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"

“Don’t roll your own crypto”

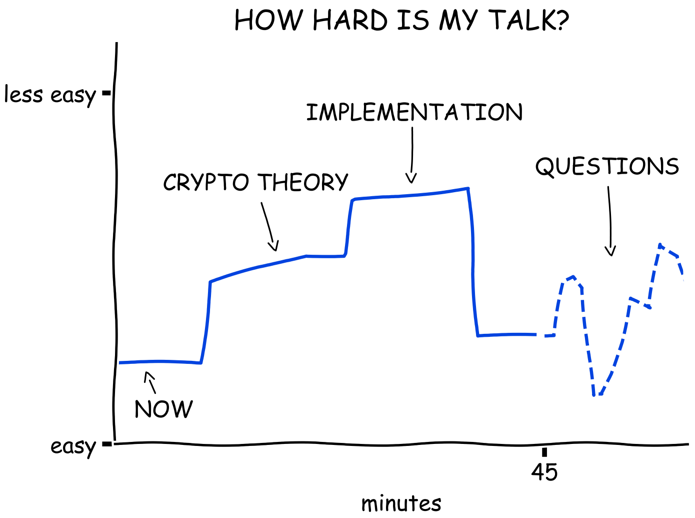
“and also don’t implement your own crypto”

Outline

Part I: Crypto theory
- What is secret sharing?
- How does it work?

Part II: Crypto implementation
- Solving integrity
- How to encode our values
- Side channel resistance
- Performance and bitslicing

Outline



HOW HARD IS MY TALK?
less easy
easy
IMPLEMENTATION
CRYPTO THEORY
QUESTIONS
NOW
45
minutes

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.

Solving our problem

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

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 $C = m \oplus A \oplus B$.

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 $m' = A \oplus B \oplus C = \cancel{A} \oplus \cancel{B} \oplus (m \oplus \cancel{A} \oplus \cancel{B})$

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

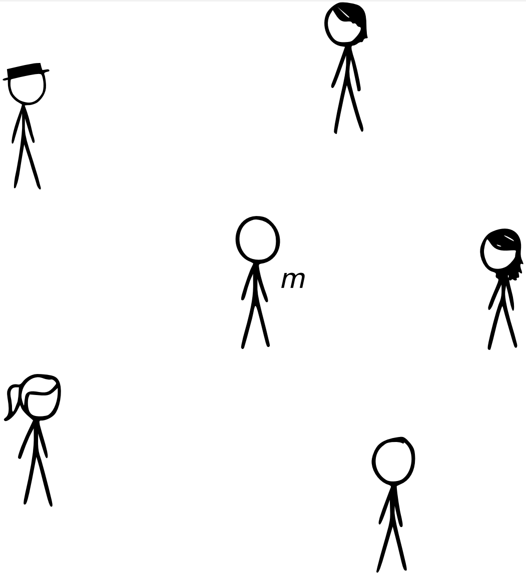
- Published almost 40 years ago by Adi Shamir
- Threshold scheme (n, k)
- “Provably secure”

A better solution

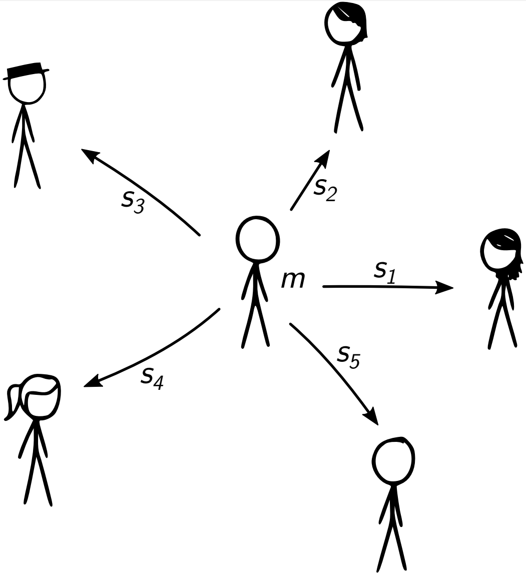
Shamir secret sharing

- Published almost 40 years ago by Adi Shamir
- Threshold scheme (n, k)
- ~~"Provably secure"~~ Information-theoretically secure

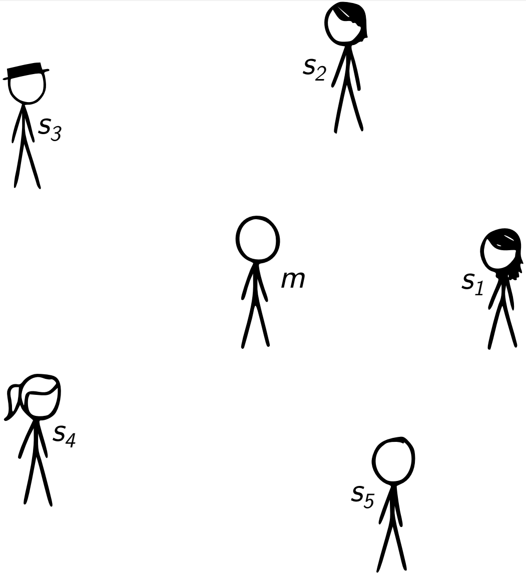
Example with $(n, k) = (5, 4)$



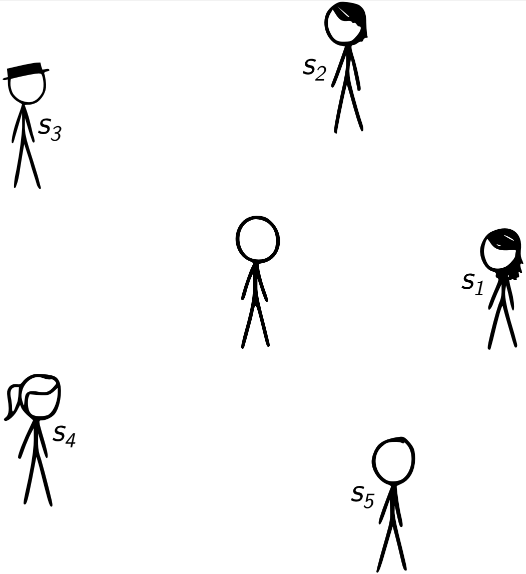
Example with $(n, k) = (5, 4)$



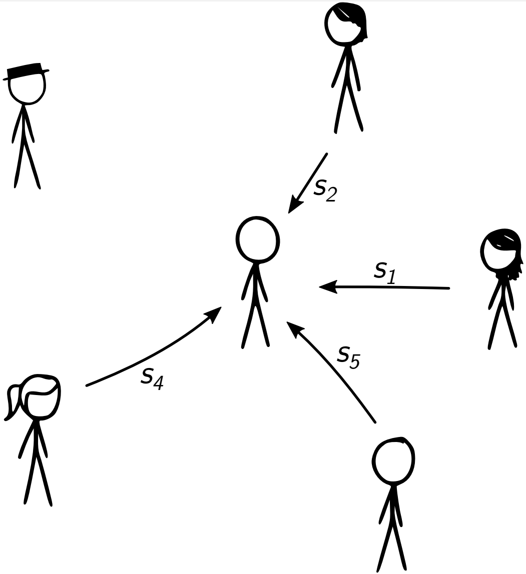
Example with $(n, k) = (5, 4)$



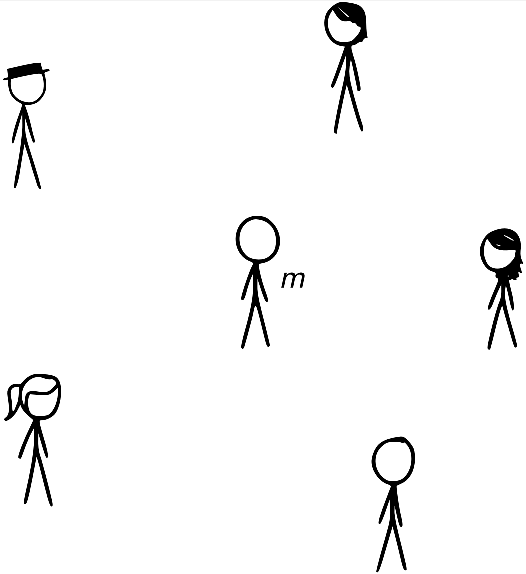
Example with $(n, k) = (5, 4)$



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 $k - 1$:

$$p(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + \boldsymbol{m} \tag{1}$$

With coefficients a_i randomly generated.

How does the math work?

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

$$p(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + \boldsymbol{m} \tag{1}$$

With coefficients a_i randomly generated.

Evaluate n points on the polynomial to get shares s_i:

$$\begin{aligned} s_1 &= (1, p(1)) \\ s_2 &= (2, p(2)) \\ &\vdots \\ s_n &= (n, p(n)) \end{aligned}$$

How does the math work?

Find $p(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + m$ such that all s_i are on $p(x)$.

Solve for m:

$$
\begin{aligned}
a_{k-1}x_1^{k-1} + \ldots + a_1 x_1 + m &= y_1 \\
a_{k-1}x_2^{k-1} + \ldots + a_1 x_2 + m &= y_2 \\
a_{k-1}x_3^{k-1} + \ldots + a_1 x_3 + m &= y_3 \\
\ldots \\
a_{k-1}x_k^{k-1} + \ldots + a_1 x_k + m &= y_k
\end{aligned}
$$

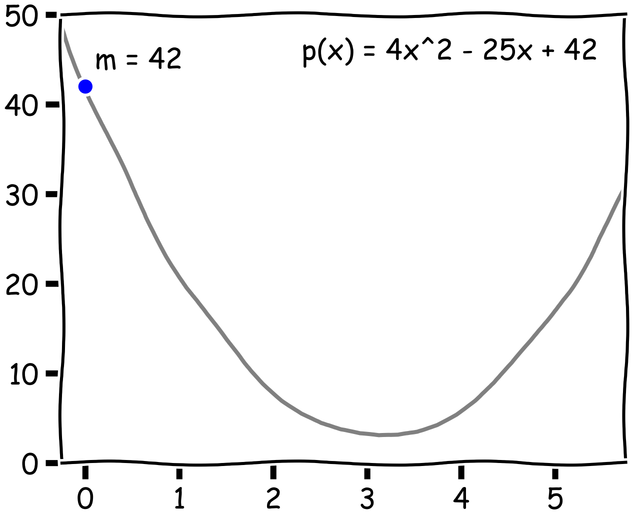
How does the math work?

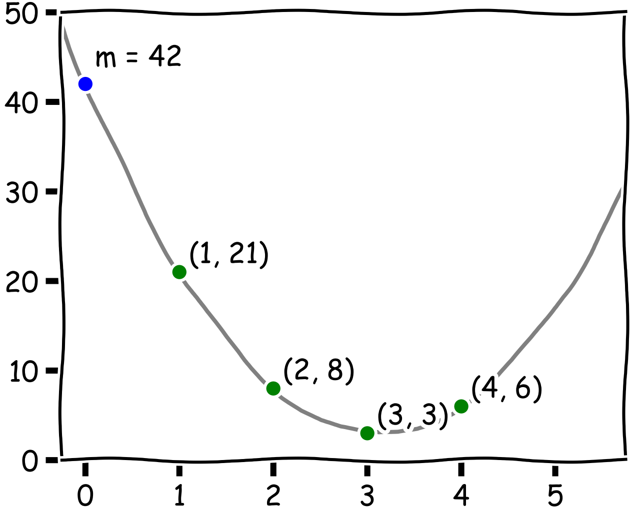
Find $p(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + m$ such that all s_i are on $p(x)$.

Solve for m:

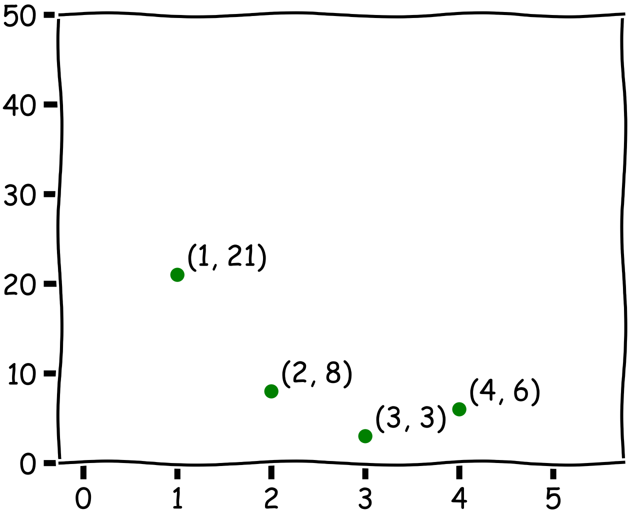
$$\begin{aligned}
a_{k-1}x_1^{k-1} + \ldots + a_1 x_1 + m &= y_1 \\
a_{k-1}x_2^{k-1} + \ldots + a_1 x_2 + m &= y_2 \\
a_{k-1}x_3^{k-1} + \ldots + a_1 x_3 + m &= y_3 \\
\ldots \\
a_{k-1}x_k^{k-1} + \ldots + a_1 x_k + m &= y_k
\end{aligned}$$

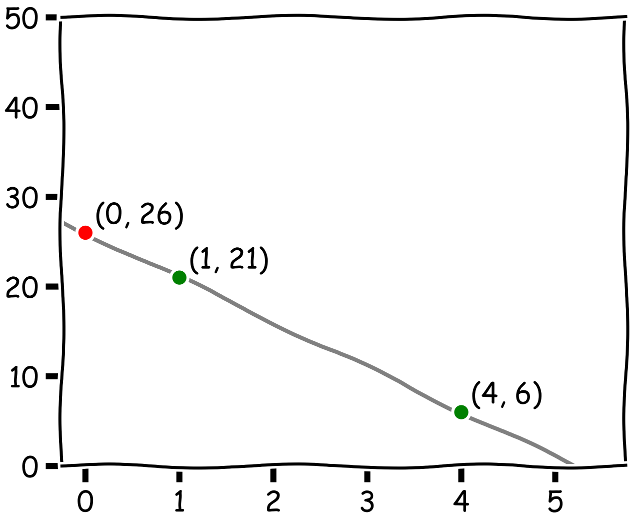
Use Lagrange interpolation for solving

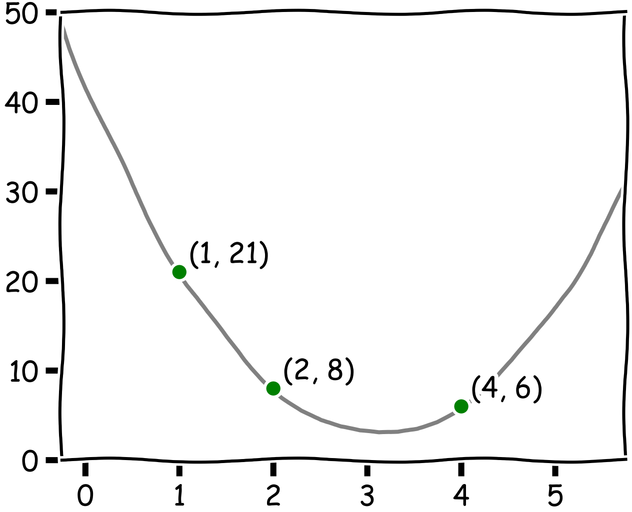

m = 42
p(x) = 4x^2 - 25x + 42
50
40
30
20
10
0
0
1
2
3
4
5

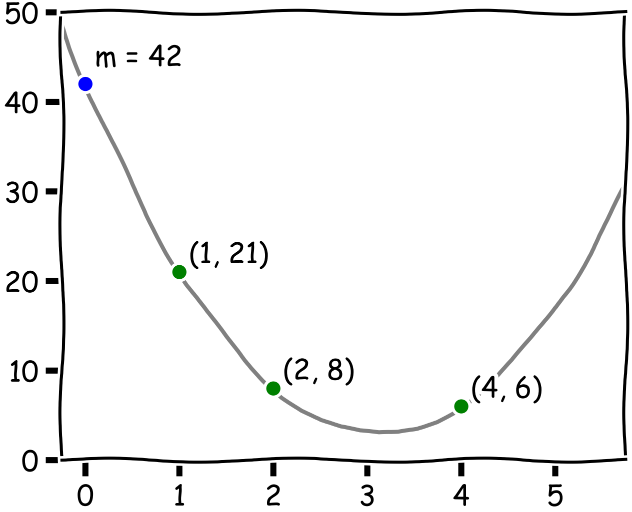


m = 42
(1, 21)
(2, 8)
(3, 3)
(4, 6)
50
40
30
20
10
0
0
1
2
3
4
5


50
40
30
20
10
0
(1, 21)
(2, 8)
(3, 3)
(4, 6)
0
1
2
3
4
5


50
40
30
20
10
0
0
1
2
3
4
5
(0, 26)
(1, 21)
(4, 6)


50
40
30
20
10
0
0
1
2
3
4
5
(1, 21)
(2, 8)
(4, 6)



m = 42
(1, 21)
(2, 8)
(4, 6)
0
10
20
30
40
50
0
1
2
3
4
5

Example: combining shares

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

Solve for m:

$$\begin{aligned} a_2 x_1^2 + a_1 x_1 + m &= y_1 \\ a_2 x_2^2 + a_1 x_2 + m &= y_2 \\ a_2 x_3^2 + a_1 x_3 + m &= y_3 \end{aligned}$$

Example: combining shares

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

Solve for m:

$$\begin{aligned} 1^2 a_2 + a_1 + m &= 21 \\ 4^2 a_2 + 4a_1 + m &= 6 \\ 2^2 a_2 + 2a_1 + m &= 8 \end{aligned}$$

Example: combining shares

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

Solve for m:

$$\begin{aligned} 1^2 a_2 + a_1 + m &= 21 \\ 4^2 a_2 + 4a_1 + m &= 6 \\ 2^2 a_2 + 2a_1 + m &= 8 \end{aligned}$$

$$m = \mathbf{42}$$

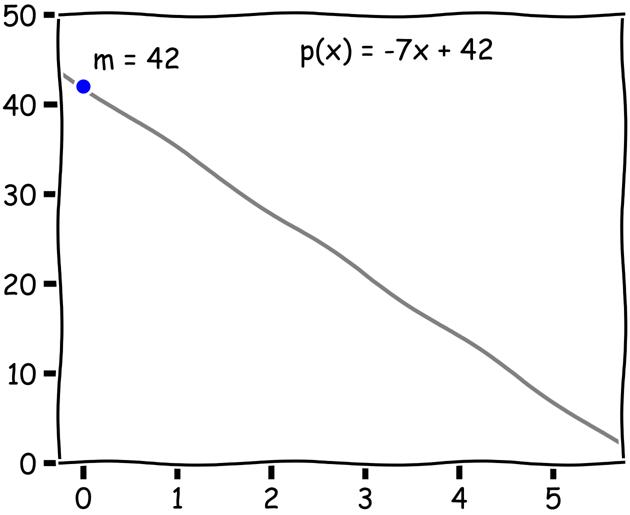
All good?

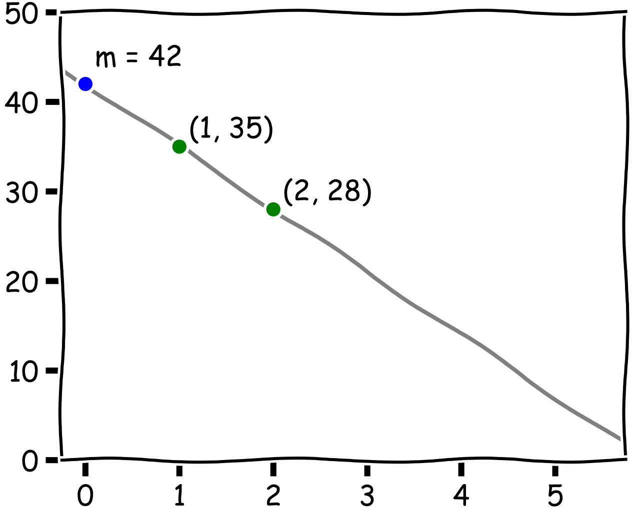
All good?

- Information-theoretically secure

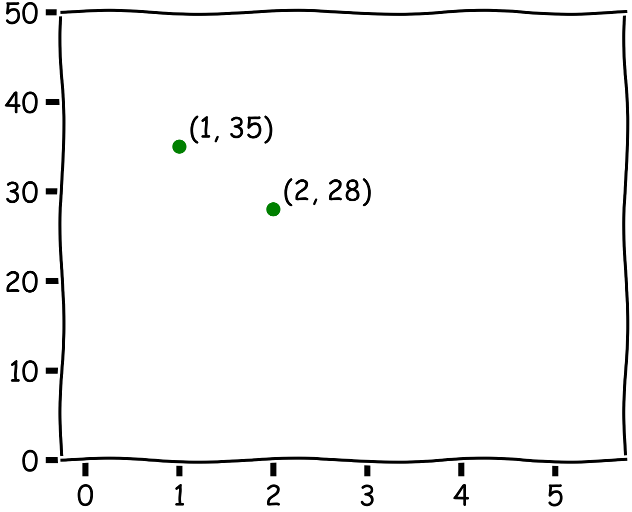
All good?

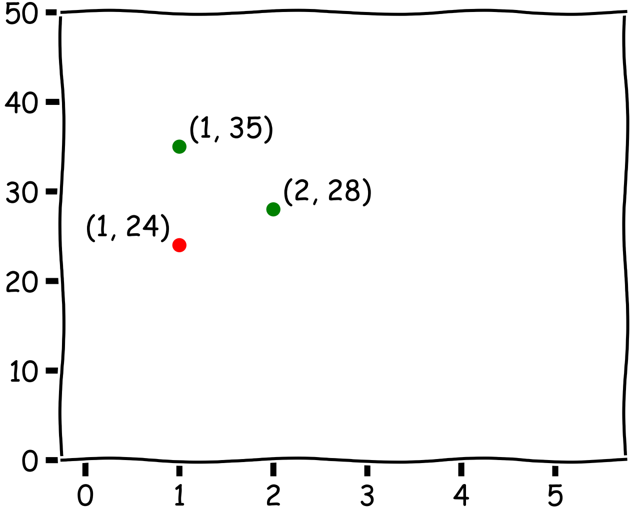
- Information-theoretically secure *for confidentiality*
- Not really secure for *integrity*


m = 42
p(x) = -7x + 42
50
40
30
20
10
0
0
1
2
3
4
5

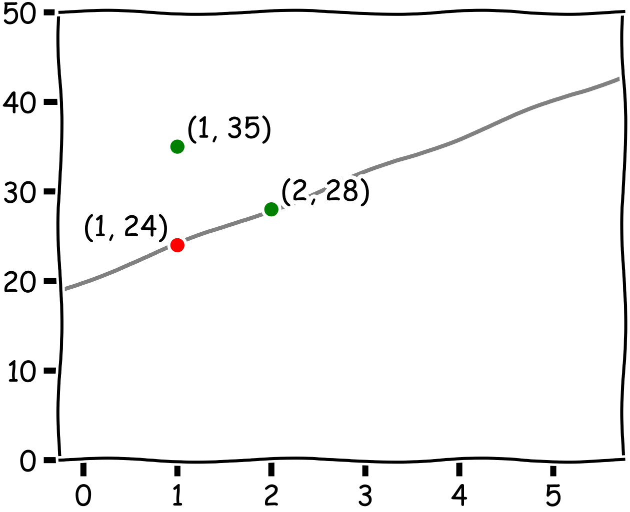


m = 42
(1, 35)
(2, 28)
50
40
30
20
10
0
0
1
2
3
4
5

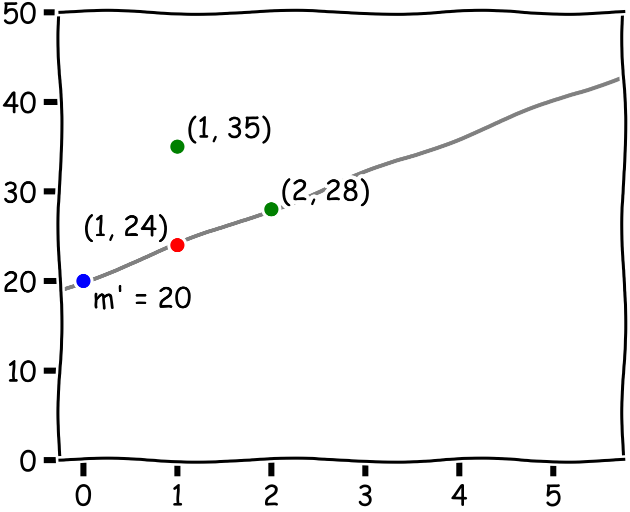

50
40
30
20
10
0
0
1
2
3
4
5
(1, 35)
(2, 28)



50
40
30
20
10
0
0
1
2
3
4
5
(1, 35)
(2, 28)
(1, 24)



50
40
30
20
10
0
0
1
2
3
4
5
(1, 35)
(2, 28)
(1, 24)


50
40
30
20
10
0
0
1
2
3
4
5
(1, 35)
(2, 28)
(1, 24)
m' = 20

Solving integrity

Solutions:

- Randomize x-values
- Only share random secrets

Part II: implementation

Requirements

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

Requirements

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`

Requirements

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`

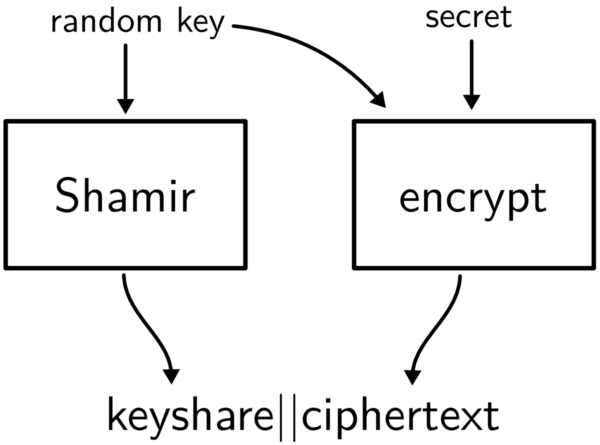
Both do not meet our requirements

Implementation challenges

On to implement it ourselves...

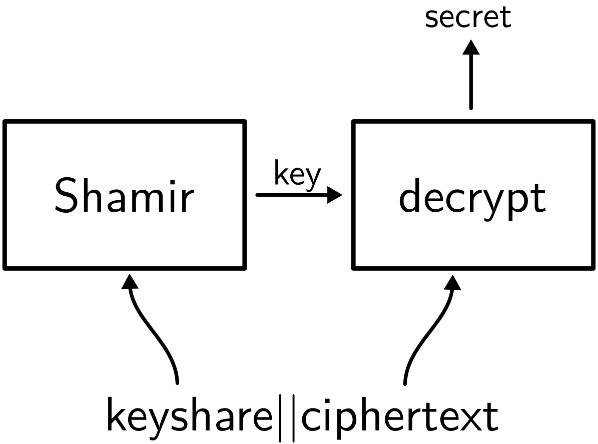
1. How to fix our integrity problem?
2. How to encode our values?
3. How to prevent side channels?
4. How to make it fast?

Use hybrid encryption:



1. Solving integrity

Use hybrid encryption:



2. How to encode our values?

Options:

- Integers modulo large prime?
- Other finite field?

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

2. 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 $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$

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 = table[key[i]];`)

[2] In *software*! Hardware implementations are a whole other story.

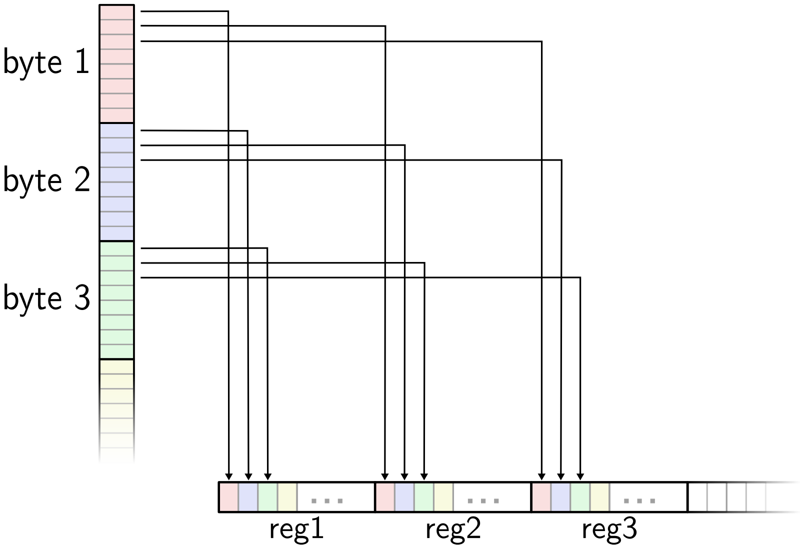
3. How to prevent side channel attacks?

Rules to protect against side channels[2]:

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

[2] In *software*! Hardware implementations are a whole other story.

4. Performance throug bitslicing

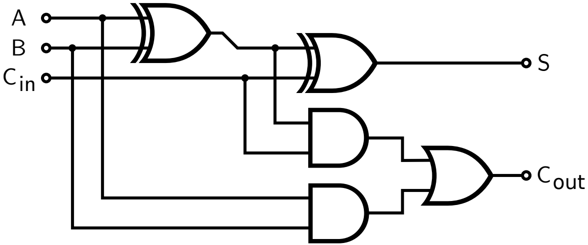


4. Performance throug bitslicing

- Working in bytes $\Rightarrow$ need only 8 registers per byte
- Implement algorithm in logic circuits

4. Performance throug bitslicing

Example: *Adding two bytes in parallel*



4. Performance throug bitslicing

- Working in bytes $\Rightarrow$ need only 8 registers per byte
- Implement algorithm in logic circuits
- 32-bit platform? 32x parallel computation

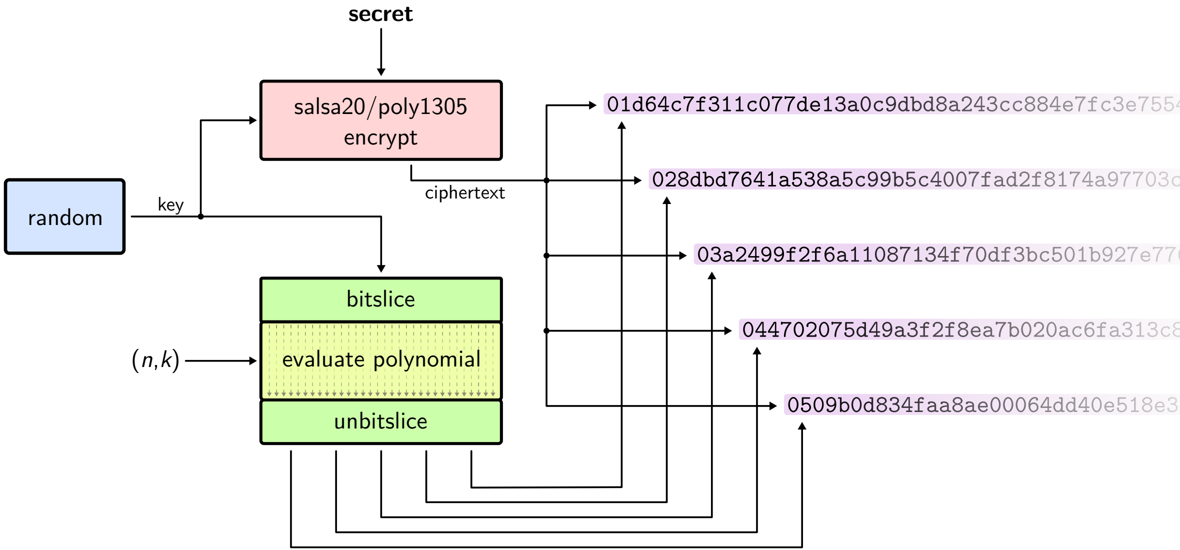
4. Performance throug bitslicing

- Working in bytes $\Rightarrow$ need only 8 registers per byte
- Implement algorithm in logic circuits
- 32-bit platform? 32x parallel computation = **performance :)**

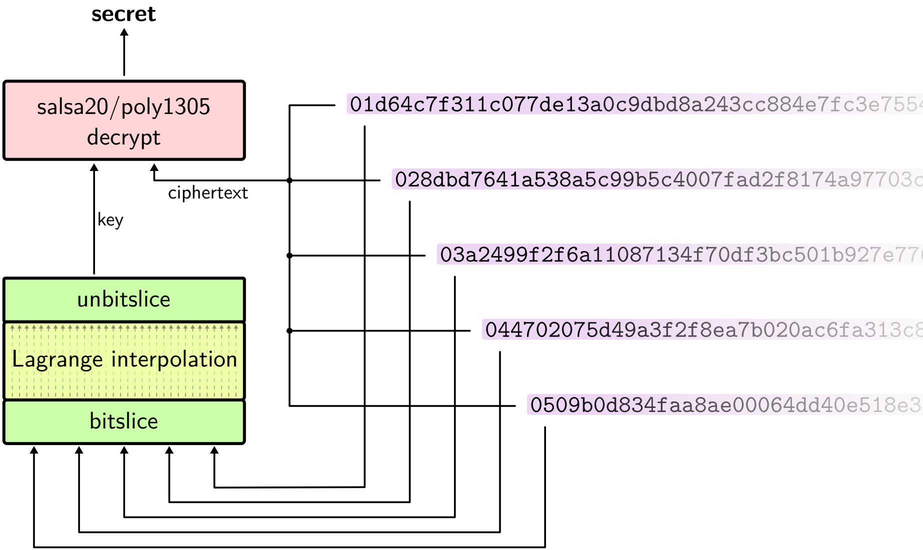
4. Performance throug bitslicing

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

Overview


secret
salsa20/poly1305
encrypt
random
key
ciphertext
bitslice
(n,k)
evaluate polynomial
unbitslice
01d64c7f311c077de13a0c9dbd8a243cc884e7fc3e755
028dbd7641a538a5c99b5c4007fad2f8174a97703
03a2499f2f6a11087134f70df3bc501b927e77
044702075d49a3f2f8ea7b020ac6fa313c8
0509b0d834faa8ae00064dd40e518e3

Overview


secret
salsa20/poly1305
decrypt
key
ciphertext
unbitslice
Lagrange interpolation
bitslice
01d64c7f311c077de13a0c9dbd8a243cc884e7fc3e755
028dbd7641a538a5c99b5c4007fad2f8174a97703
03a2499f2f6a11087134f70df3bc501b927e77
044702075d49a3f2f8ea7b020ac6fa313c
0509b0d834faa8ae00064dd40e518e3

Implementation performance

Measuring wall clock time[3] with $(n, k) = (5, 4)$

language	create	combine
Tight C loop	$9.6\mu s$	$12\mu s$
Go bindings	$11\mu s$	$15\mu s$
Rust bindings	$8.8\mu s$	$5.4\mu s$

[3] Wall clock time, best of three on my crappy laptop

Implementation performance

Measuring wall clock time[3] with $(n, k) = (5, 4)$

language	create	combine
Tight C loop	$9.6\mu s$	$12\mu s$
Go bindings	$11\mu s$	$15\mu s$
Rust bindings	$8.8\mu s$	$5.4\mu s$

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

Ethics

Ethics

Can our software be used with malicious intent?

Demo

“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`
`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
- DECT extension: 4597

Acknowledgements:

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

References

- https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf (Jun 2016)
- http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0184b/Chdedhfg.html (2017)
- https://bitcointalk.org/index.php?topic=2199659.0 (2017)
- https://cryptocoding.net/index.php/Coding_rules (2017)
- Pedersen, T.P., et al.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Crypto. vol. 91, pp. 129–140. Springer (1991)
- Poettering, B.: Shamir Secret Sharing Scheme. http://point-at-infinity.org/ssss/ (2006)
- Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (Nov 1979), http://doi.acm.org/10.1145/359168.359176
- Silverstone, D.: gfshare. http://www.digital-scurf.org/index.html (2006)

Example: combining shares (computation)

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

Solve for m:

$$
\begin{aligned}
a_2 x_1^2 + a_1 x_1 + m &= y_1 \\
a_2 x_2^2 + a_1 x_2 + m &= y_2 \\
a_2 x_3^2 + a_1 x_3 + m &= y_3
\end{aligned}
$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 1^2 a_2 + \ \ a_1 + m &= 21 \\ 4^2 a_2 + 4a_1 + m &= \ \ 6 \\ 2^2 a_2 + 2a_1 + m &= \ \ 8 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} a_2 + \ \ a_1 + m &= 21 \\ 16a_2 + 4a_1 + m &= \ \ 6 \\ 4a_2 + 2a_1 + m &= \ \ 8 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 4a_2 + 4a_1 + 4m &= 84 \\ 16a_2 + 4a_1 + m &= 6 \\ 4a_2 + 2a_1 + m &= 8 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 2a_1 + 3m &= 76 \\ 16a_2 + 4a_1 + m &= 6 \\ 4a_2 + 2a_1 + m &= 8 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 2a_1 + 3m &= 76 \\ 16a_2 + 4a_1 + m &= 6 \\ 16a_2 + 8a_1 + 4m &= 32 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 2a_1 + 3m &= 76 \\ 16a_2 + 4a_1 + \quad m &= \ \ 6 \\ 4a_1 + 3m &= 26 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$2a_1 + 3m = 76$$
$$4a_1 + 3m = 26$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 4a_1 + 6m &= 152 \\ 4a_1 + 3m &= \ \ 26 \end{aligned}$$

Example: combining shares (computation)

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

Solve for m:

$$\begin{aligned} 3m &= 126 \\ 4a_1 + 3m &= \ 26 \end{aligned}$$

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:

$$m = \mathbf{42}$$

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.

$$\ell_i(0) = \prod_{j \neq i} \frac{0 - x_j}{x_i - x_j} = \frac{(0 - x_1)}{(x_i - x_1)} \cdots \frac{(0 - x_k)}{(x_i - x_k)} \tag{2}$$

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

Lagrange interpolation

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

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

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