How to do things with numbers

A digression

Jeffrey Goldberg
jeffrey@goldmark.org

April 4, 2022
Revised August 16, 2024

Numbers

Wholesomeness

Definition 1 (Number)

For the purposes here, so called natural numbers $\mathbb{N}$ (including 0). These are the familiar numbers $\{0, 1, 2, 3, ...\}$

Definition 1 (Number)
For the purposes here, so called natural numbers $\mathbb{N}$ (including 0). These are the familiar numbers $\{0, 1, 2, 3, \ldots\}$

1. Gotlieb Frage, using Gregor Cantor's set theory, was the first to define numbers in terms of something more basic. But before you ask, I need to tell you that the definition does absolutely nothing to communicate a sense of what natural number means.
2. Cantor was effectively blacklisted from positions and publishing because Leopold Kronecker really hated that Cantor not only talked about infinities, but because Cantor developed a solid theory of them. Cantor, always a bit unstable, eventually went mad and killed himself.
3. For some people the set $\mathbb{N}$ includes 0, but for other people it doesn't. So people tend to write things like "$\mathbb{N}$ (including 0)" or "$\mathbb{N}$ (excluding 0)".

Naturally

Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk

The whole numbers were created by God, everything else is the work of men.

Leopold Kronecker (1886)

Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk

The whole numbers were created by God, everything else is the work of men.

Leopold Kronecker (1886)

1. Good quote, but Kronecker was seriously on the wrong side of mathematics.
2. The terms “natural numbers” and “integers” hadn’t been invented at the time.
3. Kronecker would be greatly pained to see how $\mathbb{N}$ is defined today. I take pleasure in that.
4. Frege was a nasty piece of work, but he enormously advanced mathematical foundations and logic, unlike Kronecker, who held it back.
5. Frege, who thought that all Jews should be exterminated from Europe, built all his work, including his definition of number, the work of the very Jewish Cantor.

Representing numbers

What you already know

Example 2

$$
\begin{aligned}
2705 &= 2 \cdot 1000 + 7 \cdot 100 + 0 \cdot 10 + 5 \cdot 1 \\
&= (5 \cdot 1) + (0 \cdot 10) + (7 \cdot 100) + (2 \cdot 1000) && \text{(left-to-right)} \\
&= (5 \cdot 10^0) + (0 \cdot 10^1) + (7 \cdot 10^2) + (2 \cdot 10^3)
\end{aligned}
$$

Example 2

$$\begin{aligned} 2705 &= 2 \cdot 1000 + 7 \cdot 100 + 0 \cdot 10 + 5 \cdot 1 \\ &= (5 \cdot 1) + (0 \cdot 10) + (7 \cdot 100) + (2 \cdot 1000) \quad \text{(left-to-right)} \\ &= (5 \cdot 10^0) + (0 \cdot 10^1) + (7 \cdot 10^2) + (2 \cdot 10^3) \end{aligned}$$

1. Through an accident of history our way of representing numbers isn't well suited for reading left to right.

A numeral system by any other name

Definition 3 (IAN System)

What is sometimes called the “Arabic-Hindu numeral system” or the “Arabic numerals” I am going to call the “Indian-Arabic numeral system” (IAN system).

Definition 3 (IAN System)

What is sometimes called the “Arabic-Hindu numeral system” or the “Arabic numerals” I am going to call the “Indian-Arabic numeral system” (IAN system).

1. 9th century CE scholars, writing in Arabic called these “hindi” numerals, but ”hindi” was the Arabic word for “Indian”.

Same thing, base 8

Example 4

$$
\begin{aligned}
5221_8 &= 5 \cdot 256 + 2 \cdot 64 + 2 \cdot 8 + 1 \cdot 1 \\
&= (1 \cdot 1) + (2 \cdot 8) + (2 \cdot 64) + (5 \cdot 256) && \text{(left-to-right)} \\
&= (1 \cdot 8^0) + (2 \cdot 8^1) + (2 \cdot 8^2) + (5 \cdot 8^3)
\end{aligned}
$$

Roman

Example 5

$$\text{MMDCCV} = 1000 + 1000 + 500 + 100 + 100 + 5$$

What makes the IAN system better?

- Works well for fairly large numbers.

- Works well for fairly large numbers.

1. It was certainly possible to talk about large numbers as Archimedes did in *The Sand Reckoner* in which he estimated how many grains of sand it would take to fill the universe. In estimating the size of the universe, he took the view that the sun was at the center.
2. We do this by moving to negative exponents of the base. $4.83 = 4 \cdot 10^0 + 8 \cdot 10^{-1} + 3 \cdot 10^{-2}$
3. 0 is necessary for the place-based system, but this didn't really invent zero, despite what you may have been told.

What makes the IAN system better?

- Works well for fairly large numbers.
- System can be adapted to work with fractional amounts

What makes the IAN system better?

- Works well for fairly large numbers.
- System can be adapted to work with fractional amounts
- It has zero.

What makes the IAN system better?

- Works well for fairly large numbers.
- System can be adapted to work with fractional amounts
- It has zero.
- It’s easier to compare which numbers are bigger

What makes the IAN system better?

- Works well for fairly large numbers.
- System can be adapted to work with fractional amounts
- It has zero.
- It's easier to compare which numbers are bigger
- *It makes arithmetic easy*

Algorithms

A division

Annoying task

Divide MMDCCV by DXLI. Give your answer in Roman numerals. Do not use a computer.

Annoying task

Divide MMDCCV by DXLI. Give your answer in Roman numerals. Do not use a computer.

1. I am not actually asking anyone to do this. But we will come back to it.

Enabling algorithms

- 9th century CE, Muhammad ibn Mūsā al-Khwārizmī wrote a textbook, *Al-jabr*.
- The words “algorithm” and “algebra” come from the author’s name and book title.
- He also wrote *Computation the Indian Way* in 825 CE.
- Leonardo of Pisa (aka Fibonacci) popularized this way of doing arithmetic in Europe with his 1202 book *Liber Abaci*.

- 9th century CE, Muhammad ibn Mūsā al-Khwārizmī wrote a textbook, *Al-jabr*.
- The words “algorithm” and “algebra” come from the author’s name and book title.
- He also wrote *Computation the Indian Way* in 825 CE.
- Leonardo of Pisa (aka Fibonacci) popularized this way of doing arithmetic in Europe with his 1202 book *Liber Abaci*.

1. al-Khwārizmī, as Chief Astronomer in Bagdad, wrote in Arabic, but his native language may have been Farsi.
2. The original Arabic text on Indian computation is lost, but it was frequently cited and was translated to Latin.

Algorithms & Representations

- The algorithms for arithmetic are deeply tied to the ways the numbers are represented
- Arithmetic becomes symbol manipulation as a consequence of the place based system.
- The representation and the algorithms are a package deal.
- I am repeating this in multiple ways because this really is a crucial point.

Learning to multiply

- Memorize single digit multiplication table.
- Memorize (through practice) the algorithm for multi-digit multiplication.
- Congratulations! You can now add and multiply numbers of any size.
- This can be taught to children. One no longer needs a university education to be able to multiply.

Learning to multiply

- Memorize single digit multiplication table.
- Memorize (through practice) the algorithm for multi-digit multiplication.
- Congratulations! You can now add and multiply numbers of any size.
- This can be taught to children. One no longer needs a university education to be able to multiply.

1. There are multiple closely related varients of these algorithms. They are all based on the place-based system in the same way, but are notationally different. Watching my Hungarian educated wife do long division confuses me.

Algorithms

In computers

It's not just base-2

- A byte is just an ordinary base-2 place-based number.
- Bytes can be seen as the base for a base-256 system.
- The order of bytes in representing a number depends on whether the chip architecture is big-endian or little-endian.
- Negative integers are represented in two's-complement form.

The maze isn't meant for you

- CPUs (and other hardware) are better at manipulating bytes than bits.
- Endianness is tied to fine details of how memory is accessed within the CPU.
- The highly specialized hardwired algorithms that chips use for arithmetic work directly on two's complement representations, without having to do something special for negative numbers.
- These representations are not meant for paper and pencil arithmetic. They are tuned to the algorithms used by computers internally.

Different ways of representing numbers enable and are tied to different algorithms for doing things with the numbers.

Homomorphisms

A division

Annoying task

Divide MMDCCV by DXLI. Give your answer in Roman numerals. Do not use a computer.

A sensible approach

1. Convert the Roman numerals to IAN system.
2. Do paper and pencil long division to get an answer in IAN system.
3. Convert that solution back to Roman numerals.

Homomorphism

Definition 6 (Homomorphism)

Two domains, both alike in dignity, in ...

(Never mind. A definition would not be useful at this point.)

There and back again

1. Start a problem that is hard/annoying to solve in its own little world.
2. Translate it to a problem in a different little world.
3. Solve the problem in that second world.
4. Translate back to the original world.

All roads lead to (and from) Rome

1. Start with multiplication problem in the world of Roman numerals.
2. Translate it to a problem in the IAN system.
3. Solve the problem in the IAN system.
4. Translate back to Roman numerals.

A homomorphism is a relationship between worlds and problems that allows that to work.

All roads lead to (and from) Rome

1. Start with multiplication problem in the world of Roman numerals.
2. Translate it to a problem in the IAN system.
3. Solve the problem in the IAN system.
4. Translate back to Roman numerals.

A homomorphism is a relationship between worlds and problems that allows that to work.

1. The term homomorphism is not really used for such simple examples. I am stretching to make use of an accessible example.

Chinese Remainder Theorem

Picking up the pieces

How many remains?

You are a serial killer, and one of your victims is from China. You've chopped up this victim's remains into many pieces (but no more than 900 pieces), which you store in in a bin. One day, the bin overturns and all of the pieces spill out.

- First you gather up the pieces in groups of 25 (because you are that kind of lunatic) and find that 14 remains remain,
- then you gather them up in groups of 9 and find that one remain remains,
- finally, you gather them up in groups of 4 and find that three remains remain.

How many pieces are there in total?

Picking up the pieces

How many remains?

You are a serial killer, and one of your victims is from China. You've chopped up this victim's remains into many pieces (but no more than 900 pieces), which you store in in a bin. One day, the bin overturns and all of the pieces spill out.

- First you gather up the pieces in groups of 25 (because you are that kind of lunatic) and find that 14 remains remain,
- then you gather them up in groups of 9 and find that one remain remains,
- finally, you gather them up in groups of 4 and find that three remains remain.

How many pieces are there in total?

1. Someone pointed out to me that the answer, 739, is interesting in its own right. The two notable births for 739 CE listed on Wikipedia are both Chinese. The number is the 131st prime.
2. 131 is a Sophie Germain prime. And it is the CRT that requires that we use safe primes for our finite fields.
3. Those are most likely coincidences, and 739 was chosen because it is annoying to compute without the CRT.

In addition to being a theorem and an algorithm, we would suggest to the reader that the Chinese remainder theorem is also a state of mind. [HPS08, p 86]

CRT representation

Definition 7 (CRT representation)

A Chinese Remainder Theorem representation of a number as a list of moduli and a list (of equal length) of remainders. In our murderous example, the moduli are $(25, 9, 4)$ and the corresponding remainders are $(14, 1, 3)$

.

Definition 7 (CRT representation)
A Chinese Remainder Theorem representation of a number as a list of moduli and a list (of equal length) of remainders. In our murderous example, the moduli are $(25, 9, 4)$ and the corresponding remainders are $(14, 1, 3)$

.

1. Alternatively we could do this as a set of ordered pairs $\{(25, 14), (9, 1), (4, 3)\}$

Chinese Remainder Theorem

Theorem 8 (Chinese Remainder Theorem)

Given a CRT representation of moduli $(M_1, M_2, \dots, M_n)$ and their corresponding remainders $(R_1, R_2, \dots, R_n)$ if the moduli are all mutually coprime, then there is a unique integer less than the product of all of the moduli that satisfies this representation.

Example 9

Given moduli $(25, 9, 4)$ and remainders $(14, 1, 3)$ we see that the moduli are all coprime with each other. This means that there is a single number less than $25 \cdot 9 \cdot 4 = 900$ which could have the given remainders. That number is 739.

Addition

Let's look at 54 plus 739 in a mod $(25, 9, 4)$ world.

$$
\begin{aligned}
739 \bmod (25, 9, 4) &= (14, 1, 3) \\
739 + 54 \bmod (25, 9, 4) &= (14 + 54, 1 + 54, 3 + 54) \\
&= (68, 55, 57) \\
&= (18, 1, 1)
\end{aligned}
$$

Multiplication

Let's look at 123 times 739 in a mod $(25, 9, 4)$ world.

$$
\begin{aligned}
739 \bmod (25, 9, 4) &= (14, 1, 3) \\
739 \cdot 123 \bmod (25, 9, 4) &= (14 \cdot 123, 1 \cdot 123, 3 \cdot 123) \\
&= (1722, 123, 369) \\
&= (22, 6, 1)
\end{aligned}
$$

CRT algorithm

CRT algorithm

```
from sympy.ntheory.modular import crt
```

CRT algorithm

```
from sympy.ntheory.modular import crt
```

1. A proof of the CRT conveniently gives an algorithm for computing the unique integer.
2. There are some good videos out there explaining it if you are interested.

CRT algorithm example

Example 10 (Returning to mod 900)

```
>>> from sympy.ntheory.modular import crt
>>> mods = [25, 9, 4]
>>> crt(mods, [14, 1, 3])[0]
739 # This many pieces, you sicko!
>>> crt(mods, [68, 55, 57])[0] # Plus 54
793
>>> crt(mods, [1722, 123, 369])[0] # Times 123
897
```

CRT and the DLP

Someone trying to solve the discrete logarithm problem in a mod p world, can use the CRT to break the problem down into solving something in a CRT world where the moduli are the factors of $p-1$.

This is why we use safe primes, $p = 2q + 1$ where q is prime, so the factors of $p - 1$ are 2 and q.

Pre-CRT in RSA

Using the CRT in RSA decryption allows us to do computations on numbers around the size of the primes p and q instead of with numbers the size of pq.

```
type PrecomputedValues struct {
    Dp, Dq *big.Int // D mod (P-1) (or mod Q-1)
    Qinv   *big.Int // Q^-1 mod P
    // ... stuff I am skipping
}
```

A fuller listing

```
type PrecomputedValues struct {
  Dp, Dq *big.Int // D mod (P-1) (or mod Q-1)
  Qinv   *big.Int // Q^-1 mod P

  // CRTValues is used for the 3rd and subsequent
  // primes. Due to a historical accident, the CRT
  // for the first two primes is handled
  // differently in PKCS #1 and interoperability
  // is sufficiently important that we mirror this.
  CRTValues []CRTValue
}
```

A FULLER LISTING

```
type PrecomputedValues struct {
  Dp, Dq *big.Int // D mod (P-1) (or mod Q-1)
  Qinv   *big.Int // Q^-1 mod P

  // CRTValues is used for the 3rd and subsequent
  // primes. Due to a historical accident, the CRT
  // for the first two primes is handled
  // differently in PKCS #1 and interoperability
  // is sufficiently important that we mirror this.
  CRTValues []CRTValue
}
```

1. I couldn't put the listing on a notes page due to, well, TEXnical difficulties.
2. Yes. The standards really do define RSA for the modulus being the product of two *or more* primes.

CRT in RSA

1. Start in the world of $N = pq$ trying to solve $m = c^d \bmod N$
2. Do exponentiations with d_p and d_q, in a mod (p, q) world.
3. Do some simple arithmetic with the remainders.
4. Use q_{inv} and the CRT algorithm to get the final decrypted message $m \pmod N$.

Some things are harder in a CRT world

1. I can go with ordering if I can't come up with a better example

Homomorphic encryption

Easy so far

Our examples so far of Roman to IAN, integer to CRT, and IAN to internal computer representations were to make certain computations easier.

Things will get harder from here on out.

Two worlds

- There are two worlds, our world, $\mathcal{W}$ and a homologous world $\mathcal{H}$.
- The function $f(x)$ translates stuff from $\mathcal{W}$ to things in $\mathcal{H}$.
- The function $f'(x')$ translates stuff back from $\mathcal{H}$ to $\mathcal{W}$.
- $x = f'(f(x))$
- In our world, $\mathcal{W}$, there are two operations: '+' and '$\times$'.
- In $\mathcal{H}$ there are two homologous operations $\oplus$ and $\otimes$.
- $x + y = f'(f(x) \oplus f(y))$
- $x \times y = f'(f(x) \otimes f(y))$

Two worlds

- There are two worlds, our world, $\mathcal{W}$ and a homologous world $\mathcal{H}$.
- The function $f(x)$ translates stuff from $\mathcal{W}$ to things in $\mathcal{H}$.
- The function $f'(x')$ translates stuff back from $\mathcal{H}$ to $\mathcal{W}$.
- $x = f'(f(x))$
- In our world, $\mathcal{W}$, there are two operations: '$+$' and '$\times$'.
- In $\mathcal{H}$ there are two homologous operations $\oplus$ and $\otimes$.
- $x + y = f'(f(x) \oplus f(y))$
- $x \times y = f'(f(x) \otimes f(y))$

1. We have an algorithm for translating Roman numerals to IAN
2. If we translate “VII” to ”7” then translating back from “7” better give us ”VII”.
3. Addition and multiplication exist in the Roman world. Even if ungainly.
4. Different algorithms, but they exist in IAN
5. Translating two numbers from Roman to IAN, performing the operation homologous to addition in IAN, and translating the results back gives you the same result you’d get if you performed addition entirely within the world of Roman numerals.
6. You now pretty much know what a homomorphism is.

Making a little bit harder

Now suppose:

- Computing functions $f(\cdot)$ and $f'(\cdot)$ is easy (polynomial).
- The $\mathcal{W}$ operations + and $\times$ are easy (polynomial).
- The $\mathcal{H}$ operation $\oplus$ is easy.
- the $\mathcal{H}$ operation $\otimes$ is hard (not polynomial).

Not really hard

1. You are in $\mathcal{H}$, and you would like to compute $x' \otimes y'$.
2. Compute $x = f'(x')$ and $y = f'(y')$;
3. Compute $z = x \times y$;
4. Compute $z' = f(z)$.

The key to translation

Now suppose that our translations functions require a key

1. We have a task t in $\mathcal{W}$.
2. We pick a random key, k.
3. We use the *keyed* function $f_k(t)$ to create t'.
4. We give the t' to Trinity (third party) to work on. We do not give Trinity the key.
5. We know that Trinity cannot perform operation $\otimes$ on it, we know that he can't translate it back to $\mathcal{W}$ without the key, and we know that he can perform $\oplus$ on it.
6. Trinity performs the task and returns his result, r'.
7. We have the key k and so we can compute $r = f'_k(r')$

Homomorphic encryption

Making this magic useful

Address book comparison problem

Alice and Bob would like to know who they know in common without revealing anyone's telephone number to the other. They want to solve the set intersection of the telephone numbers in their address book. The are looking to solve $m_{AB} = m_A \cap m_B$.

Three kinds of solutions

1. Compare hashes of phone numbers,
2. Use a trusted third party, or
3. Use the magic of homomorphic encryption.

1. Compare hashes of phone numbers,
2. Use a trusted third party, or
3. Use the magic of homomorphic encryption.

1. Hashes really don’t work when the things being hashed are low entropy
2. They don’t want to trust a third party with all of those telephone numbers if they don’t have to.
3. We are talking about homomorphic encryptionm, so Alice and Bob will pick this one.

Computing $m_A \cap m_B$

1. Alice and Bob would to compute $m_{AB} = m_A \cap m_B$
2. Alice and Bob negotiate a key, k
3. Alice transforms her address book: $m'_A = f_k(m_A)$
4. Bob transforms his address book $m'_B = f_k(m_B)$
5. They give m'_A and m'_B to Trinity.
6. Trinity computes $m'_{AB} = m'_A \cap' m'_B$ using the homologous set union operation.
7. Both Bob and Alice can get the intersection using their key: $m_{AB} = f'_k(m'_{AB})$

Computing $m_A \cap m_B$

1. Alice and Bob would to compute $m_{AB} = m_A \cap m_B$
2. Alice and Bob negotiate a key, k
3. Alice transforms her address book: $m'_A = f_k(m_A)$
4. Bob transforms his address book $m'_B = f_k(m_B)$
5. They give m'_A and m'_B to Trinity.
6. Trinity computes $m'_{AB} = m'_A \cap' m'_B$ using the homologous set union operation.
7. Both Bob and Alice can get the intersection using their key: $m_{AB} = f'_k(m'_{AB})$

1. Trinity has to be trusted to do her job properly, but she isn't trusted with any secrets.

Multi-party computation

We don't need Trinity. It is possible for Alice and Bob to exchange data directly with each other using the same kind of mathematical magic, and achieve the same result. That is often called multi-party Computation (MPC). It tends to require more back and forth communication between the parties.

We don't need Trinity. It is possible for Alice and Bob to exchange data directly with each other using the same kind of mathematical magic, and achieve the same result. That is often called multi-party Computation (MPC). It tends to require more back and forth communication between the parties.

1. See [Chi19] for a nice little article on some of the differences between HE and MPC.

Negotiating a meeting time

A group of people would like to find a time available for them all to meet, but do not wish to reveal their schedules to each other or to a third party.

Negotiating a meeting time

A group of people would like to find a time available for them all to meet, but do not wish to reveal their schedules to each other or to a third party.

1. Source code for the Kiss., Schick., and Schneider. [KSS19] web-app at https://github.com/encryptogroup/scheduling
2. Despite her name, Ágnes Kiss does not speak a word of Hungarian and is surprised/annoyed when Americans at security conferences assume that she does. Note that I was the *second* American to start speaking to her during that poster session.

The millionaires' problem

Two millionaires wish to know who is richer; however, they do not want to find out inadvertently any additional information about each other's wealth. How can they carry out such a conversation? [Yao82]

Two millionaires wish to know who is richer; however, they do not want to find out inadvertently any additional information about each other's wealth. How can they carry out such a conversation? [Yao82]

1. That problem was stated in a 1982 paper. Today, of course, it would be billionnaires, whom we might call JB and EM.

Voting

- Fully private voting, where the tallier never sees how any individual voted.
- Each voter can get a proof that their vote was counted.
- No voter can prove to a third party how they voted.

Microsoft Research [Mic19] has podcast (and transcript), with an introduction.

Voting

- Fully private voting, where the tallier never sees how any individual voted.
- Each voter can get a proof that their vote was counted.
- No voter can prove to a third party how they voted.

Microsoft Research [Mic19] has podcast (and transcript), with an introduction.

Answer: Our elections are already easily secure enough against those threats, and so we need to focus on public confidence and availability to eligible voters.

1. Why is the “can’t prove how you voted to a third party” criteron important?
2. Despite the tech for this being known and available, why would this be a terrible system in the US?

Search encrypted data

Apple's location stuff

Homomorphic encryption

Practicalities

"Efficient" doesn't always mean efficient

- Just because an algorithm runs in polynomial time doesn't mean that it will run quickly enough for our needs.
- Just because an algorithm runs in polynomial space, doesn't mean that it doesn't require an enormous amount of space.

40 YEARS OLD

- Any protocol that makes use of a *trusted* third party can be done in polynomial time without it.
- Practical protocols were out of reach when that theorem was proved.

- Any protocol that makes use of a *trusted* third party can be done in polynomial time without it.
- Practical protocols were out of reach when that theorem was proved.

1. When RSA was first invented in GCHQ, it wasn't pursued because it was too computationally intensive for the computers at the time.

Not fully homomorphic

- In our examples, there are some functions which can be computed in the encrypted world and some that can't.
- In practice, schemes are designed around the specific problem and data sizes.
- With *fully* homomorphic encryption (FHE) everything can be computed except that IND-CPA is maintained.
- An FHE scheme that is practical across a broad range of problems and data sizes would make life a lot easier.

HE v MPC

- Homomorphic encryption (HE) is about fairly general mathematical operations like addition and multiplication.
- Multi-party computation (MPC) is about some very specific computations.
- Today, I am lumping these together under HE, but the technologies to implement them are often very different.

DHKE is MPC

A zero-knowledge key exchange is an example multi-party computation.

- No secrets are communicated during the key exchange;
- Both parties compute a common value (the key);
- Each party needs the public information from the other to compute the value.

DHKE is MPC

A zero-knowledge key exchange is an example multi-party computation.

- No secrets are communicated during the key exchange;
- Both parties compute a common value (the key);
- Each party needs the public information from the other to compute the value.

1. People typically don’t talk about SRP or DHKE as MPC, but people don’t typically talk about conversion from Roman to IAN numeral systems as a Homomorphism, either.

It's gotten better

- Algorithms have improved since 1982.
- Computing devices have become more powerful since 1982.

Resources

Resources

- These slides an their sources.
- Scheduling web-app source code for Kiss., Schick., and Schneider. [KSS19].

References I

[Aum17] Jean-Philippe Aumasson. ***Serious cryptography: a practical introduction to modern encryption.*** No Starch Press, 2017.

[Chi19] Ilaria Chillotti. ***The Three Musketeers of Secure Computation: MPC, FHE and FE.*** Computer Security and Industrial Cryptography group at KU Leuven. July 17, 2019. URL: https://www.esat.kuleuven.be/cosic/blog/the-three-musketeers-of-secure-computation-mpc-fhe-and-fe/ (visited on 04/25/2022).

[HPS08] Jeffrey Hoffstein, Jill Catherine Pipher, and Joseph H. Silverman. ***An introduction to mathematical cryptography.*** Undergraduate Texts in Mathematics. New York: Springer, 2008.

References II

[KSS19] Ágnes Kiss., Oliver Schick., and Thomas Schneider. **"Web Application for Privacy-preserving Scheduling using Secure Computation."** In: *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications - SECRYPT*, INSTICC. SciTePress, 2019, pp. 456–463. DOI: 10.5220/0007947704560463.

[Mic19] Microsoft Research. ***Securing the vote with Dr. Josh Benaloh.*** (Podcast transcript). Feb. 27, 2019. URL: https://www.microsoft.com/en-us/research/podcast/securing-the-vote-with-dr-josh-benaloh/ (visited on 04/25/2022).

[NAS18] National Academies of Sciences, Engineering, and Medicine. ***Securing the Vote.*** National Academies Press, 2018. DOI: 10.17226/25120. URL: https://doi.org/10.17226/25120.

References III

[Yao82] Andrew C Yao. **“Protocols for secure computations.”** In: *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE. 1982, pp. 160–164.