CS 170: Algorithms

CS 170: Algorithms

Good Morning!

CS 170: Algorithms

Good Morning!

CS 170: Algorithms

Good Morning!

S

CS 170: Algorithms

Good Morning!

Sh

CS 170: Algorithms

Good Morning!

Shh

CS 170: Algorithms

Good Morning!

Shhh

CS 170: Algorithms

Good Morning!

Shhhh

CS 170: Algorithms

Good Morning!

Shhhh.

CS 170: Algorithms

Good Morning!

Shhhh..

CS 170: Algorithms

Good Morning!

Shhhh...

CS 170: Algorithms

Good Morning!

Shhhh....

CS 170: Algorithms

Good Morning!

Shhhh.....

CS 170: Algorithms

Good Morning!

Shhhh.....

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \dots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$$S = \{1a \pmod p, 2a \pmod p, \dots, (p-1)a \pmod p\}$$

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$$S = \{1a \pmod p, 2a \pmod p, \ldots, (p-1)a \pmod p\}$$

and

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a,N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1,\ldots,N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$S = \{1a \pmod p, 2a \pmod p, \ldots, (p-1)a \pmod p\}$

and

$T = \{1 \pmod p, \ldots, (p-1) \pmod p\}$.

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$S = \{1a \pmod p, 2a \pmod p, \ldots, (p-1)a \pmod p\}$

and

$T = \{1 \pmod p, \ldots, (p-1) \pmod p\}$.

have

Math Facts

If $f : S \rightarrow T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \rightarrow T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a, N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$$S = \{1a \pmod p, 2a \pmod p, \ldots, (p-1)a \pmod p\}$$

and

$$T = \{1 \pmod p, \ldots, (p-1) \pmod p\}.$$

have

$$|S| = |T|$$

Math Facts

If $f : S \to T$ is 1-to-1 (injective) and onto(surjective) implies $|S| = |T|$.

If $f : S \to T$ is 1-to-1 than $|T| \geq |S|$.

For a, with $gcd(a,N) = 1$, for $f(x) = ax \mod N$ and $S \subseteq \{1, \ldots, N-1\}$
$f : S$ is 1-to-1. (a has an inverse.)

Used in proof of Fermat's Theorem.

Sets

$$S = \{1a \pmod p, 2a \pmod p, \ldots, (p-1)a \pmod p\}$$

and

$$T = \{1 \pmod p, \ldots, (p-1) \pmod p\}.$$

have

$$|S| = |T|$$

and since $S \subseteq T$ xare the same!

Testing primality.

Carmichael numbers:
nonprime N, where for all $0 < a < N$, $a^{N-1} = 1 \pmod{N}$.

Testing primality.

Carmichael numbers:
nonprime N, where for all $0 < a < N$, $a^{N-1} = 1 \pmod{N}$.

Theorem: For any non prime N, except for "Carmichael" numbers (ridiculously rare), for at least half the $0 < a < N$,

$$a^{N-1} \neq 1 \pmod{N}.$$

Testing primality.

Carmichael numbers:
nonprime N, where for all $0 < a < N$, $a^{N-1} = 1 \pmod N$.

Theorem: For any non prime N, except for "Carmichael" numbers (ridiculously rare), for at least half the $0 < a < N$,

$$a^{N-1} \not\equiv 1 \pmod N.$$

Fermat's Theorem:
If p is prime, and any $a \not\equiv 0 \pmod p$,

$$a^{p-1} \equiv 1 \pmod p.$$

Testing primality.

Carmichael numbers:
nonprime N, where for all $0 < a < N$, $a^{N-1} = 1 \pmod{N}$.

Theorem: For any non prime N, except for "Carmichael" numbers (ridiculously rare), for at least half the $0 < a < N$,

$$a^{N-1} \not\equiv 1 \pmod{N}.$$

Fermat's Theorem:
If p is prime, and any $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Primality or Carmichael (N) :

Do 100 times:
  Choose random $0 < a < N$:
    if $a^{N-1} \neq 1 \mod N$ return "Not prime."
Return "Prime or Carmichael"

Correctness

Correctness

If return “Not Prime”:

Correctness

If return "Not Prime":

N not prime by Fermat's Theorem ($a^{N-1} \neq 1 \mod N$).

Correctness

If return “Not Prime”:

N not prime by Fermat’s Theorem ($a^{N-1} \neq 1 \mod N$).

If return “Prime”:

Correctness

If return "Not Prime":

N not prime by Fermat's Theorem ($a^{N-1} \neq 1 \mod N$).

If return "Prime":

Passed test 100 times.

Correctness

If return "Not Prime":

N not prime by Fermat's Theorem ($a^{N-1} \neq 1 \mod N$).

If return "Prime":

Passed test 100 times.

N is "prime or Carmichael", then the algorithm is correct.

Correctness

If return "Not Prime":

N not prime by Fermat's Theorem ($a^{N-1} \neq 1 \mod N$).

If return "Prime":

Passed test 100 times.

N is "prime or Carmichael", then the algorithm is correct.

N is "not prime (or Carmichael)":

at most $1/2$ probability of passing in each step.

Correctness

If return "Not Prime":

N not prime by Fermat's Theorem ($a^{N-1} \neq 1 \mod N$).

If return "Prime":

Passed test 100 times.

N is "prime or Carmichael", then the algorithm is correct.

N is "not prime (or Carmichael)":

at most $1/2$ probability of passing in each step.

The probability that test is passed 100 times is at most $1/2^{100}$.

Correctness

If return “Not Prime”:

N not prime by Fermat’s Theorem ($a^{N-1} \neq 1 \mod N$).

If return “Prime”:

Passed test 100 times.

N is “prime or Carmichael”, then the algorithm is correct.

N is “not prime (or Carmichael)”:

at most $1/2$ probability of passing in each step.

The probability that test is passed 100 times is at most $1/2^{100}$.

The probability that the algorithm is wrong $\leq \frac{1}{2^{100}}$.

One bad, many bad.

Primes/Carmichael number's:

One bad, many bad.

Primes/Carmichael number's:
All the $0 < a < N$ have $a^{N-1} = 1 \pmod{N}$.

One bad, many bad.

Primes/Carmichael number's:
All the $0 < a < N$ have $a^{N-1} = 1 \pmod{N}$.

Used fact that non-prime/non-Carmichael:

One bad, many bad.

Primes/Carmichael number's:
All the $0 < a < N$ have $a^{N-1} = 1 \pmod N$.

Used fact that non-prime/non-Carmichael:
At least half the a's must fail the test: $a^{N-1} \equiv 1 \pmod N$.

One bad, many bad.

Primes/Carmichael number's:
All the $0 < a < N$ have $a^{N-1} = 1 \pmod N$.

Used fact that non-prime/non-Carmichael:
At least half the a's must fail the test: $a^{N-1} \equiv 1 \pmod N$.

Stronger property than just one "a" failing test?

One bad, many bad.

Primes/Carmichael number's:
All the $0 < a < N$ have $a^{N-1} = 1 \pmod{N}$.

Used fact that non-prime/non-Carmichael:
At least half the a's must fail the test: $a^{N-1} \equiv 1 \pmod{N}$.

Stronger property than just one "a" failing test?

Theorem: If there is any a with $gcd(a, N) = 1$, where $a^{N-1} \neq 1$ then for for at least half $0 < a < N$,

$$a^{N-1} \not\equiv 1 \pmod{N}$$

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod{N}$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod{N}$.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod{N}$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod{N}$.

Proof:
There is a with $\gcd(a,N) = 1$:

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod{N}$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod{N}$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod{N}$.

One bad, half bad.

Theorem: If there is any a with $gcd(a, N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:

There is a with $\gcd(a, N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod{N}$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod{N}$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod{N}$.

Consider b: $b^{N-1} = 1 \pmod{N}$.

Then $(ab)^{N-1}$

One bad, half bad.

Theorem: If there is any a with $gcd(a,N)=1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N)=1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1} b^{N-1}$

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1} b^{N-1} = a^{N-1}$

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad!

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N)=1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N)=1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following:

One bad, half bad.

Theorem: If there is any a with $gcd(a,N)=1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N)=1$: $a^{N-1} \not\equiv 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), *ab* fails test!

How many fail test? (are good).
...*ab* fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following: are exactly half good?

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \not\equiv 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following: are exactly half good?
Yes?

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \neq 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \neq 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following: are exactly half good?
Yes? No?

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \neq 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \not\equiv 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following: are exactly half good?
Yes? No? No.

One bad, half bad.

Theorem: If there is any a with $gcd(a,N) = 1$, where $a^{N-1} \neq 1 \pmod N$ then for at least half $0 < a < N$, $a^{N-1} \neq 1 \pmod N$.

Proof:
There is a with $\gcd(a,N) = 1$: $a^{N-1} \neq 1 \pmod N$.

Consider b: $b^{N-1} = 1 \pmod N$.

Then $(ab)^{N-1} = a^{N-1}b^{N-1} = a^{N-1} \neq 1 \pmod N$.
$(ab)^{N-1} \neq 1 \mod N$!

For every bad b (passes test), ab fails test!

How many fail test? (are good).
...ab fails test if b passes!

a has inverse $\pmod N$ since $gcd(a,N) = 1 \pmod N$. $ax \mod N$ is 1-to-1

For each bad b, ab is different and good.

At least as many good (fail test) as bad! At least half are good. Q.E.D.
Test your following: are exactly half good?
Yes? No? No. More than half could be good!

Primality Testing

```
def primalityOrCarmichael(N):
   for i in xrange(100):
     a = random_int(1,N-1)
     if  not( exp(a,N-1,N) == 1):
        return False
   return True
```

Primality Testing

```
def primalityOrCarmichael(N):
   for i in xrange(100):
     a = random_int(1,N-1)
     if  not( exp(a,N-1,N) == 1):
        return False
   return True
```

Correctly determines primality with probability $1/2^{100}$.

Primality Testing

```
def primalityOrCarmichael(N):
   for i in xrange(100):
     a = random_int(1,N-1)
     if  not( exp(a,N-1,N) == 1):
        return False
   return True
```

Correctly determines primality with probability $1/2^{100}$.

Prime or Carmichael: says prime.

Primality Testing

```
def primalityOrCarmichael(N):
   for i in xrange(100):
     a = random_int(1,N-1)
     if  not( exp(a,N-1,N) == 1):
        return False
   return True
```

Correctly determines primality with probability $1/2^{100}$.

Prime or Carmichael: says prime.

Not prime or Carmichael: says prime with probability $\leq \frac{1}{2^100}$.

Public Key Cryptography

Amazon wants to speak privately with

Public Key Cryptography

Amazon wants to speak privately with you!

Public Key Cryptography

Amazon wants to speak privately with you!

and you!

Public Key Cryptography

Amazon wants to speak privately with you!

and you! and you !

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Public Key Cryptography

Amazon wants to speak privately with you!
    and you! and you ! ....

Amazon:

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p, q.

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Public Key Cryptography

Amazon wants to speak privately with you!
 and you! and you ! ....

Amazon:

Generates primes p, q.

Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p,q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N,e). (Keeps secret: p,q,d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.

Public Key Cryptography

Amazon wants to speak privately with you!
    and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.
Amazon gets your message x!!

Public Key Cryptography

Amazon wants to speak privately with you!
    and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.
Amazon gets your message x!!

Works!

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.
Amazon gets your message x!!

Works! Is "efficient".

Public Key Cryptography

Amazon wants to speak privately with you!

and you! and you ! ....

Amazon:

Generates primes p, q.

Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.

Amazon gets your message x!!

Works! Is "efficient".

No one knows how to get x from x^e without secret.

Public Key Cryptography

Amazon wants to speak privately with you!
and you! and you ! ....

Amazon:
Generates primes p, q.
Computes $N = pq$, d and $d = e^{-1} \pmod{(p-1)(q-1)}$

Amazon announces numbers (N, e). (Keeps secret: p, q, d.)

You encrypt message "x": compute $x^e \pmod{N}$.

Amazon decrypts: $(y)^d = (x^e)^d \equiv x \pmod{N}$.
Amazon gets your message x!!

Works! Is "efficient".

No one knows how to get x from x^e without secret.

A single key for everyone! No privately shared keys!!

Running time.

Find primes: $O(tn^3)$ with failure probability $1/2^t$.

Running time.

Find primes: $O(tn^3)$ with failure probability $1/2^t$.

Encrypt: $x^e \mod N$, modular exponentiation, $O(n^3)$.

Running time.

Find primes: $O(tn^3)$ with failure probability $1/2^t$.

Encrypt: $x^e \mod N$, modular exponentiation, $O(n^3)$.

Decrypt: $x^d \mod N$, modular exponentiation, $O(n^3)$.

Correctness

Public Key: (N, e)

Correctness

Public Key: (N,e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod N$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod N$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod p$ for prime p if $gcd(x, p) = 1$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.

Correctness

Public Key: (N,e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod N$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod N$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod p$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod N$.
$x^{ed} = x \pmod N$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x =$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$.
Encrypt: $y = x^e \pmod{N}$.
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$.

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p $(p | x^{ed} - x)$ and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p $(p | x^{ed} - x)$ and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p $(p | x^{ed} - x)$ and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p | x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p!

Correctness

Public Key: (N,e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p|x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.

Correctness

Public Key: (N,e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x, p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p|x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x, p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p | x$ or $gcd(x,p) = 1 \to x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! ! !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! ! ! !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$**.**
Encrypt: $y = x^e \pmod{N}$**.**
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$**.**

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x, p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x, p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! ! ! ! !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$.
Encrypt: $y = x^e \pmod{N}$.
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$.

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! ! ! ! ! !

Correctness

Public Key: (N, e)
Secret Key: $d = e^{-1} \pmod{(p-1)(q-1)}$ **and** $N = pq$.
Encrypt: $y = x^e \pmod{N}$.
Amazon decrypts $(y)^d = (x^e)^d$ **??** $\equiv x \pmod{N}$.

From definition: $ed = 1 \pmod{(p-1)(q-1)}$
Or this way: $ed = 1 + k(p-1)(q-1)$ for some k.
Fermat: $x^{(p-1)} = 1 \pmod{p}$ for prime p if $gcd(x,p) = 1$

Prove: $x^{ed} = x \pmod{N}$.
$x^{ed} = x \pmod{N}$ when $x^{ed} - x = 0 \pmod{pq}$
i.e., $x^{ed} - x$ is divisible by p ($p | x^{ed} - x$) and q.
$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x((x^{p-1})^{k(q-1)} - 1)$.
Either $p|x$ or $gcd(x,p) = 1 \rightarrow x^{(p-1)} = 1 \mod p$
...and $(x^{p-1})^{k(q-1)} - 1 = 0 \mod p$ I.e., is divisible by p.
Either way $x^{ed} - x$ is divisible by p! Also, divisible by q.
Therefore $x^{ed} - x = 0 \pmod{pq}$ and $x^{ed} = x \pmod{N}$.

Amazon correctly decrypts ! ! ! ! ! ! !

Number theory: another CS application.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a)

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree. $\Theta(\log n)$ insert/delete.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree. $\Theta(\log n)$ insert/delete.

Keep them in an array.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree. $\Theta(\log n)$ insert/delete.

Keep them in an array.

$O(1)$ access to array of size around 250.
....but how do you assign IP to location.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree. $\Theta(\log n)$ insert/delete.

Keep them in an array.

$O(1)$ access to array of size around 250.
....but how do you assign IP to location.

Use a hash function to map.

Number theory: another CS application.

Insert a subset (say 250) of IP addresses.

Keep a list. (61a) $\Theta(n)$ search.

Binary search tree. $\Theta(\log n)$ insert/delete.

Keep them in an array.

$O(1)$ access to array of size around 250.
....but how do you assign IP to location.

Use a hash function to map.

For each IP, x, place x in list at position $h(x)$ in array.

Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Hash Functions.

Hash Function: $h : U \rightarrow [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

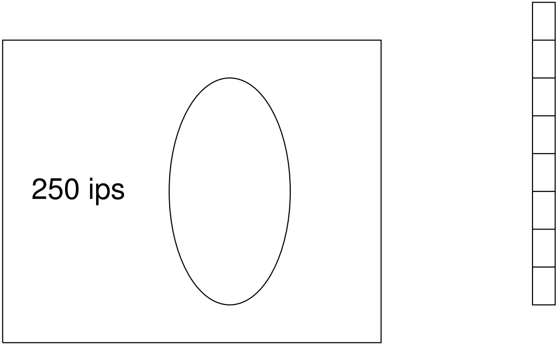
U is large.
n is relatively small.

Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

U is large.
n is relatively small.

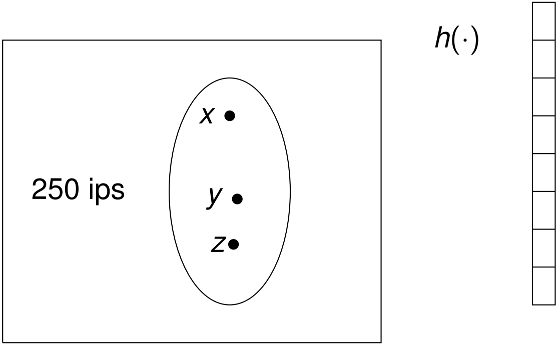


Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

U is large.
n is relatively small.

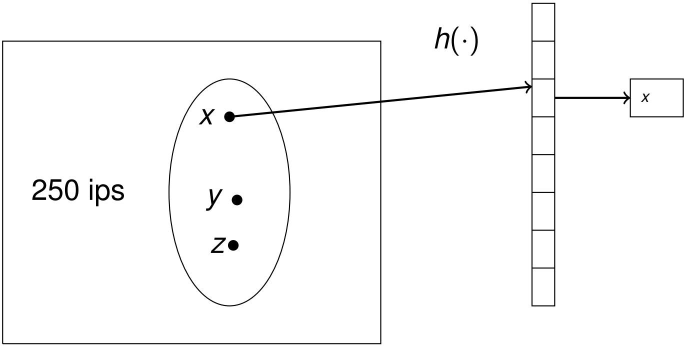


Hash Functions.

Hash Function: $h : U \to [0,\ldots,n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

U is large.
n is relatively small.

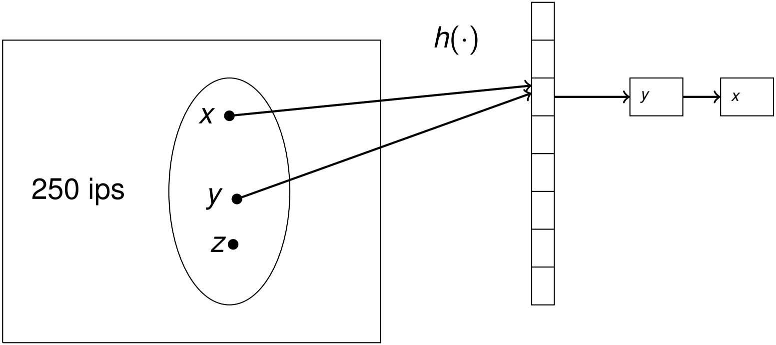


Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

U is large.
n is relatively small.

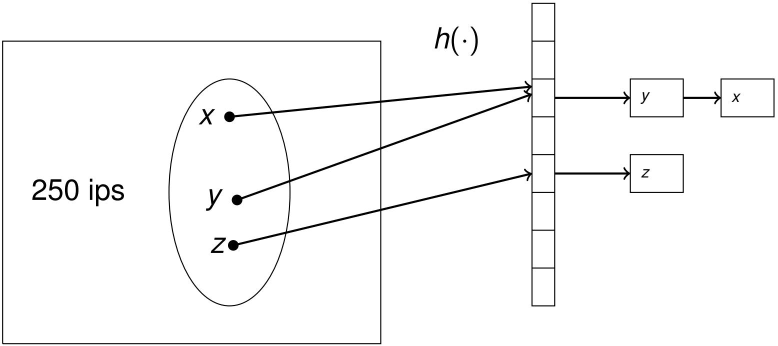


Hash Functions.

Hash Function: $h : U \to [0, \ldots, n-1]$.

Example: U is set of 2^{32} ips, $n = 250$

U is large.
n is relatively small.



Picking a hash function

Last byte?

Picking a hash function

Last byte?

What if ip addresses all have the same last byte?

Picking a hash function

Last byte?

What if ip addresses all have the same last byte?

First byte?

Picking a hash function

Last byte?

What if ip addresses all have the same last byte?

First byte?

What if ip addresses all have the same first byte?

Picking a hash function

Last byte?

What if ip addresses all have the same last byte?

First byte?

What if ip addresses all have the same first byte?

Add the bytes up, divide by two, add 3, shift by 10?

Picking a hash function

Last byte?

What if ip addresses all have the same last byte?

First byte?

What if ip addresses all have the same first byte?

Add the bytes up, divide by two, add 3, shift by 10?

Can this hash function be bad?

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

(C) Really?

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

(C) Really?

B. 4,000,000,000/250

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

(C) Really?

B. 4,000,000,000/250 (or C.)

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

(C) Really?

B. 4,000,000,000/250 (or C.)

For any hash function...

Is there a good hash function?

There are 2^{32} (around 4 billion) ips.

There are 250 table entries.

What is the average number of keys mapped to an entry?

(A) 250

(B) around 160,000,000

(C) Really?

B. 4,000,000,000/250 (or C.)

For any hash function...

there are 160,000,000 ips that map to the same entry!

Solution.

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$

(B) 0

(C) $1/n^2$

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$

(B) 0

(C) $1/n^2$

(D) $1/2^n$

(E) 1

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$
(B) 0
(C) $1/n^2$
(D) $1/2^n$
(E) 1

A.

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$

(B) 0

(C) $1/n^2$

(D) $1/2^n$

(E) 1

A.

The first one goes to some entry.

Solution.

If keys were random (put in a random entry), the probability that two keys collide for a table of size n

(A) $1/n$
(B) 0
(C) $1/n^2$
(D) $1/2^n$
(E) 1

A.
The first one goes to some entry.
The second has n choices, one of which entails a collision.

So...

"Random" keys are good.

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

So...

"Random" keys are good.

Too bad. You don't get to pick the keys!

What if keys are chosen by whomever,

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

What if keys are chosen by whomever,
and then we pick a “random” hash function afterwards?

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

What if keys are chosen by whomever,
and then we pick a “random” hash function afterwards?

Aren’t hash functions usually built first?

So...

"Random" keys are good.

Too bad. You don't get to pick the keys!

What if keys are chosen by whomever,
and then we pick a "random" hash function afterwards?

Aren't hash functions usually built first?
At least by the time the first key is put in table.

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

What if keys are chosen by whomever,
  and then we pick a “random” hash function afterwards?

Aren’t hash functions usually built first?
  At least by the time the first key is put in table.

Assumption Alert:

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

What if keys are chosen by whomever,
and then we pick a “random” hash function afterwards?

Aren’t hash functions usually built first?
At least by the time the first key is put in table.

Assumption Alert:
The set of keys does not depend on the choice of hash function.

So...

“Random” keys are good.

Too bad. You don’t get to pick the keys!

What if keys are chosen by whomever,
and then we pick a “random” hash function afterwards?

Aren’t hash functions usually built first?
At least by the time the first key is put in table.

Assumption Alert:
The set of keys does not depend on the choice of hash function.

Choose randomly from a bunch of hash functions independently from the set of keys.

So...

"Random" keys are good.

Too bad. You don't get to pick the keys!

What if keys are chosen by whomever,
and then we pick a "random" hash function afterwards?

Aren't hash functions usually built first?
At least by the time the first key is put in table.

Assumption Alert:
The set of keys does not depend on the choice of hash function.

Choose randomly from a bunch of hash functions independently from the set of keys.

"A bunch of hash functions" $\equiv$ A class of hash functions.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10)$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1, 1, 1, 1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Is this good?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Is this good?

What is this?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Is this good?

What is this? The hash family.

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Is this good?

What is this? The hash family.
What is good?

Class of hash functions.

Ip addresses consist of four bytes: x_1, x_2, x_3, x_4

Let the number of entries in table be 257, a prime.

Specify hash function: $a = (a_1, a_2, a_3, a_4)$ where $a_i \in [0, \ldots, 256]$.

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Class of functions, indexed by four-tuple from $\{0, \ldots, 256\}$.

EX: $a = (1,1,1,1)$ has $h_a = x_1 + x_2 + x_3 + x_4 \pmod{257}$.
EX: $h_a(192, 168, 1, 10) = 192 + 168 + 1 + 10 = 114 \pmod{257}$
Hash function from family $\equiv$ choice of a_1, a_2, a_3, a_4.

Is this good?

What is this? The hash family.
What is good? ...

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] =$$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

(C) $1/n$

Good hash family of functions?

Hash function h_a specified by $a = (a_1, a_2, a_3, a_4)$

$$h_a(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{257}$$

For arbitrary: $x = (x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3, y_4)$
...where $x \neq y$.

and random $a = (a_1, a_2, a_3, a_4)$,

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n^2$

(B) 1

(C) $1/n$ (as if x and y were placed randomly.)

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}.$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog).

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_4 \pmod{N}.$$

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N . (*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N . (*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1 out

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1 out of

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod{N}$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod{N}.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod{N}$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1 out of n ...

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$

Let's see.

For x and y, $h_a(x) = h_a(y)$, only if

$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \pmod N$.

How probable for random a?

Assume $x_4 \neq y_4$ (wlog). Let a_1, a_2, a_3 be chosen first.

Then $h_a(x) = h_a(y)$ only if

$$X + a_4x_4 = Y + a_4y_4 \pmod N.(*)$$

for $X = a_1x_1 + a_2x_2 + a_3x_3$, and analagous Y.

How many values of a_4 satisfy equation (*)?

$$a_4(x_4 - y_4) = Y - X \pmod N$$

N is prime. So $x_4 - y_4$ has a multiplicative inverse!

So, there is 1 value of a_4 that works..

1 out of n ... $\frac{1}{n}$ is probability of collision.

Wrapping up.

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

A. We just argued this.

Wrapping up.

$$Pr[h_a(x) = h_a(y)] = ???$$

(A) $1/n$

(B) $1/n^2$

(C) 1

A. We just argued this.

Just as if the keys were placed at random!

Universal hashing.

Design pattern.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:
table size n, domain of size n^k,

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:
table size n, domain of size n^k,
choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$h_a(x_1,\ldots,x_k) = a_1x_1 + \cdots a_kx_k \mod n.$

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1 x_1 + \cdots a_k x_k \mod n.$$

This is universal

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:
table size n, domain of size n^k,
choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1 x_1 + \cdots a_k x_k \mod n.$$

This is universal by the same argument as above

Universal hashing.

Design pattern.

Choose a hash function uniformly at random from family $\mathscr{H}$.

Example: $\mathscr{H} = \{h_a : a \in \{0,\ldots,n-1\}^4\}$

A hash family is *universal* if exactly $\frac{1}{n}$ of the hash functions map any pair x and y, $x \neq y$ to the same value.

Another universal family:

table size n, domain of size n^k,

choose a k-tuple, $a = (a_1,\ldots,a_k)$, from $\{0,\ldots,n-1\}$.

$$h_a(x_1,\ldots,x_k) = a_1x_1 + \cdots a_kx_k \mod n.$$

This is universal by the same argument as above if n is prime.

Other applications...

..from polynomials.

Other applications...

..from polynomials.

Error Correcting Codes: Reed-Solomon....

Other applications...

..from polynomials.

Error Correcting Codes: Reed-Solomon....

Complexity theory: error correcting proofs!