Applications of Polynomials: Secret Sharing and Erasure Codes

CS70 Summer 2016 - Lecture 7D

David Dinh

04 August 2016

UC Berkeley

Today

Counting polynomials

Shamir's Secret Sharing

Erasure Codes

Counting Polynomials

How many polynomials of degree at most d are there in $\mathbb{Z}_m$?

Counting Polynomials

How many polynomials of degree at most d are there in $\mathbb{Z}_m$? $d+1$ values for each coefficient, m coefficients, so m^{d+1}.

Counting Polynomials

How many polynomials of degree at most d are there in $\mathbb{Z}_m$? $d+1$ values for each coefficient, m coefficients, so m^{d+1}.

Another way to look at it: polynomial is uniquely determined by $d+1$ points, each of which can take on m values.

Counting Polynomials

How many polynomials of degree at most d are there in $\mathbb{Z}_m$? $d+1$ values for each coefficient, m coefficients, so m^{d+1}.

Another way to look at it: polynomial is uniquely determined by $d+1$ points, each of which can take on m values.

How many polynomials are there that pass through k points that I give you (assuming $k \leq d+1$)?

Counting Polynomials

How many polynomials of degree at most d are there in $\mathbb{Z}_m$? $d+1$ values for each coefficient, m coefficients, so m^{d+1}.

Another way to look at it: polynomial is uniquely determined by $d+1$ points, each of which can take on m values.

How many polynomials are there that pass through k points that I give you (assuming $k \leq d+1$)? m^{d+1-k}. Why? Polynomial fully determined by $d+1$ points. We have k. How we set the remaining $d+1-k$ fully specifies the polynomial.

Secret Sharing (1/2)

Suppose we are designing nuclear launch protocols for the government. Want to require multiple people to get the launch codes (so no one person can launch nukes) but in a nuclear war you can't guarantee that everyone will be alive when the codes are needed.

Secret Sharing (1/2)

Suppose we are designing nuclear launch protocols for the government. Want to require multiple people to get the launch codes (so no one person can launch nukes) but in a nuclear war you can't guarantee that everyone will be alive when the codes are needed.

Shamir's secret sharing scheme: a way to distribute a secret (e.g. nuclear launch codes) such that:

1. A group of sufficient size can recover the secret without all of them needing to be present.
2. No group that is too small to recover the entire secret can recover any information about the secret without the cooperation of more people.

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.
2. Pick a degree-$k-1$ polynomial P such that $P(0) = s$, i.e. $P(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$, where $a_1, \ldots, a_{k-1}$ are chosen randomly.

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.
2. Pick a degree-$k-1$ polynomial P such that $P(0) = s$, i.e. $P(x) = s + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$, where $a_1, ..., a_{k-1}$ are chosen randomly.
3. Give $P(i)$ to the ith official.

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.
2. Pick a degree-$k-1$ polynomial P such that $P(0) = s$, i.e. $P(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$, where $a_1, \ldots, a_{k-1}$ are chosen randomly.
3. Give $P(i)$ to the ith official.

In the event that k officials decide to launch nukes, they can get together, interpolate the polynomial, and get P (and thus $P(0)$).

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.
2. Pick a degree-$k-1$ polynomial P such that $P(0) = s$, i.e. $P(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$, where $a_1, \ldots, a_{k-1}$ are chosen randomly.
3. Give $P(i)$ to the ith official.

In the event that k officials decide to launch nukes, they can get together, interpolate the polynomial, and get P (and thus $P(0)$).

What happens when fewer that k officials go rogue and try to order a nuclear strike? They have less than k points so they can't gain iny information about what $P(0)$ is!

Secret Sharing (2/2)

Suppose we have n government officials. We want to make sure at least k officials approve a nuclear launch before they can get the launch code s.

1. Pick some prime $q > s, n$. We will operate in $GF(q)$.
2. Pick a degree-$k-1$ polynomial P such that $P(0) = s$, i.e. $P(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$, where $a_1, \ldots, a_{k-1}$ are chosen randomly.
3. Give $P(i)$ to the ith official.

In the event that k officials decide to launch nukes, they can get together, interpolate the polynomial, and get P (and thus $P(0)$).

What happens when fewer that k officials go rogue and try to order a nuclear strike? They have less than k points so they can't gain iny information about what $P(0)$ is!To see this: what happens if $k-1$ officials try to get P? There are q polynomials passing through their points, one for every possible value of $P(0)$. No new information gained!

Live Demo

Erasure Codes (1/2)

Polynomial interpolation can also be used to recover data.

Erasure Codes (1/2)

Polynomial interpolation can also be used to recover data.

Same principle as secret sharing!

Packets dropped $\rightarrow$ dead officials.

Packets you receive $\rightarrow$ live officials.

Erasure Codes (1/2)

Polynomial interpolation can also be used to recover data.

Same principle as secret sharing!

Packets dropped $\rightarrow$ dead officials.

Packets you receive $\rightarrow$ live officials.

You want to recover the original message if you receive enough information!

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Has to be a unique degree-$n-1$ polynomial passing through n points in $GF(q)$.

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Has to be a unique degree-$n-1$ polynomial passing through n points in $GF(q)$.

Define a degree-$n-1$ polynomial $P(x)$ passing through $(1, m_1), (2, m_2), \ldots, (n, m_n)$ in $GF(q)$. Want to send enough information to reconstruct this polynomial on the other side of the channel.

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Has to be a unique degree-$n-1$ polynomial passing through n points in $GF(q)$.

Define a degree-$n-1$ polynomial $P(x)$ passing through $(1, m_1), (2, m_2), \ldots, (n, m_n)$ in $GF(q)$. Want to send enough information to reconstruct this polynomial on the other side of the channel.

Trick: send d extra points too! $(n+1, P(n+1)), \ldots, (n+d, P(n+d))$.

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Has to be a unique degree-$n-1$ polynomial passing through n points in $GF(q)$.

Define a degree-$n-1$ polynomial $P(x)$ passing through $(1, m_1), (2, m_2), \ldots, (n, m_n)$ in $GF(q)$. Want to send enough information to reconstruct this polynomial on the other side of the channel.

Trick: send d extra points too! $(n+1, P(n+1)), \ldots, (n+d, P(n+d))$.

No matter which packets are dropped we can recover P and find the original packets!

Erasure Codes (2/2)

Want to send n packets over a lossy channel (each one some number over $GF(q)$, q prime); call the packets $m_1, m_2, \ldots, m_n$. Say the channel drops d packets (although we don't know which).

Has to be a unique degree-$n-1$ polynomial passing through n points in $GF(q)$.

Define a degree-$n-1$ polynomial $P(x)$ passing through $(1, m_1), (2, m_2), \ldots, (n, m_n)$ in $GF(q)$. Want to send enough information to reconstruct this polynomial on the other side of the channel.

Trick: send d extra points too! $(n+1, P(n+1)), \ldots, (n+d, P(n+d))$.

No matter which packets are dropped we can recover P and find the original packets!

Note: does require that $q \geq n+d$, but finding big primes is easy so it's not normally a problem.

Live Demo