UC SANTA CRUZ

Math 134: Cryptography

Lecture 13: Choosing a prime *p* and a primitive root α

Eoin Mackall

February 7, 2025

University of California, Santa Cruz

Last time

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.
2. Bob chooses a secret integer b with $1 < b < p - 1$ and computes $\beta \equiv \alpha^b \pmod{p}$.

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.
2. Bob chooses a secret integer b with $1 < b < p - 1$ and computes $\beta \equiv \alpha^b \pmod{p}$.
3. Bob sends Alice the triple (p, α, β). This is Bob's *public* key.

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.
2. Bob chooses a secret integer b with $1 < b < p - 1$ and computes $\beta \equiv \alpha^b \pmod{p}$.
3. Bob sends Alice the triple (p, α, β). This is Bob's *public* key.
4. Alice will send her message in the form of an integer $m \in \mathbb{Z}$. If $m > p$, then Alice breaks $m = m_1 m_2 m_3 ...$ into blocks m_i of sizes less than p.

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.
2. Bob chooses a secret integer b with $1 < b < p - 1$ and computes $\beta \equiv \alpha^b \pmod{p}$.
3. Bob sends Alice the triple (p, α, β). This is Bob's *public* key.
4. Alice will send her message in the form of an integer $m \in \mathbb{Z}$. If $m > p$, then Alice breaks $m = m_1 m_2 m_3 \ldots$ into blocks m_i of sizes less than p.
5. For each block m_i, Alice will generate a random integer k_i and she will compute $r_i \equiv \alpha^{k_i} \pmod{p}$ and $t_i \equiv \beta^{k_i} \cdot m_i \pmod{p}$.

The ElGamal encryption algorithm

Alice wants to send a message to Bob. Bob has indicated that he wants to use the ElGamal public-key cryptosystem. What is the process for encryption?

1. Bob chooses a large prime p and a primitive root α for p.
2. Bob chooses a secret integer b with $1 < b < p - 1$ and computes $\beta \equiv \alpha^b \pmod{p}$.
3. Bob sends Alice the triple (p, α, β). This is Bob's *public* key.
4. Alice will send her message in the form of an integer $m \in \mathbb{Z}$. If $m > p$, then Alice breaks $m = m_1 m_2 m_3 \ldots$ into blocks m_i of sizes less than p.
5. For each block m_i, Alice will generate a random integer k_i and she will compute $r_i \equiv \alpha^{k_i} \pmod{p}$ and $t_i \equiv \beta^{k_i} \cdot m_i \pmod{p}$.
6. Alice then sends all of the pairs (r_i, t_i) to Bob.

The ElGamal decryption algorithm

Bob receives a message (r, t) from Alice. What is the process for decryption?

The ElGamal decryption algorithm

Bob receives a message (r, t) from Alice. What is the process for decryption?

1. Bob computes a multiplicative inverse s to r modulo p (i.e. such that $sr \equiv 1 \pmod{p}$) using, for example, the Euclidean Algorithm.

The ElGamal decryption algorithm

Bob receives a message (r, t) from Alice. What is the process for decryption?

1. Bob computes a multiplicative inverse s to r modulo p (i.e. such that $sr \equiv 1 \pmod{p}$) using, for example, the Euclidean Algorithm.
2. Bob uses the secret integer b that he chose earlier to compute the message as $m \equiv ts^b \pmod{p}$.

The ElGamal decryption algorithm

Bob receives a message (r, t) from Alice. What is the process for decryption?

1. Bob computes a multiplicative inverse s to r modulo p (i.e. such that $sr \equiv 1 \pmod{p}$) using, for example, the Euclidean Algorithm.
2. Bob uses the secret integer b that he chose earlier to compute the message as $m \equiv ts^b \pmod{p}$.
3. This works since

$$\begin{aligned} ts^b \equiv (\beta^k m)s^b &\equiv (\alpha^b)^k ms^b \\ &\equiv (\alpha^k)^b ms^b \\ &\equiv r^b ms^b \\ &\equiv (rs)^b m \equiv (1)^b m \equiv m \pmod{p}. \end{aligned}$$

The ElGamal decryption algorithm

Bob receives a message (r, t) from Alice. What is the process for decryption?

1. Bob computes a multiplicative inverse s to r modulo p (i.e. such that $sr \equiv 1 \pmod{p}$) using, for example, the Euclidean Algorithm.
2. Bob uses the secret integer b that he chose earlier to compute the message as $m \equiv ts^b \pmod{p}$.
3. This works since

$$\begin{aligned} ts^b \equiv (\beta^k m)s^b &\equiv (\alpha^b)^k m s^b \\ &\equiv (\alpha^k)^b m s^b \\ &\equiv r^b m s^b \\ &\equiv (rs)^b m \equiv (1)^b m \equiv m \pmod{p}. \end{aligned}$$

Question

How should Bob choose his public key?

Table of contents

Choosing a prime p

The Pohlig-Hellman Algorithm

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

The Pohlig-Hellman Algorithm

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

Let α be a primitive root modulo p, and let $\beta \equiv \alpha^x \pmod{p}$ be some power of α.

The Pohlig-Hellman Algorithm

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

Let α be a primitive root modulo p, and let $\beta \equiv \alpha^x \pmod{p}$ be some power of α.

Note that since $2 \mid p - 1$, we can assume $q_1 = 2$. Let $t = (p-1)/2 \in \mathbb{Z}$.

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

Let α be a primitive root modulo p, and let $\beta \equiv \alpha^x \pmod{p}$ be some power of α.

Note that since $2 \mid p - 1$, we can assume $q_1 = 2$. Let $t = (p - 1)/2 \in \mathbb{Z}$. Then:

$$(\alpha^t)^2 \equiv \alpha^{t \cdot 2} \equiv \alpha^{p-1} \equiv 1 \pmod{p}.$$

Since $t < (p - 1)$ and α is primitive, we must have $\alpha^t \equiv -1 \pmod{p}$.

The Pohlig-Hellman Algorithm

Note that since $2 \mid p-1$, we can assume $q_1 = 2$. Let $t = (p-1)/2 \in \mathbb{Z}$. Then:

$$(\alpha^t)^2 \equiv \alpha^{t \cdot 2} \equiv \alpha^{p-1} \equiv 1 \pmod{p}.$$

Since $t < (p-1)$ and α is primitive, we must have $\alpha^t \equiv -1 \pmod{p}$.

The Pohlig-Hellman Algorithm

Note that since $2 \mid p-1$, we can assume $q_1 = 2$. Let $t = (p-1)/2 \in \mathbb{Z}$. Then:

$$(\alpha^t)^2 \equiv \alpha^{t \cdot 2} \equiv \alpha^{p-1} \equiv 1 \pmod{p}.$$

Since $t < (p-1)$ and α is primitive, we must have $\alpha^t \equiv -1 \pmod{p}$.

Result

Now if we raise $\beta \equiv \alpha^x \pmod{p}$ to the tth power we get:

$$\beta^t \equiv (\alpha^x)^t \equiv (\alpha^t)^x \equiv (-1)^x \pmod{p}.$$

If $\beta^t \equiv 1 \pmod{p}$ then $x \equiv 0 \pmod{2}$, otherwise $x \equiv 1 \pmod{2}$.

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

The Pohlig-Hellman Algorithm

The Pohlig-Hellman Algorithm

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

The Pohlig-Hellman Algorithm

Our goal is to find the discrete log $x = \mathrm{ind}_\alpha(\beta)$.

The Pohlig-Hellman Algorithm

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

The Pohlig-Hellman Algorithm

Our goal is to find the discrete log $x = \mathrm{ind}_\alpha(\beta)$.

To do this we first find, by a successive procedure, the remainder of $x \pmod{q_i^{t_i}}$ for all integers $1 \le i \le s$ and for all $1 \le t_i \le r_i$.

Let p be an odd prime and write

$$p - 1 = q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s}$$

for the prime factorization of $p - 1$, i.e. each q_i is prime and appears with power $r_i > 0$ dividing $p - 1$.

The Pohlig-Hellman Algorithm

Our goal is to find the discrete log $x = \mathrm{ind}_\alpha(\beta)$.

To do this we first find, by a successive procedure, the remainder of $x \pmod{q_i^{t_i}}$ for all integers $1 \leq i \leq s$ and for all $1 \leq t_i \leq r_i$.

Then, we use our results and the Chinese Remainder Theorem to find the value of $x \pmod{p-1}$.

The Pohlig-Hellman Algorithm

Assume that q is prime and that q^r is the power of q dividing $p-1$.

The Pohlig-Hellman Algorithm

Assume that q is prime and that q^r is the power of q dividing $p-1$.

We write $x = \mathrm{ind}_\alpha(\beta)$ in base q so that

$$x = x_0 + x_1 q + x_2 q^2 + x_3 q^3 + \cdots$$

where $x_0, x_1, x_2, \ldots \in [0, q-1]$ are integers.

The Pohlig-Hellman Algorithm

Assume that q is prime and that q^r is the power of q dividing $p-1$.

We write $x = \mathrm{ind}_\alpha(\beta)$ in base q so that

$$x = x_0 + x_1 q + x_2 q^2 + x_3 q^3 + \cdots$$

where $x_0, x_1, x_2, \ldots \in [0, q-1]$ are integers.

Let $t_0 = (p-1)/q \in \mathbb{Z}$. Then:

The Pohlig-Hellman Algorithm

Assume that q is prime and that q^r is the power of q dividing $p-1$.

We write $x = \mathrm{ind}_\alpha(\beta)$ in base q so that

$$x = x_0 + x_1 q + x_2 q^2 + x_3 q^3 + \cdots$$

where $x_0, x_1, x_2, \ldots \in [0, q-1]$ are integers.

Let $t_0 = (p-1)/q \in \mathbb{Z}$. Then:

$$\begin{aligned} t_0 x &= t_0 x_0 + q t_0 (x_1 + x_2 q + x_3 q^2 + \cdots) \\ &= t_0 x_0 + (p-1) n \end{aligned}$$

where $n = (x_1 + x_2 q + x_3 q^2 + \cdots) \in \mathbb{Z}$.

Assume that q is prime and that q^r is the power of q dividing $p-1$.

We write $x = \mathrm{ind}_\alpha(\beta)$ in base q so that

$$x = x_0 + x_1 q + x_2 q^2 + x_3 q^3 + \cdots$$

where $x_0, x_1, x_2, \ldots \in [0, q-1]$ are integers.

Let $t_0 = (p-1)/q \in \mathbb{Z}$. Then:

$$\begin{aligned} t_0 x &= t_0 x_0 + q t_0 (x_1 + x_2 q + x_3 q^2 + \cdots) \\ &= t_0 x_0 + (p-1)n \end{aligned}$$

where $n = (x_1 + x_2 q + x_3 q^2 + \cdots) \in \mathbb{Z}$. This implies

$$\beta^{t_0} \equiv \alpha^{x t_0} \equiv \alpha^{t_0 x_0} \alpha^{(p-1)n} \equiv \alpha^{t_0 x_0} \pmod{p}.$$

The Pohlig-Hellman Algorithm

This implies

$$\beta^{t_0} \equiv \alpha^{xt_0} \equiv \alpha^{t_0 x_0} \alpha^{(p-1)n} \equiv \alpha^{t_0 x_0} \pmod{p}.$$

We can find the value of x_0 by comparing β^{t_0} with the values

$$\alpha^{k(p-1)/q} \pmod{p} \qquad k = 0, 1, ..., q-1.$$

The Pohlig-Hellman Algorithm

This implies

$$\beta^{t_0} \equiv \alpha^{xt_0} \equiv \alpha^{t_0 x_0} \alpha^{(p-1)n} \equiv \alpha^{t_0 x_0} \pmod{p}.$$

We can find the value of x_0 by comparing β^{t_0} with the values

$$\alpha^{k(p-1)/q} \pmod{p} \qquad k = 0, 1, ..., q-1.$$

If $q^2 \mid p-1$ then we can find x_1 by setting $t_1 = (p-1)/q^2$ and $\beta_1 \equiv \beta\alpha^{-x_0} \pmod{p}$.

The Pohlig-Hellman Algorithm

This implies

$$\beta^{t_0} \equiv \alpha^{xt_0} \equiv \alpha^{t_0 x_0} \alpha^{(p-1)n} \equiv \alpha^{t_0 x_0} \pmod{p}.$$

We can find the value of x_0 by comparing β^{t_0} with the values

$$\alpha^{k(p-1)/q} \pmod{p} \qquad k = 0, 1, ..., q-1.$$

If $q^2 \mid p-1$ then we can find x_1 by setting $t_1 = (p-1)/q^2$ and $\beta_1 \equiv \beta\alpha^{-x_0} \pmod{p}$.

A similar procedure gives

$$\beta_1^{t_1} \equiv (\beta\alpha^{-x_0})^{t_1} \equiv (\alpha^{x-x_0})^{t_1} \equiv (\alpha^{(x_1 q + x_2 q^2 + x_3 q^3 + \cdots)})^{t_1} \equiv \alpha^{t_0 x_1} \pmod{p}.$$

The Pohlig-Hellman Algorithm

This implies

$$\beta^{t_0} \equiv \alpha^{x t_0} \equiv \alpha^{t_0 x_0} \alpha^{(p-1)n} \equiv \alpha^{t_0 x_0} \pmod{p}.$$

We can find the value of x_0 by comparing β^{t_0} with the values

$$\alpha^{k(p-1)/q} \pmod{p} \qquad k = 0, 1, ..., q-1.$$

If $q^2 \mid p-1$ then we can find x_1 by setting $t_1 = (p-1)/q^2$ and $\beta_1 \equiv \beta\alpha^{-x_0} \pmod{p}$.

A similar procedure gives

$$\beta_1^{t_1} \equiv (\beta\alpha^{-x_0})^{t_1} \equiv (\alpha^{x-x_0})^{t_1} \equiv (\alpha^{(x_1 q + x_2 q^2 + x_3 q^3 + \cdots)})^{t_1} \equiv \alpha^{t_0 x_1} \pmod{p}.$$

By comparing with the values $\alpha^{k(p-1)/q} \pmod{p}$ again, we may find x_1.

The Pohlig-Hellman Algorithm

In this way we find, for each prime q dividing $p-1$, the value

$$x \equiv x_0 + x_1 q + x_2 q^2 + \cdots + x_{r-1} q^{r-1} \pmod{q^r}$$

where q^r is the power of q dividing $p-1$.

The Pohlig-Hellman Algorithm

In this way we find, for each prime q dividing $p-1$, the value

$$x \equiv x_0 + x_1 q + x_2 q^2 + \cdots + x_{r-1} q^{r-1} \pmod{q^r}$$

where q^r is the power of q dividing $p-1$.

Putting these together using the Chinese Remainder Theorem yields the value of $x \pmod{p-1}$.

The Pohlig-Hellman Algorithm

In this way we find, for each prime q dividing $p-1$, the value

$$x \equiv x_0 + x_1 q + x_2 q^2 + \cdots + x_{r-1} q^{r-1} \pmod{q^r}$$

where q^r is the power of q dividing $p-1$.

Putting these together using the Chinese Remainder Theorem yields the value of $x \pmod{p-1}$.

Take-away

If $p-1$ has only small(ish) prime factors, then using the Pohlig-Hellman algorithm, one can solve the discrete logarithm problem for p.

Example

Let $p = 257 = 2^8 + 1$. This is prime. A primitive root for p is $\alpha = 3$. Let's find the discrete log x of 157 to base 3, so $3^x \equiv 157 \pmod{257}$.

Example

Let $p = 257 = 2^8 + 1$. This is prime. A primitive root for p is $\alpha = 3$. Let's find the discrete log x of 157 to base 3, so $3^x \equiv 157 \pmod{257}$.

We write $x = x_0 + x_1 \cdot 2 + x_2 \cdot 4 + x_3 \cdot 8 + \cdots$. A calculation shows that

$$157^{(257-1)/2} \equiv 1 \pmod{257}$$

so that $x_0 = 0$.

Example

Let $p = 257 = 2^8 + 1$. This is prime. A primitive root for p is $\alpha = 3$. Let's find the discrete log x of 157 to base 3, so $3^x \equiv 157 \pmod{257}$.

We write $x = x_0 + x_1 \cdot 2 + x_2 \cdot 4 + x_3 \cdot 8 + \cdots$. A calculation shows that

$$157^{(257-1)/2} \equiv 1 \pmod{257}$$

so that $x_0 = 0$.

Now

$$\beta_1 \equiv 157 \cdot 3^{-x_0} \equiv 157 \pmod{257}$$

so $\beta_1^{(256)/4} \equiv (157)^{(256)/4} \equiv -1 \pmod{257}$. So $x_1 = 1$.

The Pohlig-Hellman Algorithm

Example

Let $p = 257 = 2^8 + 1$. This is prime. A primitive root for p is $\alpha = 3$. Let's find the discrete log x of 157 to base 3, so $3^x \equiv 157 \pmod{257}$.

We write $x = x_0 + x_1 \cdot 2 + x_2 \cdot 4 + x_3 \cdot 8 + \cdots$. A calculation shows that

$$157^{(257-1)/2} \equiv 1 \pmod{257}$$

so that $x_0 = 0$.

Now

$$\beta_1 \equiv 157 \cdot 3^{-x_0} \equiv 157 \pmod{257}$$

so $\beta_1^{(256)/4} \equiv (157)^{(256)/4} \equiv -1 \pmod{257}$. So $x_1 = 1$.

Continuing we eventually find $x = 2 + 4 + 8 + 64 = 78$.

Choosing a primitive root

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

Recall that for an integer $n > 1$ we have

$$\phi(n) = n \cdot \prod_{q|n,\, q \text{ prime}} \left(1 - \frac{1}{q}\right).$$

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

Recall that for an integer $n > 1$ we have

$$\phi(n) = n \cdot \prod_{q|n,\, q \text{ prime}} \left(1 - \frac{1}{q}\right).$$

Also, we've seen that the number of primitive roots modulo an odd prime p is $\phi(\phi(p)) = \phi(p-1)$.

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

Recall that for an integer $n > 1$ we have

$$\phi(n) = n \cdot \prod_{q|n,\, q \text{ prime}} \left(1 - \frac{1}{q}\right).$$

Also, we've seen that the number of primitive roots modulo an odd prime p is $\phi(\phi(p)) = \phi(p-1)$.

Since $2 \mid (p-1)$ always, we have that $\phi(p-1) \leq (1/2)(p-1)$ with equality if and only if $p-1$ is a power of 2.

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

Recall that for an integer $n > 1$ we have

$$\phi(n) = n \cdot \prod_{q|n,\, q \text{ prime}} \left(1 - \frac{1}{q}\right).$$

Also, we've seen that the number of primitive roots modulo an odd prime p is $\phi(\phi(p)) = \phi(p-1)$.

Since $2 \mid (p-1)$ always, we have that $\phi(p-1) \leq (1/2)(p-1)$ with equality if and only if $p-1$ is a power of 2.

If q is a large prime power then $(1 - \frac{1}{q}) \approx 1$.

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

Recall that for an integer $n > 1$ we have

$$\phi(n) = n \cdot \prod_{q|n,\ q \text{ prime}} \left(1 - \frac{1}{q}\right).$$

Also, we've seen that the number of primitive roots modulo an odd prime p is $\phi(\phi(p)) = \phi(p-1)$.

Since $2 \mid (p-1)$ always, we have that $\phi(p-1) \leq (1/2)(p-1)$ with equality if and only if $p-1$ is a power of 2.

If q is a large prime power then $(1 - \frac{1}{q}) \approx 1$.

If $p-1$ has only a few prime factors, then this value should be close to $(p-1)/2$.

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Empirical evidence

By choosing p such that $p - 1$ has few, large prime factors gives a good probability that any randomly chosen element from $[2, p - 1)$ is a primitive root.

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Example

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Example

Let $p = 2003$, which is prime. We have $p - 1 = 2002 = 2 \cdot 7 \cdot 11 \cdot 13$.

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Example

Let $p = 2003$, which is prime. We have $p - 1 = 2002 = 2 \cdot 7 \cdot 11 \cdot 13$.

For a to be a primitive root we need that

$$\begin{aligned} a^{(p-1)/2} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/7} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/11} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/13} &\not\equiv 1 \pmod{2003} \end{aligned}$$

The proportion of primitive roots mod p

Let p be an odd prime. How do we find a primitive root α modulo p?

Example

Let $p = 2003$, which is prime. We have $p - 1 = 2002 = 2 \cdot 7 \cdot 11 \cdot 13$.

For a to be a primitive root we need that

$$\begin{aligned} a^{(p-1)/2} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/7} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/11} &\not\equiv 1 \pmod{2003} \\ a^{(p-1)/13} &\not\equiv 1 \pmod{2003} \end{aligned}$$

Going through the first few values for a, we see

$$2^{(p-1)/7} \equiv 1 \pmod{2003} \quad \text{and} \quad 3^{(p-1)/2} \equiv 1 \pmod{2003},$$

but $a = 5$ satisfies all of the above.