a)^2 \pmod{n}$.

2. Clearly, 0 is a quadratic residue modulo p as $0 = 0^2 \pmod{p}$. Since there are exactly p elements in $\{0, 1, \ldots, p-1\}$, it remains to show that there are exactly $(p+1)/2$ quadratic residues modulo p, including 0. If we exclude 0, this means we need to show that there are exactly $(p-1)/2$ quadratic residues in $\mathbb{Z}_p^*$. To see that there are exactly this many quadratic residues modulo a prime p, the most elegant means of showing that is via some group theory:

   An automorphism f of a group $(G, \circ, 1)$ is a function $f\colon G \to G$ such that $f(1) = 1$, $f(a^{-1}) = f(a)^{-1}$, and $f(a \circ b) = f(a) \circ f(b)$ for all a and b

in G. For the group $(\mathbb{Z}_p^*, \cdot, 1)$ it is easy to verify that $f(x) = x^2 \,(\mathsf{mod}\, p)$ is an automorphism on this group.

Note that there are exactly 2 elements in that group $(\mathbb{Z}_p^*, \cdot, 1)$ that f maps to 1, namely 1 and $p-1$ (the latter we can think of as being -1): this is since $x^2 = 1 \,(\mathsf{mod}\, p)$ iff $(x-1)\cdot(x+1) = 0 \,(\mathsf{mod}\, p)$. But then it follows from a simple counting argument, using that f is an automorphism, that the image of f has exactly $(p-1)/2$ elements.

By definition, the image of f is the set of quadratic residues modulo p.

3. Since the QR_p and QNR_p sets are defined to be subsets of $\mathbb{Z}_p^*$, this excludes 0 in QR_p and so the claim follows from the previous item.

4. We have that $1 = 1^1 \,(\mathsf{mod}\, n)$ and so 1 is in QR_n. Let a and b be in QR_n. Then there are x and y with $a = x^2 \,(\mathsf{mod}\, n)$ and $b = y^2 \,(\mathsf{mod}\, n)$. From this, we get that $a \cdot b = x^2 \cdot y^2 = (x \cdot y)^2 \,(\mathsf{mod}\, n)$. This proves that $a \cdot b$ is in QR_n. It remains to show that a^{-1} is in QR_n. By the next item (not an ideal enumeration of items, sorry!), we know that its square root x is also in $\mathbb{Z}_n^*$. Therefore, its multiplicative inverse x^{-1} modulo n exists and we have $a^{-1} = (x^2)^{-1} = (x^{-1})^2 \,(\mathsf{mod}\, n)$, which shows that a^{-1} is a quadratic residue modulo n.

5. There is some $0 \leq x \leq n-1$ with $a = x^2 \,(\mathsf{mod}\, n)$ since a is in QR_n. Since $\mathsf{QR}_n \subseteq \mathbb{Z}_n^*$ we have $a \neq 0$ which implies $x \neq 0$. We need to show that x is in $\mathbb{Z}_n^*$, i.e. that n and x don't have a common factor. Proof by Contradicton: suppose that $k > 1$ is a common factor of x and n. There this is also a common factor of x^2 and n, but $x^2 = a \,(\mathsf{mod}\, n)$ and so this means that a and n have k as common factor. Thus, a cannot be in $\mathbb{Z}_n^*$ – a contradiction.

Exercise 2:

1. We have that p is an odd prime and $a \neq 0$ such that a and p have no common factor. By Fermat's Little Theorem, we get that $a^{p-1} = 1 \,(\mathsf{mod}\, p)$. Using this, we compute

$$\begin{aligned} \left(a^{(p-1)/2} - 1\right) \cdot \left(a^{(p-1)/2} + 1\right) &= \left(a^{(p-1)/2}\right)^2 + a^{(p-1)/2} - a^{(p-1)/2} - 1 \\ &= \left(a^{(p-1)/2}\right)^2 - 1 \\ &= a^{p-1} - 1 \\ &= 0 \,(\mathsf{mod}\, p) \end{aligned}$$

2. (a) We show both directions of this "if, and only if":

   - Let $\left(\frac{a}{p}\right) = 1$, i.e. $a^{(p-1)/2} = 1 \pmod{p}$. Since p is a prime, there exists a primitive root g – meaning that the subgroup $\langle g \rangle$ of $(\mathbb{Z}_p^*, \cdot, 1)$ generated by g equals the entire group $(\mathbb{Z}_p^*, \cdot, 1)$. Since a is in $\mathbb{Z}_p^*$, there exists therefore some integer t with $a = g^t \pmod{p}$. Then we compute $g^{t \cdot (p-1)/2} = a^{(p-1)/2} = \left(\frac{a}{p}\right) = 1 \pmod{p}$. Since the order of g (which is the size of the subgroup it generates) is $\phi(p) = p - 1$, this means that

   $$t \cdot (p-1)/2 = 0 \pmod{p-1} \tag{10}$$

   Therefore, there is some k in $\mathbb{Z}$ such that $t \cdot (p-1)/2 = k \cdot (p-1)$. Since this is an equation over $\mathbb{Z}$ and $p - 1$ is different from 0, this implies that $t/2 = k$, and so $t = 2 \cdot k$ shows that t is even. But then $x = g^{t/2} \pmod{p}$ is well defined, and clearly a square root of a modulo p. Thus, a is in QR_p.
   - Let a be in QR_p. There there is some x in $\mathbb{Z}$ with $a = x^2 \pmod{p}$. But then $a^{(p-1)/2} = x^{2 \cdot (p-1)/2} = x^{p-1} = 1 \pmod{p}$ by Fermat's Little Theorem.

   (b) This follows from the previous item and Fermat's Little Theorem: the square of $\left(\frac{a}{p}\right)$ equals 1 modulo p. But there are only two square roots of 1 modulo p, which are 1 and -1 (which equals $p - 1$).

Exercise 3: We know that p is an odd prime, p equals 3 modulo 4, and a is a quadratic residue modulo p. The claim is that $a^{(p+1)/4} \pmod{p}$ is a square root of a modulo p. That is, the claim is that $(a^{(p+1)/4})^2 = a \pmod{p}$.

First, note that $(p + 1)/4$ is a natural number since p equals 3 modulo 4. Now, $(a^{(p+1)/4})^2 = a^{2 \cdot (p+1)/4} = a^{(p+1)/2} = a \cdot a^{(p-1)/2} \pmod{p}$. Since a is a quadratic residue modulo p, Euler's Criterion gives us that $a^{(p-1)/2} = 1 \pmod{p}$. Therefore, $a \cdot a^{(p-1)/2} = a \cdot 1 = a \pmod{p}$, which proves the claim.

We prove Exercise 5 first, as its statement is handy in proving Exercise 4.

Exercise 5: Let $n = p \cdot q$ where p and q are two different odd primes. Moreover, let a be a quadratic residue modulo p such that s_p is a square root of a modulo p, and s_q is a square root of a modulo q. The claim is that the system of equations

$$\begin{aligned} x &= s_p \,(\mathsf{mod}\, p) \\ x &= s_q \,(\mathsf{mod}\, q) \end{aligned} \tag{11}$$

has a unique solution modulo n that is also a square root of a modulo n.

Since p and q are different primes, they have no common factor. But then the Chinese Remainder Theorem guarantees the uniqueness of a solution x modulo $n = p \cdot q$ in the system of equations in (11). It remains to show that x is a square root of a modulo p.

We have $x^2 = s_p^2 = a \,(\mathsf{mod}\, p)$ since s_p is a square root of a modulo p. Similarly, $x^2 = s_q^2 = a \,(\mathsf{mod}\, q)$ since s_q is a square root of a modulo q.

From $x^2 = a \,(\mathsf{mod}\, p)$ we infer that there is some k in $\mathbb{Z}$ with $x^2 = a + k \cdot p$. Similarly, $x^2 = a \,(\mathsf{mod}\, q)$ implies that there is some l in $\mathbb{Z}$ with $x^2 = a + l \cdot q$. But then we infer that

$$x^2 - a = k \cdot p = l \cdot q \tag{12}$$

Since p and q are different and primes, the equation $k \cdot p = l \cdot q$ implies that q is a divisor of k. Clearly, p is a divisor of $k \cdot p$. But then $p \cdot q = n$ is a divisor of $k \cdot p$, and so $k \cdot p$ equals 0 modulo n. In (12), we therefore get that $x^2 - a = 0 \,(\mathsf{mod}\, n)$, which means that x is a square root of a modulo n.

Exercise 4: Let $n = p \cdot q$ where p and q are different odd primes. We need to show an "if, and only if" which consists of two parts:

- Let a be a quadratic residue modulo n. By definition, this means that there is some x in $\mathbb{Z}$ such that $x^2 = a \,(\mathsf{mod}\, n)$. Since $k \cdot n = (k \cdot q) \cdot p$ for any integer k, we know that $x^2 = a \,(\mathsf{mod}\, n)$ implies $x^2 = a \,(\mathsf{mod}\, p)$. Therefore, a is a quadratic residue modulo p. By Euler's Criterion, this means that $\left(\frac{a}{p}\right) = 1$. Similarly, we infer that $x^2 = a \,(\mathsf{mod}\, n)$ implies $x^2 = a \,(\mathsf{mod}\, q)$. So a is a quadratic residue modulo q. By Euler's Criterion, this means that $\left(\frac{a}{q}\right) = 1$ as well. This shows that $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$ as claimed.

- Conversely, let $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$. By Euler's criterion, this implies that a is a qaudratic residue modulo p and a quadratic residue modulo q. By definition, this means that there are integers s_p and s_q such that $s_p^2 = a \,(\mathsf{mod}\, p)$ and $s_q^2 = a \,(\mathsf{mod}\, q)$. Consider the system of equations in (11) for these values s_p and s_q. By Exercise 5, this has a unique solution x which is also a square root of a modulo n. But then a is by definition a quadratic residue modulo n, since $x^2 = a \,(\mathsf{mod}\, n)$.