Computational Threshold
Secret Sharing
Zelda has a secret $s \in \{0, 1\}^n$.

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

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

**Cannot learn the secret** Last lecture this was Info-Theoretic. This lecture we consider comp-theoretic.
Computational Threshold Secret Sharing: Shorter Shares
Info-Theoretic: Shares are $\geq n$

Info-theoretic $(t, L)$-Secret Sharing.

If $A_t$ has a share of length $n - 1$ then $A_1, \ldots, A_{t-1}$ CAN learn something (so NOT info-theoretic security).

$A_1, \ldots, A_{t-1}$ do the following:

- $CAND = \emptyset$. $CAND$ will be set of Candidates for $s$.
- For $x \in \{0, 1\}^{n-1}$ (go through ALL shares $A_t$ could have)
  - $A_1, \ldots, A_{t-1}$ pretend $A_t$ has $x$ and deduce candidates secret $s'$
  - $CAND := CAND \cup \{s'\}$

Secret is in $CAND$. $|CAND| = 2^{n-1} < 2^n$. So we have eliminated many strings from being the $s$
Are Shorter Shares Ever Possible?

If we demand info-security then everyone gets a share $\geq n$. What if we only demand comp-security?

VOTE

1. Can get shares $< \beta n$ with a hardness assumption.
2. Even with hardness assumption REQUIRES shares $\geq n$. 
Are Shorter Shares Ever Possible?

If we demand info-security then everyone gets a share \( \geq n \).

What if we only demand comp-security?

\textbf{VOTE}

1. Can get shares \( < \beta n \) with a hardness assumption.
2. Even with hardness assumption \textbf{REQUIRES} shares \( \geq n \).

\textbf{Can get shares} \( < \beta n \) \textbf{with a hardness assumption}.

Will do that later.
Recall

Threshold Secret Sharing: Information-Theoretic

1. Secret is $s \in \{0, 1\}^n$.
2. $(t, L)$: $t$ people can find $s$, but $t - 1$ cannot.
3. There is a $(t, L)$-scheme where all gets a share of size $n$.
4. There is no scheme where someone gets a share of size $< n$.

That is for **Information-Theoretic Security**.
What if we settle for **Computational Security**?
Recall

Threshold Secret Sharing: Information-Theoretic

1. Secret is $s \in \{0, 1\}^n$.
2. $(t, L)$: $t$ people can find $s$, but $t - 1$ cannot.
3. There is a $(t, L)$-scheme where all gets a share of size $n$.
4. There is no scheme where someone gets a share of size $< n$.

That is for Information-Theoretic Security. What if we settle for Computational Security?

Promise to you: No more Punking
Review of an Aspect of Private Key Crypto

For ciphertext only:

1. Shift is crackable if text is long
2. Affine is crackable if text is long
3. Vig is crackable if text is long compared to the key
4. Matrix is crackable if text is long compared to the key
   (actually I do not know if this is true)

Is there an encryption system where the key is shorter than the text and the system is computationally secure?
Need to define terms first.
Def: Let $0 < \alpha \leq 1$. An $\alpha$-Symm Enc. System ($\alpha$-SES) is a three tuple of functions $(GEN, ENC, DEC)$ where

1. $GEN$ takes $n$ and GENerates $k \in \{0, 1\}^{\alpha n}$.
2. $ENC$ takes $k \in \{0, 1\}^{\alpha n}$ and $m \in \{0, 1\}^n$, outputs $c \in \{0, 1\}^n$. ($ENC$ ENCrypts $m$ with key $k$. We denote $ENC_k(m)$.)
3. $DEC$ takes $k \in \{0, 1\}^{\alpha n}$ and $c \in \{0, 1\}^n$ and outputs $m \in \{0, 1\}^n$ such that $DEC_k(ENC_k(m)) = m$. So $DEC$ DECrypts.

Def: We will not define security formally here; however, intuitively Eve cannot learn $m$ from $c$. We are concerned with ciphertext only.

Note: $\alpha$-SES encrypts a length $n$ message by a length $n$ ciphertext.
Psuedorandom Generators

Def: (Informal) A pseudorandom gen maps a short seed to a long sequence that a limited Eve cannot distinguish from random.

Idea: Do the one-time-pad but with a psuedorandom sequence.
Discuss

PROS and CONS
Psuedorandom Generators

Def: (Informal) A pseudorandom gen maps a short seed to a long sequence that a limited Eve cannot distinguish from random.

Idea: Do the one-time-pad but with a psuedorandom sequence.

Discuss

PROS and CONS

CON: All Powerful Even can crack it!
Psuedorandom Generators

**Def:** (Informal) A pseudorandom gen maps a short seed to a long sequence that a limited Eve cannot distinguish from random.

**Idea:** Do the one-time-pad but with a psuedorandom sequence.

**Discuss**

**PROS** and **CONS**

**CON:** All powerful Even can crack it!

**PRO:** Limited Eve cannot crack it!
Psuedorandom Generators

**Def:** (Informal) A pseudorandom gen maps a short seed to a long sequence that a limited Eve cannot distinguish from random.

**Idea:** Do the one-time-pad but with a pseudorandom sequence.

**Discuss**

**PROS** and **CONS**

**CON:** All Powerful Eve can crack it!

**PRO:** Limited Eve cannot crack it!

**PRO:** Can Actually use!
BBS Generator

Blum-Blum-Shub psuedo-random Generator. Recall that LSB means *Least Significant Bit*.

1. Seed: $p, q$ primes, $x_0 \in \mathbb{Z}_{N=pq}$. $p, q \equiv 3 \pmod{4}$.
2. Sequence:

$$
\begin{align*}
    x_1 &= x_0^2 \mod N & b_1 &= \text{LSB}(x_1) \\
    x_2 &= x_1^2 \mod N & b_2 &= \text{LSB}(x_2) \\
    \vdots & \vdots & \vdots & \vdots & \vdots \\
    x_L &= x_{L-1}^2 \mod N & b_L &= \text{LSB}(x_L)
\end{align*}
$$

$r = b_1 \cdots b_L$ is pseudo-random.

**Known:** Assuming Factoring is hard, this is $\frac{1}{2}$-SES. If $L$ is twice the length of seed, and seed long enough, then secure.
Example of $\frac{1}{2}$-SES

Name of this System BBS-Psuedo 1-time Pad, or BBS-POTP.

1. **GEN:** $k = (p, q, x_0)$. $|k| = \frac{n}{2}$. $p, q$ prime $p \equiv q \equiv 3 \pmod{4}$.

2. **ENC:** Use $k$ to BBS-gen $b_1, \ldots, b_n$. $m \in \{0, 1\}^n$.

   \[ ENC_k(m_1, \ldots, m_n) = (m_1 \oplus b_1, \ldots, m_n \oplus b_n). \]

3. **DEC:** Bob can use $k = (p, q, x_0)$ to find $b_0, \ldots, b_n$, and decode.

**Known:** Assume determining if a number is in $SQ_N$ is hard. For large enough $n$ this is secure.

**Note:** Message is twice as long as key, so this is $\frac{1}{2}$-SES.

**Note:** Will not be using this particular SES but have it here as a concrete example.
Blum-Goldwasser (BG) vs BBS-POPT

1. BG is a Public Key Cryptosystem. Bob sends Alice stuff from which she can reconstruct the pseudo-one-time-pad and then use it.

2. BBS-POPT is a Private Key Cryptosystem. Alice and Bob both have a way to generate a long string from a short one. They meet and determine a short string, and both use it to generate a long one. They use the long string for the pad. Easier than real 1-time pad, though not as secure.
Short Shares

**Thm:** Assume there exists an $\alpha$-SES. Assume that for message of length $n$, it is secure. Then, for all $1 \leq t \leq L$ there is a $(t, L)$-scheme for $|s| = n$ where each share is of size $\frac{n}{t} + \alpha n$.

1. Zelda does $k \leftarrow GEN(n)$. Note $|k| = \alpha n$.
2. $u = ENC_k(s)$. Let $u = u_0 \cdots u_{t-1}$, $|u_i| \sim \frac{n}{t}$.
3. Let $p \sim 2^{n/t}$. Zelda forms poly over $\mathbb{Z}_p$:

   $$f(x) = u_{t-1}x^{t-1} + \cdots + u_1x + u_0$$

4. Let $q \sim 2^{\alpha n}$. Zelda forms poly over $\mathbb{Z}_q$ by choosing $r_{t-1}, \ldots, r_1 \in \{0, \ldots, q - 1\}$ at random and then:

   $$g(x) = r_{t-1}x^{t-1} + \cdots + r_1x + k$$

5. Zelda gives $A_i$, $(f(i), g(i))$. Length: $\sim \frac{n}{t} + \alpha n$. 
Length and Recovery

**Length:**
1. $f(i) \in \mathbb{Z}_p$ where $p \sim 2^{n/t}$, so $|f(i)| \sim \frac{n}{t}$.
2. $g(i) \in \mathbb{Z}_q$ where $q \sim 2^{\alpha n}$, so $|g(i)| \sim \alpha n$.

**Recovery:** If $t$ get together:
1. Have $t$ points of $f$, can get $u_{t-1}, \ldots, u_0$, hence $u$.
2. $u = ENC_k(s)$. So need $k$.
3. Have $t$ points of $g$, can get $k$.
4. With $k$ and $u$ can get $s = DEC_k(u)$. 
Length and Recovery

**Length:**
1. $f(i) \in \mathbb{Z}_p$ where $p \sim 2^{n/t}$, so $|f(i)| \sim \frac{n}{t}$.
2. $g(i) \in \mathbb{Z}_q$ where $q \sim 2^{\alpha n}$, so $|g(i)| \sim \alpha n$.

**Recovery:** If $t$ get together:
1. Have $t$ points of $f$, can get $u_{t-1}, \ldots, u_0$, hence $u$.
2. $u = ENC_k(s)$. So need $k$.
3. Have $t$ points of $g$, can get $k$.
4. With $k$ and $u$ can get $s = DEC_k(u)$.

If $t - 1$ get together:
Length and Recovery

Length:
1. $f(i) \in \mathbb{Z}_p$ where $p \sim 2^{n/t}$, so $|f(i)| \sim \frac{n}{t}$.
2. $g(i) \in \mathbb{Z}_q$ where $q \sim 2^{\alpha n}$, so $|g(i)| \sim \alpha n$.

Recovery: If $t$ get together:
1. Have $t$ points of $f$, can get $u_{t-1}, \ldots, u_0$, hence $u$.
2. $u = ENC_k(s)$. So need $k$.
3. Have $t$ points of $g$, can get $k$.
4. With $k$ and $u$ can get $s = DEC_k(u)$.

If $t - 1$ get together:
Next Slide
The scheme I showed you is due to Hugo Krawczyk, *Secret Sharing Made Short*, *Advances in Crypto – CRYPTO 1993 Lecture notes in computer science 773, 1993*

However, the proof of security was not quite right.

Mihir Bellar and Phillip Rogaway wrote a paper that proved Krawczyk’s protocol secure by adding a condition to the $\alpha$-SES. We omit since its complicated.

Can we do better than $\frac{n}{t} + \alpha n$?

**Ill Formed Question:** Can we do better than $\frac{n}{t} + \alpha n$?

The question is not quite right – if we have a smaller $\alpha$ can do better.

**Better Question:** Assume there is an $\alpha$-SES. Is the following true:

*For all $0 < \beta < 1$ there exists an $(t, L)$ secret sharing scheme where everyone gets $\frac{n}{t} + \beta n$.**

Discuss
Can we do better than $\frac{n}{t} + \alpha n$?

Ill Formed Question: Can we do better than $\frac{n}{t} + \alpha n$? The question is not quite right – if we have a smaller $\alpha$ can do better.

Better Question: Assume there is an $\alpha$-SES. Is the following true:

$\text{For all } 0 < \beta < 1 \text{ there exists an } (t, L) \text{ secret sharing scheme where everyone gets } \frac{n}{t} + \beta n.$

Discuss

Can be done by iterating the above construction. Might be HW or Exam.
Breaking the $\frac{n}{t}$ Barrier!

(2, 2): $A, B$ share the secret $s$, $|s| = n$. Computational Secret Sharing, so can make a hardness assumption.

**Question:** Is there a (2, 2) secret sharing scheme where $A$ and $B$ both get a share $\leq \frac{n}{3}$? **Discuss.** Vote!

1. YES! There is such a Scheme.
2. NO! We can prove there is NO such scheme.
3. PUNKED! Bill will shows us a scheme that looks like it works but he’ll be PUNKING US!
4. Unknown to science!
(2, 2): $A, B$ share the secret $s$, $|s| = n$. Computational Secret Sharing, so can make a hardness assumption.

**Question:** Is there a (2, 2) secret sharing scheme where $A$ and $B$ both get a share $\leq \frac{n}{3}$?

**Discuss.** Vote!

1. **YES!** There is such a Scheme.
2. **NO!** We can prove there is NO such scheme.
3. **PUNKED!** Bill will shows us a scheme that looks like it works but he’ll be PUNKING US!
4. **Unknown to science!**

**NO!** We can prove there is NO such scheme.
Theorem: There is no $(2, 2)$-scheme with shares $\frac{n}{3}$.

Proof: Assume there is.

Map $s \in \{0, 1\}^n$ to the ordered pair ($A$’s share, $B$’s share)
$2^n$ elements in the domain.
$2^{n/3} \times 2^{n/3} = 2^{2n/3}$ elements in the co-domain.

Hence exists $s, s' \in \{0, 1\}^n$ that map to same $(a, b)$.
If $A$ gets $a$, and $B$ gets $b$, will not decode uniquely into one secret.

Contradiction!
This Generalizes. Might be on HW or Exam
Computational Threshold
Secret Sharing:
Verifiable S.S.
1. (5, 9) Secret Sharing.
2. The secret is $s$. $s \sim 2^p$. Zelda picks random $r_4, r_3, r_2, r_1$ and forms the polynomial $f(x) = r_4x^4 + r_3x^3 + r_2x^2 + r_1x + s$.
3. For $1 \leq i \leq 9$ Zelda gives $A_i$ the element $f(i)$.
A Scenario

1. (5, 9) Secret Sharing.
2. The secret is $s$. $s \sim 2^p$. Zelda picks random $r_4, r_3, r_2, r_1$ and forms the polynomial $f(x) = r_4x^4 + r_3x^3 + r_2x^2 + r_1x + s$.
3. For $1 \leq i \leq 9$ Zelda gives $A_i$ the element $f(i)$.

$A_2, A_4, A_7, A_8, A_9$ get together. BUT they do not trust each other!

1. $A_2$ thinks that $A_7$ is a traitor!
2. $A_7$ thinks $A_4$ will confuse them just for the fun of it.
3. $A_8$ and $A_9$ got into a knife fight over who proved that the muffin problem always has a rational solution.
4. The list goes on
A Scenario

1. (5, 9) Secret Sharing.
2. The secret is $s$. $s \sim 2^p$. Zelda picks random $r_4, r_3, r_2, r_1$ and forms the polynomial $f(x) = r_4x^4 + r_3x^3 + r_2x^2 + r_1x + s$.
3. For $1 \leq i \leq 9$ Zelda gives $A_i$ the element $f(i)$.

$A_2, A_4, A_7, A_8, A_9$ get together. BUT they do not trust each other!

1. $A_2$ thinks that $A_7$ is a traitor!
2. $A_7$ thinks $A_4$ will confuse them just for the fun of it.
3. $A_8$ and $A_9$ got into a knife fight over who proved that the muffin problem always has a rational solution.
4. The list goes on

Hence we need to VERIFY that everyone is telling the truth. This is called VERIFIABLE secret sharing, or VSS.
First Attempt at \((t, L)\) VSS

1. Secret is \(s\), \(|s| = n\). Zelda finds \(p \sim n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1\), \(f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i f(i)\).
5. Zelda broadcasts \(g^s\) (this does not reveal \(s\)).

**Recover:** Any group of \(t\) can determine \(f\) and hence \(s\).

**Verify:** Once a group has \(s\) they compute \(g^s\) and see if it matches. If so then they **know** they have the correct secret. If no then they **know** someone is a **stinking rotten liar**

\[f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s.\]
First Attempt at \((t, L)\) VSS

1. Secret is \(s\), \(|s| = n\). Zelda finds \(p \sim n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1\), \(f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i f(i)\).
5. Zelda broadcasts \(g^s\) (this does not reveal \(s\)).

**Recover:** Any group of \(t\) can determine \(f\) and hence \(s\).

**Verify:** Once a group has \(s\) they compute \(g^s\) and see if it matches. If so then they know they have the correct secret. If no then they know someone is a stinking rotten liar

1. If verify \(s\) there may still be two liars who cancel out.
2. If do not agree they do not know who the liar was.
3. Does not serve as a deterrent.
Second Attempt at \((t, L)\) VSS

1. Secret is \(s, |s| = n\). Zelda finds \(p \sim 2^n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1, f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i f(i)\).
5. Zelda broadcasts \(g^{f(1)}, \ldots, g^{f(L)}\). (No \(f(i)\) not revealed.)

**Recover:** The usual – any group of \(t\) can blah blah.

**Verify:** If \(A_i\) says \(f(i) = 17\), they can all then check if \(g^{17}\) is what Zelda said \(g^{f(i)}\) is, so can determine if \(A_i\) is truthful.
Second Attempt at \((t, L)\) VSS

1. Secret is \(s\), \(|s| = n\). Zelda finds \(p \sim 2^n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1\), \(f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i\) \(f(i)\).
5. Zelda broadcasts \(g^{f(1)}, \ldots, g^{f(L)}\). (No \(f(i)\) not revealed.)

**Recover:** The usual – any group of \(t\) can blah blah.

**Verify:** If \(A_i\) says \(f(i) = 17\), they can all then check if \(g^{17}\) is what Zelda said \(g^{f(i)}\) is, so can determine if \(A_i\) is truthful.

1. **PRO:** If someone lies they know right away.
2. **CON:** Leaks! If \(f(i) = f(j)\) then \(i, j\) can get together and knowing \(f(i) = f(j)\), SOME info leaks.
3. **CON:** \(L\) strings is a lot.
4. **CON:** If more come then need to update public info.
Third Attempt at \((t, L)\) VSS

1. Secret is \(s\), \(|s| = n\). Zelda finds \(p \sim 2^n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1\), \(f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i f(i)\).
5. Zelda broadcasts \(g^{r_1}, \ldots, g^{r_{t-1}}, g^s, g\) (\(r_i\) not revealed).

**Recover:** The usual – any group of \(t\) can blah blah.

**Verify:** \(A_i\) reveals \(f(i) = 17\). Group computes:

1) \(g^{17}\).
2) \((g^{r_{t-1}})^{i_{t-1}} \times (g^{r_{t-2}})^{i_{t-2}} \times \cdots \times (g^{r_1})^{i_1} \times (g^s)^{i_0} = g^{f(i)}\)

If this is \(g^{17}\) then \(A_i\) is truthful. If not then \(A_i\) is dirty stinking liar.
Third Attempt at \((t, L)\) VSS

1. Secret is \(s\), \(|s| = n\). Zelda finds \(p \sim 2^n\).
2. Zelda finds a generator \(g\) for \(\mathbb{Z}_p\).
3. Zelda picks rand \(r_{t-1}, \ldots, r_1\), \(f(x) = r_{t-1}x^{t-1} + \cdots + r_1x + s\).
4. For \(1 \leq i \leq L\) Zelda gives \(A_i f(i)\).
5. Zelda broadcasts \(g^{r_1}, \ldots, g^{r_{t-1}}, g^s, g\) (\(r_i\) not revealed).

**Recover:** The usual – any group of \(t\) can blah blah.

**Verify:** \(A_i\) reveals \(f(i) = 17\). Group computes:

1) \(g^{17}\).
2) \((g^{r_{t-1}})^{i^{t-1}} \times (g^{r_{t-2}})^{i^{t-2}} \times \cdots \times (g^{r_1})^{i_1} \times (g^s)^{i_0} = g^{f(i)}\)

If this is \(g^{17}\) then \(A_i\) is truthful. If not then \(A_i\) is dirty stinking liar.

1. **PRO:** If someone lies they know right away.
2. **PRO:** Serves as a deterrent.
3. **PRO:** Zelda is communicating only \(t\) strings.
4. **PRO:** If more people come do not need to update public info.
5. **CON:** Security – see next slide.
Security and References

The scheme above for VSS is by Paul Feldman.

A Practical Scheme for non-interactive Verifiable Secret Sharing
28th Conference on Foundations of Computer Science (FOCS)
1987

They give proof of security based on zero-knowledge protocols which are themselves based on blah blah.
More Can Be Said About Secret Sharing

arXiv is a website where Academics in Math, Comp Sci, and Physics post papers. How many of those papers are on Secret Sharing?

Vote

1. Between 0 and 100
2. Between 100 and 1000
3. Between 1000 and 10,000
4. Over 10,000

Answer

About 14,500 so over 10,000.
More Can Be Said About Secret Sharing

arXiv is a website where Academics in Math, Comp Sci, and Physics post papers. How many of those papers are on Secret Sharing?

Vote

1. Between 0 and 100
2. Between 100 and 1000
3. Between 1000 and 10,000
4. Over 10,000

Answer About 14,500 so over 10,000.