We should share our secrets
Shamir secret sharing: how it works and how to implement it

Daan Sprenkels
hello@dsprenkels.com

Radboud University Nijmegen

28 December 2017
Who am I?

- Student at Radboud University Nijmegen
- Bachelor in Chemistry
- Currently studying Cyber Security
- On a regular day I implement elliptic curve crypto\(^1\)

The others:
- Peter Schwabe\(^2\) (@cryptojedi)
- Sean Moss-Pultz\(^3\) (@moskovich)

---

\(^1\)The meaning of “crypto” is cryptography, not cryptocurrency!
\(^2\)Radboud University
\(^3\)Bitmark Inc. (https://bitmark.com)
Don’t roll your own crypto

and also don’t implement your own crypto

Daan Sprenkels

We should share our secrets

28 December 2017
“Don’t roll your own crypto”
“Don’t roll your own crypto”

“and also don’t implement your own crypto”
Part I: Crypto theory
  What is secret sharing?
  How does it work?

Part II: Crypto implementation
  How to encode our values
  Solving integrity
  Side channel resistance
  Performance and bitslicing
HOW HARD IS MY TALK?

CRYPTO THEORY

IMPLEMENTATION

QUESTIONS

NOW

easy

less easy

minutes

45
Part I: crypto theory
Problem statement

- How to backup your secrets (wallet keys, passwords, etc.)?
Problem statement

- How to backup your secrets (wallet keys, passwords, etc.)?
- Need to trust a single entity
Problem statement

- How to backup your secrets (wallet keys, passwords, etc.)?
- Need to trust a single entity
- How to split up our trust?
Solving our problem

1. Cut my key into pieces
   Secret message $m = A||B||C$.
   Distribute $A$, $B$, $C$.

2. Use one-time-pad construction?
   Generate random $A$, $B$.
   Choose $C = m \oplus A \oplus B$. 
Solving our problem

1. Cut my key into pieces
   Secret message $m = A||B||C$.
   Distribute $A, B, C$.
   *Bad security!*
1. Cut my key into pieces
   Secret message $m = A || B || C$.
   Distribute $A$, $B$, $C$.
   Bad security!

2. Use one-time-pad construction?
   Generate random $A$, $B$
   Choose $C = m \oplus A \oplus B$. 
Solving our problem

1. Cut my key into pieces
   Secret message \( m = A | | B | | C \).
   Distribute \( A, B, C \).
   *Bad security!*

2. Use one-time-pad construction?
   Generate random \( A, B \)
   Choose \( C = m \oplus A \oplus B \).
   Restore by computing \( m' = A \oplus B \oplus C \)
Solving our problem

1. Cut my key into pieces
   Secret message $m = A||B||C$.
   Distribute $A, B, C$.
   Bad security!

2. Use one-time-pad construction?
   Generate random $A, B$
   Choose $C = m \oplus A \oplus B$.
   Restore by computing $m' = A \oplus B \oplus C = A \oplus B \oplus (m \oplus A \oplus B)$
Solving our problem

1. Cut my key into pieces
   Secret message \( m = A \| B \| C \).
   Distribute \( A, B, C \).
   *Bad security!*

2. Use one-time-pad construction?
   Generate random \( A, B \)
   Choose \( C = m \oplus A \oplus B \).
   Restore by computing \( m' = A \oplus B \oplus C = A \oplus B \oplus (m \oplus A \oplus B) \)
Solving our problem

1. Cut my key into pieces
   Secret message $m = A||B||C$.
   Distribute $A, B, C$.
   *Bad security!*

2. Use one-time-pad construction?
   Generate random $A, B$
   Choose $C = m \oplus A \oplus B$.
   Restore by computing $m' = A \oplus B \oplus C = m$
Solving our problem

1. Cut my key into pieces
   Secret message $m = A||B||C$.
   Distribute $A, B, C$.
   *Bad security!*

2. Use one-time-pad construction?
   Generate random $A, B$
   Choose $C = m \oplus A \oplus B$.
   Restore by computing $m' = A \oplus B \oplus C = m$
   *Needs all pieces to restore the secret*
A better solution

Shamir secret sharing

- Published almost 40 years ago by Adi Shamir
- Threshold scheme \((n, k)\)
- “Provably secure”
A better solution

Shamir secret sharing

- Published almost 40 years ago by Adi Shamir
- Threshold scheme \((n, k)\)
- “Provably secure” Information-theoretically secure
Example with \((n, k) = (5, 4)\)
Example with \((n, k) = (5, 4)\)
Example with \((n, k) = (5, 4)\)
Example with \((n, k) = (5, 4)\)
Example with \((n, k) = (5, 4)\)
Example with \((n, k) = (5, 4)\)
How does the math work?

Given parameters \((n, k)\) and message \(m\):

Construct a polynomial of degree \(k - 1\):

\[
p(x) = a_{k-1}x^{k-1} + \ldots + a_1x + m
\]  

(1)

With coefficients \(a_i\) randomly generated.
How does the math work?

Given parameters \((n, k)\) and message \(m\):
Construct a polynomial of degree \(k - 1\):

\[
p(x) = a_{k-1}x^{k-1} + \ldots + a_1x + m
\]  

(1)

With coefficients \(a_i\) randomly generated.

Evaluate \(n\) points on the polynomial to get shares \(s_i\):

\[
s_1 = (1, p(1))
\]
\[
s_2 = (2, p(2))
\]
\[\vdots\]
\[
s_n = (n, p(n))
\]
How does the math work?

Find \( p(x) = a_{k-1}x^{k-1} + \ldots + a_1x + m \) such that all \( s_i \) are on \( p(x) \).

Solve for \( m \):

\[
\begin{align*}
    a_{k-1}x_1^{k-1} + \ldots + a_1x_1 + m &= y_1 \\
    a_{k-1}x_2^{k-1} + \ldots + a_1x_2 + m &= y_2 \\
    a_{k-1}x_3^{k-1} + \ldots + a_1x_3 + m &= y_3 \\
    \ldots \\
    a_{k-1}x_k^{k-1} + \ldots + a_1x_k + m &= y_k
\end{align*}
\]
How does the math work?

Find \( p(x) = a_{k-1}x^{k-1} + \ldots + a_1x + m \) such that all \( s_i \) are on \( p(x) \).

Solve for \( m \):

\[
\begin{align*}
    a_{k-1}x_1^{k-1} + \ldots + a_1x_1 + m &= y_1 \\
    a_{k-1}x_2^{k-1} + \ldots + a_1x_2 + m &= y_2 \\
    a_{k-1}x_3^{k-1} + \ldots + a_1x_3 + m &= y_3 \\
    \ldots \\
    a_{k-1}x_k^{k-1} + \ldots + a_1x_k + m &= y_k
\end{align*}
\]

*Use Lagrange interpolation for solving*
We should share our secrets

\[ p(x) = 4x^2 - 25x + 42 \]

\[ m = 42 \]
We should share our secrets
We should share our secrets

(0, 26)
(1, 21)
(4, 6)
We should share our secrets
Example: combining shares

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
    a_2 x_1^2 + a_1 x_1 + m &= y_1 \\
    a_2 x_2^2 + a_1 x_2 + m &= y_2 \\
    a_2 x_3^2 + a_1 x_3 + m &= y_3
\end{align*}
\]
Example: combining shares

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
1^2 a_2 + a_1 + m &= 21 \\
4^2 a_2 + 4a_1 + m &= 6 \\
2^2 a_2 + 2a_1 + m &= 8
\end{align*}
\]
Example: combining shares

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
1^2 a_2 + a_1 + m &= 21 \\
4^2 a_2 + 4a_1 + m &= 6 \\
2^2 a_2 + 2a_1 + m &= 8
\end{align*}
\]

\[ m = 42 \]
All good?
All good?

- Information-theoretically secure
All good?

- Information-theoretically secure *for confidentiality*
- Not really secure for *integrity*
We should share our secrets

$m = 42$

$p(x) = -7x + 42$
We should share our secrets
(1, 35)
(1, 24)
(2, 28)
m' = 20
Solving integrity

Solutions:

- Randomize $x$-values
- Only share random secrets
Part II: implementation
Bitmark Inc. asks us for a Shamir secret sharing library.

- Secure for integrity ($\geq 128$ bits)
- Side channel resistant (timing, cache-timing)
- Portable to any platform
Bitmark Inc. asks us for a Shamir secret sharing library.

- Secure for integrity ($\geq 128$ bits)
- Side channel resistant (timing, cache-timing)
- Portable to any platform

Existing libraries:
- ssss
- gfshare
Bitmark Inc. asks us for a Shamir secret sharing library.

- Secure for integrity ($\geq 128$ bits)
- Side channel resistant (timing, cache-timing)
- Portable to any platform

Existing libraries:

- ssss
- gfshare

Both do not meet our requirements
Implementation challenges

On to implement it ourselves...

1. How to encode our values?
2. How to fix our integrity problem?
3. How to prevent side channels?
4. How to make it fast?
1. How to encode our values?

Options:

- Integers modulo large prime?
- Other finite field?

\footnote{For the maths people, we are using $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$}
1. How to encode our values?

Options:
- Integers modulo large prime?
- Other finite field?

Piece up the secret in bytes and map them to $\mathbb{F}_{2^8}$ (note\(^1\))
- Fast arithmetic
- Can secret-share every byte independently

---

\(^1\)For the maths people, we are using $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$
2. Solving integrity

Use hybrid encryption:

\[
\text{random key} \quad \text{secret} \\
\downarrow \quad \downarrow \\
\text{Shamir} \quad \text{encrypt} \\
\downarrow \quad \downarrow \\
\text{keyshare||ciphertext}
\]
2. Solving integrity

Use hybrid encryption:

```
secret

Shamir  decrypt

keyshare || ciphertext

key
```
3. How to prevent side channel attacks?

Rules to protect against side channels\(^2\):

1. No branching (if, &&, ||, etc.)

\(^2\)In *software*! Hardware implementations are a whole other story.
3. How to prevent side channel attacks?

Rules to protect against side channels:\(^2\):

1. No branching (if, &&, ||, etc.)
2. No secret-dependent lookups (\(y = \text{table}[\text{key}[i]]\);)

\(^2\)In *software!* Hardware implementations are a whole other story.
3. How to prevent side channel attacks?

Rules to protect against side channels\(^2\):

1. No branching (if, &&, ||, etc.)
2. No secret-dependent lookups (\(y = \text{table}[\text{key}[i]]\));
3. No variable-time instructions (\(\text{div, mul} \ [2], \) etc.)

\(^2\)In *software!* Hardware implementations are a whole other story.
4. Performance through bitslicing

![Diagram showing the process of bitslicing with bytes and registers]

Daan Sprenkels

We should share our secrets

28 December 2017 23 / 35
4. Performance through bitslicing

- Working in bytes ⇒ need only 8 registers per byte
- Implement algorithm in logic circuits
Example: *Adding two bytes in parallel*
4. Performance through bitslicing

- Working in bytes → need only 8 registers per byte
- Implement algorithm in logic circuits
- 32-bit platform? 32× parallel computation
4. Performance through bitslicing

- Working in bytes ⇒ need only 8 registers per byte
- Implement algorithm in logic circuits
- 32-bit platform? 32× parallel computation = performance :)

Daan Sprenkels
We should share our secrets
28 December 2017
4. Performance through bitslicing

- Working in bytes $\Rightarrow$ need only 8 registers per byte
- Implement algorithm in logic circuits
- 32-bit platform? $32 \times$ parallel computation $=$ performance :)
- Scales to 64-bit, avx{, 2, 512}, etc. :)

Daan Sprenkels
We should share our secrets
28 December 2017
Overview

We should share our secrets

Daan Sprekels

28 December 2017
Overview

salsa20/poly1305 decrypt

unbitslice

bitslice

Lagrange interpolation

key
ciphertext

secret

Daan Sprenkels

We should share our secrets

28 December 2017
Implementation performance

Measuring wall clock time\textsuperscript{3} with \((n, k) = (5, 4)\)

<table>
<thead>
<tr>
<th>language</th>
<th>create</th>
<th>combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight C loop</td>
<td>9.6(\mu)s</td>
<td>12(\mu)s</td>
</tr>
<tr>
<td>Go bindings</td>
<td>11(\mu)s</td>
<td>15(\mu)s</td>
</tr>
<tr>
<td>Rust bindings</td>
<td>8.8(\mu)s</td>
<td>5.4(\mu)s</td>
</tr>
</tbody>
</table>

\textsuperscript{3}Wall clock time, best of three on my crappy laptop
Implementation performance

Measuring wall clock time\(^3\) with \((n, k) = (5, 4)\)

<table>
<thead>
<tr>
<th>language</th>
<th>create</th>
<th>combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight C loop</td>
<td>9.6(\mu s)</td>
<td>12(\mu s)</td>
</tr>
<tr>
<td>Go bindings</td>
<td>11(\mu s)</td>
<td>15(\mu s)</td>
</tr>
<tr>
<td>Rust bindings</td>
<td>8.8(\mu s)</td>
<td>5.4(\mu s)</td>
</tr>
</tbody>
</table>

Conclusion: i.e. roughly 100 000 calls per second.

\(^3\)Wall clock time, best of three on my crappy laptop
Stuff that can go wrong

Possible mistakes:
- Assuming integrity
- Timing attacks
- Bad randomness
Can our software be used with malicious intent?
Can our software be used with malicious intent?
“Don’t implement your own crypto”
Acknowledgements

- Ed Schouten
- Ken Swenson
- Pol van Aubel
- Thijs Miedema

Cartoons on frame 9 authored by Randall Monroe
Thank you!

Slides can be found at https://dsprenkels.com/files/sss-34c3.pdf
sss project is at https://github.com/dsprenkels/sss

Extra reading:
▶ http://loup-vaillant.fr/articles/implemented-my-own-crypto
▶ https://dsprenkels.com/mysterion.html

Find me through
▶ Email: hello@dsprenkels.com
▶ PGP key: 951D 6F6E C19E 5D87 1A61 A7F4 1445 C075 FFD5 68CD
References


Example: combining shares (computation)

\[ s_1 = (1, 21), \ s_3 = (4, 6), \ s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
    a_2 x_1^2 + a_1 x_1 + m &= y_1 \\
    a_2 x_2^2 + a_1 x_2 + m &= y_2 \\
    a_2 x_3^2 + a_1 x_3 + m &= y_3 
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
1^2 a_2 + a_1 + m &= 21 \\
4^2 a_2 + 4a_1 + m &= 6 \\
2^2 a_2 + 2a_1 + m &= 8
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), \ s_3 = (4, 6), \ s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
    a_2 + a_1 + m &= 21 \\
    16a_2 + 4a_1 + m &= 6 \\
    4a_2 + 2a_1 + m &= 8
\end{align*}
\]
Example: combining shares (computation)

$$s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8)$$

Solve for \( m \):

$$4a_2 + 4a_1 + 4m = 84$$
$$16a_2 + 4a_1 + m = 6$$
$$4a_2 + 2a_1 + m = 8$$
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
2a_1 + 3m &= 76 \\
16a_2 + 4a_1 + m &= 6 \\
4a_2 + 2a_1 + m &= 8
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
2a_1 + 3m &= 76 \\
16a_2 + 4a_1 + m &= 6 \\
16a_2 + 8a_1 + 4m &= 32
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[ 2a_1 + 3m = 76 \]
\[ 16a_2 + 4a_1 + m = 6 \]
\[ 4a_1 + 3m = 26 \]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
2a_1 + 3m &= 76 \\
4a_1 + 3m &= 26
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), \ s_3 = (4, 6), \ s_4 = (2, 8) \]

Solve for \( m \):

\[
\begin{align*}
4a_1 + 6m &= 152 \\
4a_1 + 3m &= 26
\end{align*}
\]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solve for \( m \):

\[ 3m = 126 \]
\[ 4a_1 + 3m = 26 \]
Example: combining shares (computation)

\[ s_1 = (1, 21), \ s_3 = (4, 6), \ s_4 = (2, 8) \]

Solve for \( m \):

\[ 3m = 126 \]
Example: combining shares (computation)

\[ s_1 = (1, 21), s_3 = (4, 6), s_4 = (2, 8) \]

Solved for \( m \):

\[ m = 42 \]
Lagrange interpolation

Given shares \( s_1, \ldots, s_k = (x_1, y_1), \ldots, (x_k, y_k) \).
Use Lagrange interpolation to get \( m \).

\[
\ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_1)}{(x_i - x_1)} \cdots \frac{(x - x_k)}{(x_i - x_k)} \tag{2}
\]

\[
L(x) = \sum_{i=0}^{k} y_i \ell_i(x) = y_1 \ell_1(x) + \ldots + y_k \ell_k(x) \tag{3}
\]
Lagrange interpolation

Given shares \( s_1, \ldots, s_k = (x_1, y_1), \ldots, (x_k, y_k) \).
Use Lagrange interpolation to get \( m \).

\[
\ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_1)}{(x_i - x_1)} \cdots \frac{(x - x_k)}{(x_i - x_k)} \tag{2}
\]

\[
m = L(0) = \sum_{i=0}^{k} y_i \ell_i(0) = y_1 \ell_1(0) + \ldots + y_k \ell_k(0) \tag{3}
\]
Lagrange interpolation

Given shares $s_1, \ldots, s_k = (x_1, y_1), \ldots, (x_k, y_k)$.
Use Lagrange interpolation to get $m$.

$$
\ell_i(0) = \prod_{j \neq i} \frac{0 - x_j}{x_i - x_j} = \frac{(0 - x_1)}{(x_i - x_1)} \cdots \frac{(0 - x_k)}{(x_i - x_k)}
$$

(2)

$$
m = L(0) = \sum_{i=0}^{k} y_i \ell_i(0) = y_1 \ell_1(0) + \ldots + y_k \ell_k(0)
$$

(3)
Lagrange interpolation

Given shares $s_1, \ldots, s_k = (x_1, y_1), \ldots, (x_k, y_k)$.
Use Lagrange interpolation to get $m$.

$$
\ell_i = \prod_{j \neq i} \frac{-x_j}{x_i - x_j} = \frac{(-x_1)}{(x_i - x_1)} \cdots \frac{(-x_k)}{(x_i - x_k)} \quad (2)
$$

$$
m = \sum_{i=0}^{k} y_i \ell_i = y_1 \ell_1 + \ldots + y_k \ell_k \quad (3)
$$