Threshold Secret Sharing: Information-Theoretic

October 17, 2019
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** will be info-theoretic. Even if $t - 1$ people have big fancy supercomputers they cannot learn $s$. We will later look at comp-security.
**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

A_1, A_2, A_3, A_4 such that

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 \(s = s_1 s_1 s_3 s_4\) 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 $s = s_1 s_1 s_3 s_4$ 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 $s$. 

$2.1 A_1$ 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!
An Attempt at (4, 4)-Secret Sharing

1. Zelda breaks $s$ up into $s = s_1s_1s_3s_4$ 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 \(s = s_1 s_1 s_3 s_4\) 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 $s = s_1 s_1 s_3 s_4$ 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 $BLAH_{i,j,k}$.

3. Giving out the shares tells each triple NOTHING they did not already know.

If $A_i, A_j, A_k$ 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 $BLAH_{i,j,k}$.
3. Giving out the shares tells each triple NOTHING they did not already know.

If $A_i, A_j, A_k$ 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 \( BLAH_{i,j,k} \).
3. Giving out the shares tells each triple NOTHING they did not already know.

If \( A_i, A_j, A_k \) 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 $r_1, r_2, r_3 \in \{0, 1\}^n$. 

Can Recover the Secret

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

Easy to see that if a 3 get together they learn NOTHING
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 $A_4 s_4 = s \oplus r_1 \oplus r_2 \oplus r_3$

Can Recover the Secret

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

Easy to see that if a 3 get together they learn NOTHING
Zelda gives out shares of the secret

1. Secret \( s \in \{0, 1\}^n \). Zelda generates 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

For each $1 \leq i < j \leq 4$
(2, 4)-Secret Sharing using Random Strings

For each $1 \leq i < j \leq 4$

1. Zelda generates random $r \in \{0, 1\}^n$. 
(2, 4)-Secret Sharing using Random Strings

For each $1 \leq i < j \leq 4$

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

$A_i$, $A_j$ 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.
For each $1 \leq i < j \leq 4$

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

$A_i$ and $A_j$ can recover the secret.

$A_i$ takes $((i,j), r)$ and just uses $r$.

$A_j$ takes $((i,j), r \oplus s)$ and just uses $r \oplus s$.

They both compute $r \oplus r \oplus s = s$. Easy to see that one person learns NOTHING.
(2, 4)-Secret Sharing using Random Strings

For each $1 \leq i < j \leq 4$

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

$A_i$, $A_j$ 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

For each $1 \leq i < j \leq 4$

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

$A_i$, $A_j$ 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
(L, L)-Random String Method

People: A₁, ..., Aₖ. Secret s.
(\(L, L\))-Random String Method

People: \(A_1, \ldots, A_L\). Secret \(s\).

1. Zelda gen rand \(r_1, \ldots, r_{L-1}\).
(L, L)-Random String Method

People: A₁, ..., A_L. Secret s.

1. Zelda gen rand r₁, ..., r_{L−1}.
2. A₁ get r₁
   A₂ get r₂
   ...
   A_{L−1} gets r_{L−1}
A_L gets s ⊕ r₁ ⊕ ⋯ ⊕ r_{L−1}

We use this as building block for gen case.
(L, L)-Random String Method

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

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

2. $A_1$ get $r_1$
   $A_2$ get $r_2$
   
   $\vdots$

   $A_{L-1}$ gets $r_{L-1}$
   $A_L$ gets $s \oplus r_1 \oplus \cdots \oplus r_{L-1}$

3. If they all get together they will XOR all their strings to get $s$
$(L, L)$-Random String Method

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

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

2. $A_1$ get $r_1$
   $A_2$ get $r_2$
   
   $\vdots$
   
   $A_{L-1}$ gets $r_{L-1}$
   $A_L$ gets $s \oplus r_1 \oplus \cdots \oplus r_{L-1}$

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, L) Secret Sharing

People: $A_1, \ldots, A_L$. $S_1, \ldots, S_m \subseteq \{A_1, \ldots, A_L\}$ are all the sets of size $t$. ($m = \binom{L}{t}$).

1. For every $1 \leq j \leq m$ 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 subset can get the secret.

**PRO:** Can always do Threshold Secret Sharing.

**CON:** You are giving people A LOT of strings!
How Many Strings Does $A_i$ Get 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
How Many Strings Does $A_i$ Get 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**

$$\binom{9}{4} = 126 \text{ strings}$$
How Many Strings Does $A_i$ Get in $(L/2, L)$-Secret Sharing?

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

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

Equivalent to:

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

How many sets? Discuss
How Many Strings Does $A_i$ Get in $(L/2, L)$-Secret Sharing?

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

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

Equivalent to:

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

How many sets? Discuss

$$\left(\frac{L - 1}{2} - 1\right) \sim \frac{2^L}{\sqrt{L}} \text{ strings}$$

That's A LOT of Strings!
Can We Reduce The Number of Strings for \((L/2, L)\)?

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

**VOTE**

1. Requires roughly \(2^L\) strings.
2. \(O(\beta^L)\) strings for some \(1 < \beta < 2\) but not poly.
3. \(O(L^a)\) strings for some \(a > 1\) but not linear.
4. \(O(L)\) strings but not sublinear.
5. \(O(\log L)\) strings but not constant.
6. \(O(1)\) strings.

You can always do this with everyone getting 1 string.
Can We Reduce The Number of Strings for \((L/2, L)\)?

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

**VOTE**

1. Requires roughly \(2^L\) strings.
2. \(O(\beta^L)\) strings for some \(1 < \beta < 2\) but not poly.
3. \(O(L^a)\) strings for some \(a > 1\) but not linear.
4. \(O(L)\) strings but not sublinear.
5. \(O(\log L)\) strings but not constant.
6. \(O(1)\) strings.

You can always do this with everyone getting 1 string.
Can We Reduce The Number of Strings for \((L/2, L)\)?

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

**VOTE**

1. Requires roughly \(2^L\) strings.
2. \(O(\beta^L)\) strings for some \(1 < \beta < 2\) but not poly.
3. \(O(L^a)\) strings for some \(a > 1\) but not linear.
4. \(O(L)\) strings but not sublinear.
5. \(O(\log L)\) strings but not constant.
6. \(O(1)\) strings.

You can always do this with everyone getting 1 string.

I know what you are thinking: LOOOONG string. No.
Can We Reduce The Number of Strings for \((L/2, L)\)?

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

**VOTE**

1. Requires roughly \(2^L\) strings.
2. \(O(\beta^L)\) strings for some \(1 < \beta < 2\) but not poly.
3. \(O(L^a)\) strings for some \(a > 1\) but not linear.
4. \(O(L)\) strings but not sublinear.
5. \(O(\log L)\) strings but not constant.
6. \(O(1)\) 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
We do $(3, 6)$-Secret Sharing.

1. Secret $s$. Zelda picks prime $p \sim s$, Zelda works mod $p$.
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 $A_1 f(1), A_2 f(2), \ldots, A_6 f(6)$ (all mod $p$). These are all of length $\sim |s|$.

1. Any 3 have 3 points from $f(x)$ so can find $f(x)$, $s$.
2. Any 2 have 2 points from $f(x)$. Constant term ($s$) anything!
Example

$s = 20$. We’ll use $p = 23$.

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**

**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 number do not have mult inverse.
Due to Adi Shamir

**How to Share a Secret**

*Communication of the ACM*

*Volume 22, Number 11*

*1979*
Zelda wants to give strings to $A_1, \ldots, A_L$ such that
Any $t$ of $A_1, \ldots, A_L$ can find $s$. Any $t - 1$ learn NOTHING.

1. Secret $s$. Zelda picks prime $p \sim s$, Zelda works mod $p$.
2. Zelda gen rand $a_{t-1}, \ldots, a_1 \in \{0, \ldots, p - 1\}$
3. Zelda forms polynomial $f(x) = a_{t-1}x^{t-1} + \cdots + a_1x + s$.
4. For $1 \leq i \leq L$ Zelda gives $A_i$ $f(i)$ mod $p$.

1. Any $t$ have $t$ points of $f(x)$ so can find $f(x)$ and $s$.
2. Any $t - 1$ have $t - 1$ points of $f(x)$. Constant term ($s$) could be anything!
We Used Polynomials. Could Use…

What did we use about degree \( t - 1 \) polynomials?

1. \( t \) points determine a 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 \( d \).


We will not do secret sharing this way, though one could.
We won’t go into details but there are two ways to use the **Chinese Remainder Theorem** to do Secret Sharing.

Due to:

And Independently
Imagine that you’ve done \((t, L)\) secret sharing with polynomial, \(p(x)\). So for \(1 \leq i \leq L\), \(A_i\) has \(f(i)\).

1. **Feature:** If more people come FINE- can extend to \((t, L + a)\) by giving \(A_{L+1}, f(L+1), \ldots, A_{L+a}, f(L+a)\).

2. **Caveat:** If \(L > p\) then you run out of points to give people. We will always assume \(L < p\).

3. **Caveat:** If \(L > p\) there are still ways to do this, but we won’t get into that.