Threshold Secret Sharing: Information-Theoretic

March 10, 2020

Threshold Secret Sharing

Zelda has a **secret** $s \in \{0,1\}^n$.

Def: Let $1 \le t \le m$. (t,m)**-secret sharing** is a way for Zelda to give strings to $A_1, \ldots, A_m$ such that:

1. If any t get together than they can learn s
2. If any $t-1$ get together they cannot learn s

What do we mean by **Cannot learn the secret**? We mean info-theory-security. Even if $t-1$ people have big fancy supercomputers they cannot learn s. We will later look at comp-security.

Applications

Rumor: Secret Sharing is used for the Russian Nuclear Codes. There are three people (one is Putin) and if two of them agree to launch, they can launch.

For people signing a contract long distance, secret sharing is used as a building block in the protocol.

$(4,4)$-secret sharing

Zelda has a secret s. A_1, A_2, A_3, A_4 are people. We want:

1. If all four of A_1, A_2, A_3, A_4 get together, they can find s.
2. If any three of them get together, then learn **NOTHING**.

An Attempt at $(4,4)$-Secret Sharing

1. Zelda breaks s up into $s = s_1 s_2 s_3 s_4$ where
$$|s_1| = |s_2| = |s_3| = |s_4| = \frac{n}{4}$$
2. Zelda gives A_i the string s_i.

Does this work?

An Attempt at $(4,4)$-Secret Sharing

1. Zelda breaks s up into $s = s_1 s_2 s_3 s_4$ where

$$|s_1| = |s_2| = |s_3| = |s_4| = \frac{n}{4}$$

2. Zelda gives A_i the string s_i.

Does this work?

1. If A_1, A_2, A_3, A_4 get together they can find s.

An Attempt at $(4,4)$-Secret Sharing

1. Zelda breaks s up into $s = s_1 s_2 s_3 s_4$ where

$$|s_1| = |s_2| = |s_3| = |s_4| = \frac{n}{4}$$

2. Zelda gives A_i the string s_i.

Does this work?

1. If A_1, A_2, A_3, A_4 get together they can find s. **YES!!**

An Attempt at $(4,4)$-Secret Sharing

1. Zelda breaks s up into $s = s_1 s_2 s_3 s_4$ where
$$|s_1| = |s_2| = |s_3| = |s_4| = \frac{n}{4}$$
2. Zelda gives A_i the string s_i.

Does this work?

1. If A_1, A_2, A_3, A_4 get together they can find s. **YES!!**
2. If any three of them get together they learn **NOTHING**.

An Attempt at $(4, 4)$-Secret Sharing

1. Zelda breaks s up into $s = s_1 s_2 s_3 s_4$ where

$$|s_1| = |s_2| = |s_3| = |s_4| = \frac{n}{4}$$

2. Zelda gives A_i the string s_i.

Does this work?

1. If A_1, A_2, A_3, A_4 get together they can find s. **YES!!**
2. If any three of them get together they learn **NOTHING**. **NO**.
   2.1 A_1 learns s_1 which is $\frac{1}{4}$ **of the secret!**
   2.2 A_1, A_2 learn $s_1 s_2$ which is $\frac{1}{2}$ **of the secret!**
   2.3 A_1, A_2, A_3 learn $s_1 s_2 s_3$ which is $\frac{3}{4}$ **of the secret!**

What do we mean by **NOTHING**?

If any three of them get together they learn **NOTHING**
Informally:

1. Before Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$.
2. After Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$. (This is the same $BLAH_{i,j,k}$ as in the first point.
3. Giving out the shares tells A_1, A_2, A_3, A_4 **NOTHING** that they did not already know.

We assume A_i, A_j, A_k have **unlimited computing power**.

What do we mean by NOTHING?

If any three of them get together they learn **NOTHING**
Informally:

1. Before Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$.
2. After Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$. (This is the same $BLAH_{i,j,k}$ as in the first point.
3. Giving out the shares tells A_1, A_2, A_3, A_4 **NOTHING** that they did not already know.

We assume A_i, A_j, A_k have **unlimited computing power**.
they still learn **NOTHING**.

What do we mean by **NOTHING**?

If any three of them get together they learn **NOTHING**
Informally:

1. Before Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$.
2. After Zelda gives out shares, if any three A_i, A_j, A_k get together, they know $BLAH_{i,j,k}$. (This is the same $BLAH_{i,j,k}$ as in the first point.
3. Giving out the shares tells A_1, A_2, A_3, A_4 **NOTHING** that they did not already know.

We assume A_i, A_j, A_k have **unlimited computing power**.
they still learn **NOTHING**.

Information-Theoretic Security

Is $(4,4)$-Secret Sharing Possible?

VOTE: Is $(4,4)$-Secret sharing possible?

1. YES
2. NO
3. YES given some hardness assumption
4. UNKNOWN TO SCIENCE

Is $(4,4)$-Secret Sharing Possible?

VOTE: Is $(4,4)$-Secret sharing possible?

1. YES
2. NO
3. YES given some hardness assumption
4. UNKNOWN TO SCIENCE

YES

Random String Approach

Zelda gives out shares of the secret

Random String Approach

Zelda gives out shares of the secret

1. Secret $s \in \{0,1\}^n$. Zelda gen **random** $r_1, r_2, r_3 \in \{0,1\}^n$.

Random String Approach

Zelda gives out shares of the secret

1. Secret $s \in \{0,1\}^n$. Zelda gen **random** $r_1, r_2, r_3 \in \{0,1\}^n$.
2. Zelda gives A_1 $s_1 = r_1$.
   Zelda gives A_2 $s_2 = r_2$.
   Zelda gives A_3 $s_3 = r_3$.
   Zelda gives A_4 $s_4 = s \oplus r_1 \oplus r_2 \oplus r_3$

Random String Approach

Zelda gives out shares of the secret

1. Secret $s \in \{0,1\}^n$. Zelda gen **random** $r_1, r_2, r_3 \in \{0,1\}^n$.
2. Zelda gives A_1 $s_1 = r_1$.
   Zelda gives A_2 $s_2 = r_2$.
   Zelda gives A_3 $s_3 = r_3$.
   Zelda gives A_4 $s_4 = s \oplus r_1 \oplus r_2 \oplus r_3$

A_1, A_2, A_3 A_4 Can Recover the Secret

$$s_1 \oplus s_2 \oplus s_3 \oplus s_4 = r_1 \oplus r_2 \oplus r_3 \oplus r_1 \oplus r_2 \oplus r_3 \oplus s = s$$

Random String Approach

Zelda gives out shares of the secret

1. Secret $s \in \{0,1\}^n$. Zelda gen **random** $r_1, r_2, r_3 \in \{0,1\}^n$.
2. Zelda gives A_1 $s_1 = r_1$.
   Zelda gives A_2 $s_2 = r_2$.
   Zelda gives A_3 $s_3 = r_3$.
   Zelda gives A_4 $s_4 = s \oplus r_1 \oplus r_2 \oplus r_3$

A_1, A_2, A_3 A_4 Can Recover the Secret

$$s_1 \oplus s_2 \oplus s_3 \oplus s_4 = r_1 \oplus r_2 \oplus r_3 \oplus r_1 \oplus r_2 \oplus r_3 \oplus s = s$$

Easy to see that if a 3 get together they learn NOTHING

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.
Want A_1, A_2 to determine s, but neither A_1 nor A_2 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_2.

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.
Want A_1, A_2 to determine s, but neither A_1 nor A_2 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_2.
Want A_1, A_3 to determine s, but neither A_1 nor A_3 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_3.

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.
Want A_1, A_2 to determine s, but neither A_1 nor A_2 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_2.
Want A_1, A_3 to determine s, but neither A_1 nor A_3 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_3.
Do same for (A_1, A_4), (A_2, A_3), (A_3, A_4).
Question Is there a problem with this?

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.
Want A_1, A_2 to determine s, but neither A_1 nor A_2 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_2.
Want A_1, A_3 to determine s, but neither A_1 nor A_3 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_3.
Do same for (A_1, A_4), (A_2, A_3), (A_3, A_4).
Question Is there a problem with this? **Answer** Yes.
Zelda gives A_1 r (to use when talking to A_2)
Zelda gives A_1 r (to use when talking to A_3)
Same variable name r is fine if done carefully.
But Zelda needs to tell each A_i which string is used to talk to who.

$(2,4)$-Secret Sharing using Random Strings-Intuitive

The secret is $s \in \{0,1\}^n$.
Want A_1, A_2 to determine s, but neither A_1 nor A_2 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_2.
Want A_1, A_3 to determine s, but neither A_1 nor A_3 alone can.
Idea I Zelda gen rand $r \in \{0,1\}^n$ and gives r to A_1, $r \oplus s$ to A_3.
Do same for (A_1, A_4), (A_2, A_3), (A_3, A_4).
Question Is there a problem with this? **Answer** Yes.
Zelda gives A_1 r (to use when talking to A_2)
Zelda gives A_1 r (to use when talking to A_3)
Same variable name r is fine if done carefully.
But Zelda needs to tell each A_i which string is used to talk to who.

Zelda needs to give A_1 strings of the form
$((1,j),r)$: This is a string to be used when A_1 and A_j are talking.
Caveat Don't need to tell A_1 who he is, but notation will generalize.

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \leq i < j \leq 4$

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \le i < j \le 4$

1. Zelda generates **random** $r \in \{0,1\}^n$.

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \le i < j \le 4$

1. Zelda generates **random** $r \in \{0,1\}^n$.
2. Zelda gives A_i the strings $((i,j), r)$.

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \leq i < j \leq 4$

1. Zelda generates **random** $r \in \{0,1\}^n$.
2. Zelda gives A_i the strings $((i,j), r)$.
3. Zelda gives A_j the strings $((i,j), r \oplus s)$.

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \le i < j \le 4$

1. Zelda generates **random** $r \in \{0,1\}^n$.
2. Zelda gives A_i the strings $((i,j),r)$.
3. Zelda gives A_j the strings $((i,j),r \oplus s)$.

A_i, A_j Can Recover the Secret

A_i takes $((i,j),r)$ and just uses the r.
A_j takes $((i,j),r \oplus s)$ and just uses the $r \oplus s$.
They both compute $r \oplus r \oplus s = s$.

$(2,4)$-Secret Sharing using Random Strings-Formally

The secret is $s \in \{0,1\}^n$.
For each $1 \le i < j \le 4$

1. Zelda generates **random** $r \in \{0,1\}^n$.
2. Zelda gives A_i the strings $((i,j), r)$.
3. Zelda gives A_j the strings $((i,j), r \oplus s)$.

A_i, A_j Can Recover the Secret
A_i takes $((i,j), r)$ and just uses the r.
A_j takes $((i,j), r \oplus s)$ and just uses the $r \oplus s$.
They both compute $r \oplus r \oplus s = s$.
Easy to see that one person learns NOTHING

(m, m)-Random String Method

People: $A_1, \ldots, A_m$. Secret s.

(m, m)-Random String Method

People: $A_1, \ldots, A_m$. Secret s.

1. Zelda gen rand $r_1, \ldots, r_{m-1}$.

(m, m)-Random String Method

People: $A_1, \ldots, A_m$. Secret s.

1. Zelda gen rand $r_1, \ldots, r_{m-1}$.
2. A_1 get r_1
   A_2 get r_2
   $\vdots$
   A_{m-1} gets r_{m-1}
   A_m gets $s \oplus r_1 \oplus \cdots \oplus r_{m-1}$

(m,m)-Random String Method

People: $A_1, \ldots, A_m$. Secret s.

1. Zelda gen rand $r_1, \ldots, r_{m-1}$.
2. A_1 get r_1
   A_2 get r_2
   $\vdots$
   A_{m-1} gets r_{m-1}
   A_m gets $s \oplus r_1 \oplus \cdots \oplus r_{m-1}$
3. If they all get together they will XOR all their strings to get s

(m, m)-Random String Method

People: $A_1, \ldots, A_m$. Secret s.

1. Zelda gen rand $r_1, \ldots, r_{m-1}$.
2. A_1 get r_1
   A_2 get r_2
   $\vdots$
   A_{m-1} gets r_{m-1}
   A_m gets $s \oplus r_1 \oplus \cdots \oplus r_{m-1}$
3. If they all get together they will XOR all their strings to get s

We use this as building block for gen case.

(t, m) Secret Sharing

People: $A_1, \ldots, A_m$. $S_1, \ldots, S_{\binom{m}{t}} \subseteq \{A_1, \ldots, A_m\}$ are t-subsets.

1. For every $1 \le j \le \binom{m}{t}$ Zelda does (t, t) secret sharing with the elements of S_j but also prepends every string with j.
2. If the people in S_j get together they XOR together strings prepended with j (do not use the j).
3. No smaller subset can get the secret.

PRO: Can always do Threshold Secret Sharing.
CON: You are giving people A LOT of strings!

A_i Gets ??? Strings in $(5,10)$-Secret Sharing

If do $(5,10)$ secret sharing then how many strings does A_1 get?

A_1 gets a string for every $J \subseteq \{1,\ldots,10\}$, $|J| = 5$, $1 \in J$.

Equivalent to:

A_1 gets a string for every $J \subseteq \{2,\ldots,10\}$, $|J| = 4$.

How many sets? **Discuss**

A_i Gets ??? Strings in $(5, 10)$-Secret Sharing

If do $(5, 10)$ secret sharing then how many strings does A_1 get?

A_1 gets a string for every $J \subseteq \{1, \ldots, 10\}$, $|J| = 5$, $1 \in J$.

Equivalent to:

A_1 gets a string for every $J \subseteq \{2, \ldots, 10\}$, $|J| = 4$.

How many sets? **Discuss**

$$\binom{9}{4} = 126 \text{ strings}$$

A_i Gets ??? Strings in $(m/2, m)$-Secret Sharing

If do $(m/2, m)$ secret sharing then how many strings does A_1 get?

A_1 gets a string for every $J \subseteq \{1, \ldots, m\}$, $|J| = \frac{m}{2}$, $1 \in J$.
Equivalent to:

A_1 gets a string for every $J \subseteq \{2, \ldots, m\}$, $|J| = \frac{m}{2} - 1$.

How many sets? **Discuss**

A_i Gets ??? Strings in $(m/2, m)$-Secret Sharing

If do $(m/2, m)$ secret sharing then how many strings does A_1 get?

A_1 gets a string for every $J \subseteq \{1, \ldots, m\}$, $|J| = \frac{m}{2}$, $1 \in J$.
Equivalent to:

A_1 gets a string for every $J \subseteq \{2, \ldots, m\}$, $|J| = \frac{m}{2} - 1$.

How many sets? **Discuss**

$$\binom{m-1}{\frac{m}{2}-1} \sim \frac{2^m}{\sqrt{m}} \text{ strings}$$

Thats A LOT of Strings!

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.
VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.

VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

You can always do this with everyone getting 1 string

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.
VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

You can always do this with everyone getting 1 string
I know what you are thinking:

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.

VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

You can always do this with everyone getting 1 string

I know what you are thinking:LOOOONG string.

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.

VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

You can always do this with everyone getting 1 string

I know what you are thinking:LOOOONG string.No.

Reduce The Number of Strings for $(m/2, m)$?

In our $(m/2, m)$-scheme each A_i gets $\sim \frac{2^m}{\sqrt{m}}$ strings.

VOTE

1. Requires roughly 2^m strings.
2. $O(\beta^m)$ strings for some $1 < \beta < 2$ but not poly.
3. $O(m^a)$ strings for some $a > 1$ but not linear.
4. $O(m)$ strings but not m^a with $a < 1$.
5. $O(m^a)$ strings for some $a < 1$ but not logarithmic.
6. $O(\log m)$ strings but not constant.
7. $O(1)$ strings.

You can always do this with everyone getting 1 string

I know what you are thinking:LOOOONG string.No.

You can always do this with everyone getting 1 string that is the same length as the secret

Secret Sharing With Polynomials

Definition $a \sim b$ means $\frac{b}{2} \le a \le 2b$.
We do $(3,6)$-Secret Sharing.

1. Secret s. Zelda picks prime $p \sim 2^{|s|}$, Zelda works mod p. View s as a number is in $\{0, \ldots, p-1\}$.
2. Zelda gen rand numbers $a_2, a_1 \in \{0, \ldots, p-1\}$
3. Zelda forms polynomial $f(x) = a_2x^2 + a_1x + s$.
4. Zelda gives A_1 $f(1)$, A_2 $f(2)$, …, A_6 $f(6)$ (all mod p). These are all of length $|s|$ by padding with 0's. Also give everyone p (does not count for length).

1. Any 3 have 3 points from $f(x)$ so can find $f(x)$, s.
2. Any 2 have 2 points from $f(x)$. From these two points what can they conclude?

Secret Sharing With Polynomials

Definition $a \sim b$ means $\frac{b}{2} \le a \le 2b$.
We do $(3,6)$-Secret Sharing.

1. Secret s. Zelda picks prime $p \sim 2^{|s|}$, Zelda works mod p. View s as a number is in $\{0,\ldots,p-1\}$.
2. Zelda gen rand numbers $a_2, a_1 \in \{0,\ldots,p-1\}$
3. Zelda forms polynomial $f(x) = a_2x^2 + a_1x + s$.
4. Zelda gives A_1 $f(1)$, A_2 $f(2)$, ..., A_6 $f(6)$ (all mod p). These are all of length $|s|$ by padding with 0's. Also give everyone p (does not count for length).

1. Any 3 have 3 points from $f(x)$ so can find $f(x)$, s.
2. Any 2 have 2 points from $f(x)$. From these two points what can they conclude? NOTHING! If they know $f(1) = 3$ and $f(2) = 7$ and f is degree 2 then the constant term can be **anything** in $\{0,\ldots,p\}$. So they know NOTHING about s.

What Counts

We are concerned about the size of the shares.

1. If Zelda **broadcasts to everyone** a string p, that is not counted towards someone share.
2. If Zelda gives A_1 a string that nobody else gets then that is A_1's share and that counts.
3. If Zelda gives A_1 and A_2 a string (and they both know its the same string) but nobody else, should that count as the length of the share?

What Counts

We are concerned about the size of the shares.

1. If Zelda **broadcasts to everyone** a string p, that is not counted towards someone share.
2. If Zelda gives A_1 a string that nobody else gets then that is A_1's share and that counts.
3. If Zelda gives A_1 and A_2 a string (and they both know its the same string) but nobody else, should that count as the length of the share? There is no scheme that works that way.

Example

$s = 10100$. We'll use $p = 23$.
(ADDED LATER- TAKING $P = 23$ IS IS INCORRECT!! WILL REVIST THIS POINT IN THIRD SET OF SLIDES ON SEC SHARING.)

1. Zelda picks $a_2 = 8$ and $a_1 = 13$.
2. Zelda forms polynomial $f(x) = 8x^2 + 13x + 20$.
3. Zelda gives A_1 $f(1) = 18$, A_2 $f(2) = 9$, A_3 $f(3) = 16$, A_4 $f(4) = 16$, A_5 $f(5) = 9$, A_6 $f(6) = 18$.

If A_1, A_3, A_4 get together and want to find $f(x)$ hence s.
$f(x) = a_2x^2 + a_1x + s$.
$f(1) = 18$: $a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$f(3) = 16$: $a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
$f(4) = 16$: $a_2 \times 4^2 + a_1 \times 4 + s \equiv 16 \pmod{23}$
3 linear equations in, 3 variable, over mod 23 can be solved.
Note: Only need constant term s but can get all coeffs.

What if Two Get Together?

What if A_1 and A_3 get together:
$f(1) = 18$: $a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$f(3) = 16$: $a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
Can they solve these to find s **Discuss**.

What if Two Get Together?

What if A_1 and A_3 get together:
$f(1) = 18$: $a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$f(3) = 16$: $a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
Can they solve these to find s **Discuss**.

No. However, can they use these equations to eliminate some values of s? **Discuss**.

What if Two Get Together?

What if A_1 and A_3 get together:
$f(1) = 18$: $a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$f(3) = 16$: $a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
Can they solve these to find s **Discuss**.

No. However, can they use these equations to eliminate some values of s? **Discuss**.

No. ANY s is consistent. If you pick a value of s, you then have two equations in two variables that can be solved.

What if Two Get Together?

What if A_1 and A_3 get together:
$f(1) = 18$: $a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$f(3) = 16$: $a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
Can they solve these to find s **Discuss**.

No. However, can they use these equations to eliminate some values of s? **Discuss**.

No. ANY s is consistent. If you pick a value of s, you then have two equations in two variables that can be solved.

Important: Information-Theoretic Secure: if A_1 and A_3 meet they learn NOTHING. If they had big fancy supercomputers they would still learn NOTHING.

A Note About Linear Equations

The three equations below, over mod 23, can be solved:

$a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$

$a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$

$a_2 \times 4^2 + a_1 \times 4 + s \equiv 16 \pmod{23}$

Could we have solved this had we used mod 24?

VOTE

1. YES
2. NO

A Note About Linear Equations

The three equations below, over mod 23, can be solved:
$a_2 \times 1^2 + a_1 \times 1 + s \equiv 18 \pmod{23}$
$a_2 \times 3^2 + a_1 \times 3 + s \equiv 16 \pmod{23}$
$a_2 \times 4^2 + a_1 \times 4 + s \equiv 16 \pmod{23}$

Could we have solved this had we used mod 24?
VOTE

1. YES
2. NO

These equations, Don't know, but in general, NO
Need a domain where every number has a mult inverse.
Over mod p, p primes, all numbers have mult inverses.
Over mod 24, even numbers do not have mult inverse.

Subtle Point about Length p

You may have noticed the following oddness:

1. I said **pick** $p \sim 2^{|s|}$.
2. When $s = 10100$ I picked $p = 23$.

Subtle Point about Length p

You may have noticed the following oddness:

1. I said **pick** $p \sim 2^{|s|}$.
2. When $s = 10100$ I picked $p = 23$.

Let $s \in \{0,1\}^n$. So how to best pick prime p?

Subtle Point about Length p

You may have noticed the following oddness:

1. I said **pick** $p \sim 2^{|s|}$.
2. When $s = 10100$ I picked $p = 23$.

Let $s \in \{0,1\}^n$. So how to best pick prime p?

1. Need prime p such that the string s **interpreted as a number in binary** is in $\{0, \ldots, p-1\}$.
2. Want smallest such prime p.
3. p a prime $\geq 2^{|s|}$ always works.
4. Often can use a smaller prime.
5. $s = 10100$. Need a prime such that $20 \in \{0, \ldots, p-1\}$.
   $p = 23$ is smallest.
6. $s = 11111$. Need a prime such that $31 \in \{0, \ldots, p-1\}$.
   $p = 37$ is smallest.

Threshold Secret Sharing With Polynomials: Ref

Due to Adi Shamir

How to Share a Secret
Communication of the ACM
Volume 22, Number 11
1979

Threshold Secret Sharing With Polynomials

Zelda wants to give strings to $A_1, \ldots, A_m$ such that

Any t of $A_1, \ldots, A_m$ can find s. Any $t-1$ learn **NOTHING**.

1. Secret s. Zelda picks prime $p \sim 2^{|s|}$, Zelda works mod p.
2. Zelda gen rand $a_{t-1}, \ldots, a_1 \in \{0, \ldots, p-1\}$
3. Zelda forms polynomial $f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + s$.
4. For $1 \le i \le m$ Zelda gives A_i $f(i)$ mod p.

We Used Polynomials. Could Use...

What did we use about degree $t-1$ polynomials?

1. t points determine the polynomial (we need constant term).
2. $t-1$ points give **no information** about constant term.

Could do geometry over $\mathbb{Z}_p^3$. A **Plane** in $\mathbb{Z}_p^3$ is:

$$\{(x,y,z) : ax+by+cz=d\}$$

1. 3 points in $\mathbb{Z}_p^3$ determine a plane.
2. 2 points in $\mathbb{Z}_p^3$ give **no information** about d.

This approach is due to George Blakely, **Safeguarding Cryptographic Keys**, **International Workshop on Managing Requirements**, **Vol 48**, **1979**.

We will not do secret sharing this way, though one could.

We Used Polynomials. Could Use...

We won't go into details but there are two ways to use the **Chinese Remainder Theorem** to do Secret Sharing.

Due to:
C.A. Asmuth and J. Bloom. **A modular approach to key safeguarding. IEEE Transactions on Information Theory Vol 29**, **Number 2**, **208-210**, **1983**.

And Independently

M. Mignotte **How to share a secret**, **Cryptography: Proceedings of the Workshop on Cryptography, Burg Deursetein**, **Volume 149 of Lecture Notes in Computer Science**, **1982**.

Features and Caveats of Poly Method

Imagine that you've done (t, m) secret sharing with polynomial, $p(x)$. So for $1 \leq i \leq m$, A_i has $f(i)$.

1. **Feature:** If more people come FINE- can extend to $(t, m + a)$ by giving A_{m+1}, $f(m + 1)$, ..., A_{m+a}, $f(m + a)$.
2. **Caveat:** If $m > p$ then you run out of points to give people. There are ways to deal with this, but we will not bother. We will always assume $m < p$.