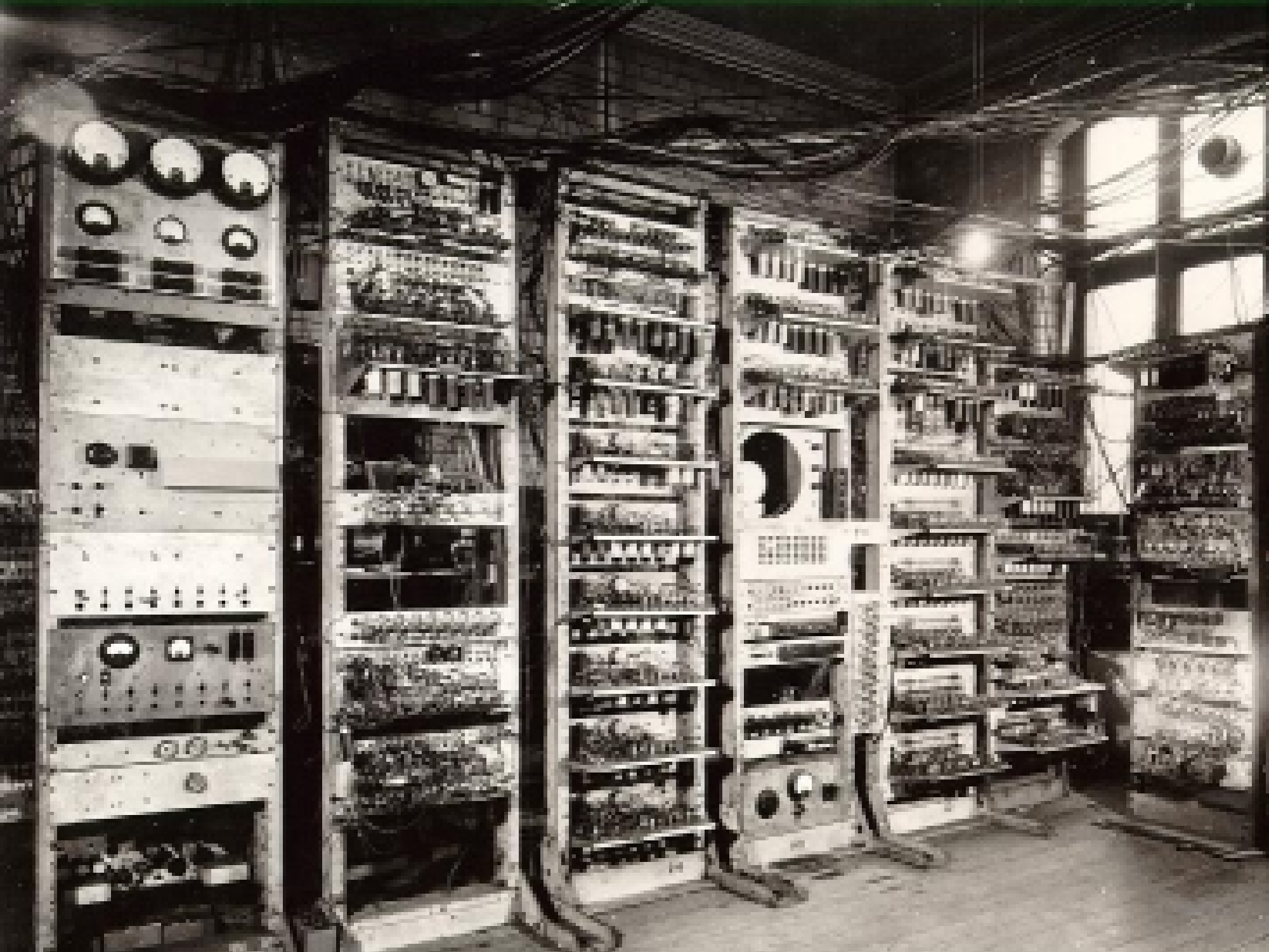
Post-quantum cryptography

Daniel J. Bernstein

Turing, 1950

"I have set up on the Manchester computer a small programme using only 1000 units of storage, whereby the machine supplied with one sixteen figure number replies with another within two seconds. I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values."



Turing, 1950

"I have set up on the Manchester computer a small programme using only 1000 units of storage, whereby the machine supplied with one sixteen figure number replies with another within two seconds. I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values."

Let's try playing this game . . .

Let's try playing this game . . .

How long do we have to figure out the pattern?

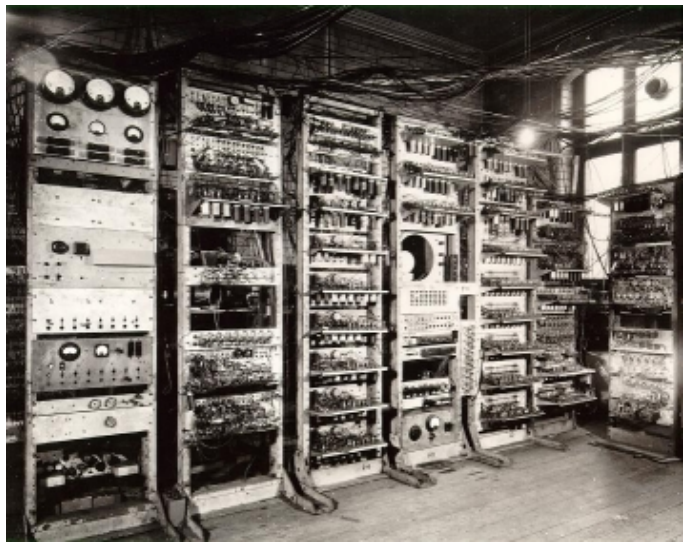
Let's try playing this game . . .

How long do we have to figure out the pattern?

Turing: "... within a reasonable time,
say a **thousand years** ..." (emphasis added)

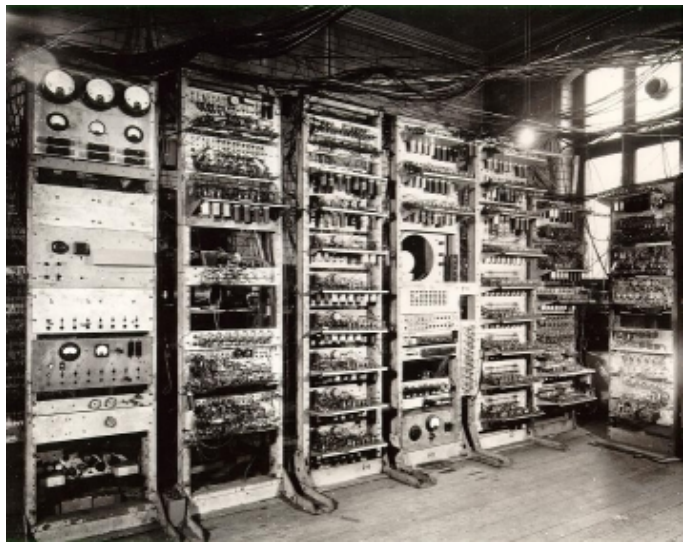
An input

0000000000000000



An input and a response

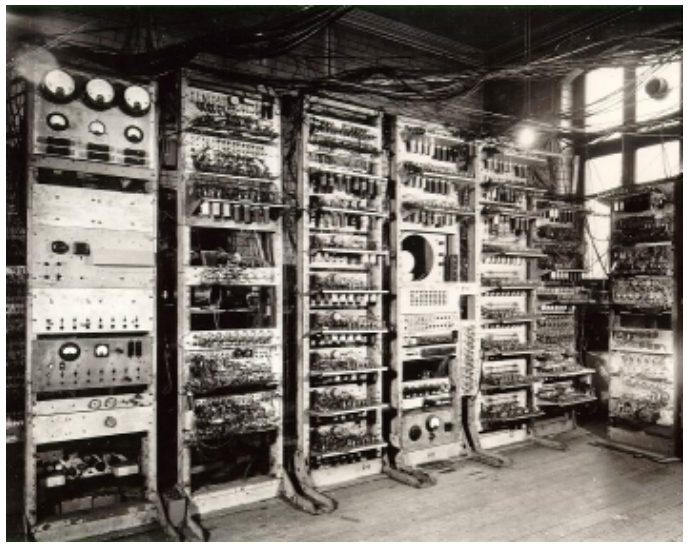
0000000000000000



2771478205812714

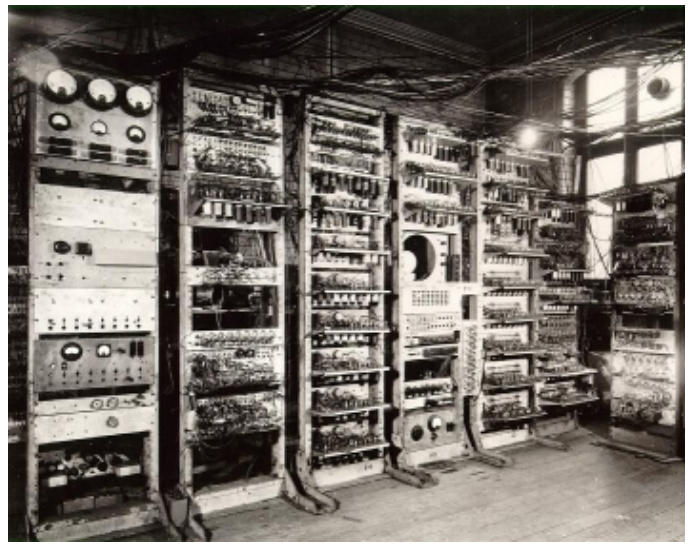
Same input again

0000000000000000



Same input again $\Rightarrow$ same response again

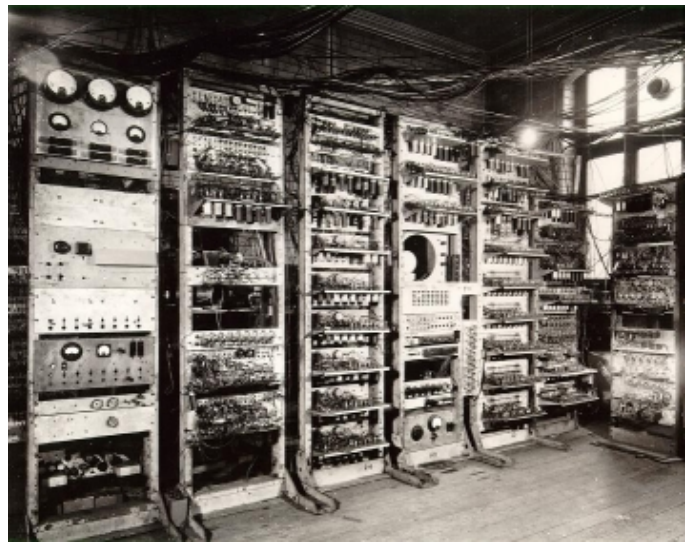
0000000000000000



2771478205812714

Another input and a response

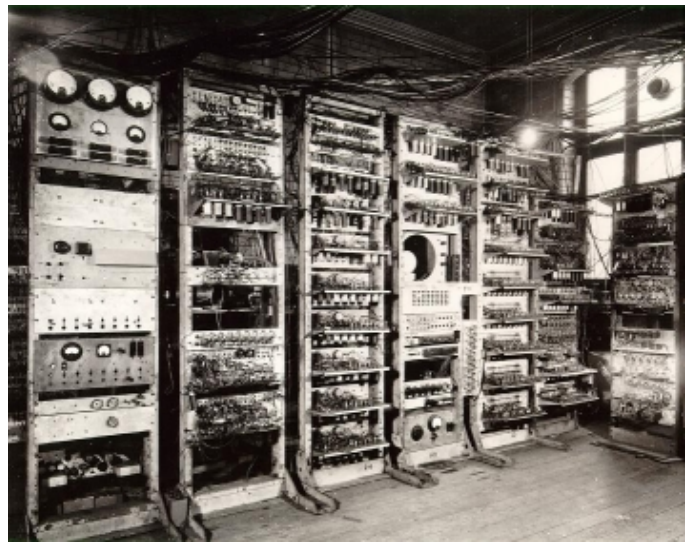
0000000000000001



1993902994537966

Another input and a response

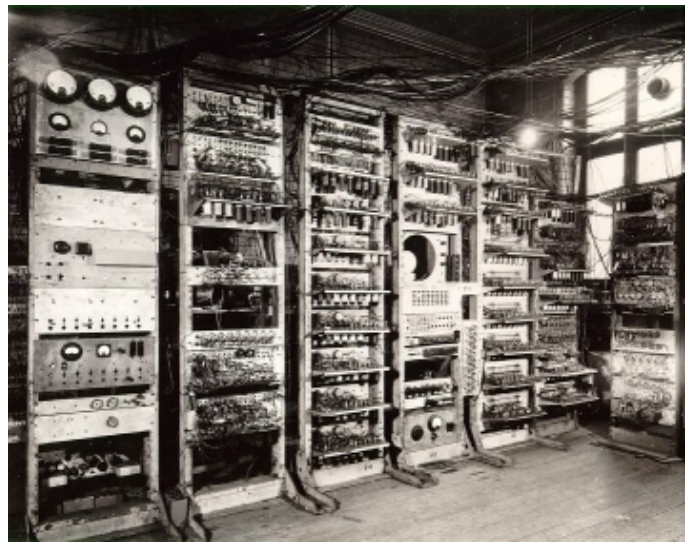
0000000000000002



0047824705410258

Another input and a response

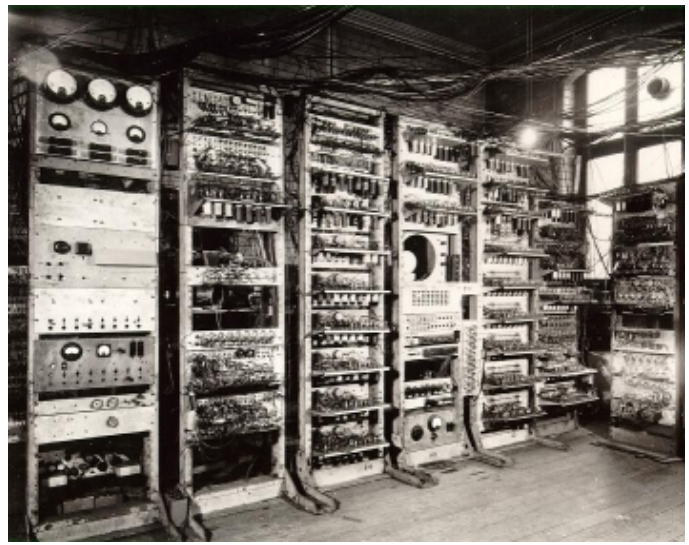
0000000000000003



709942513952598 9

Another input and a response

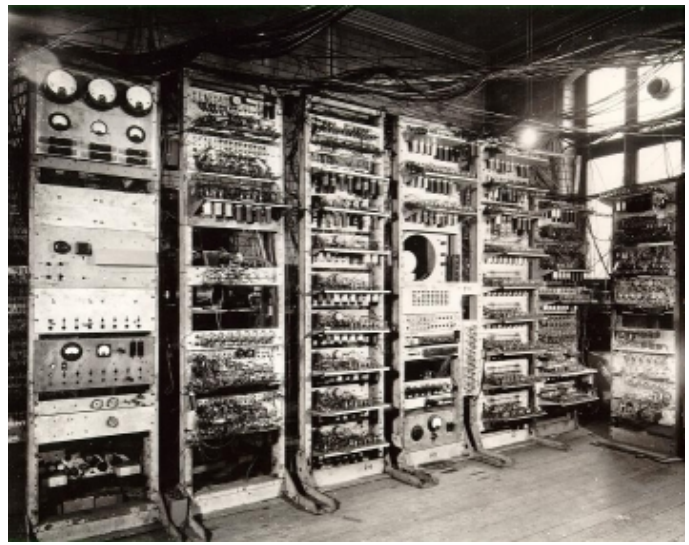
9999999999999999



2263574462999230

Another input and a response

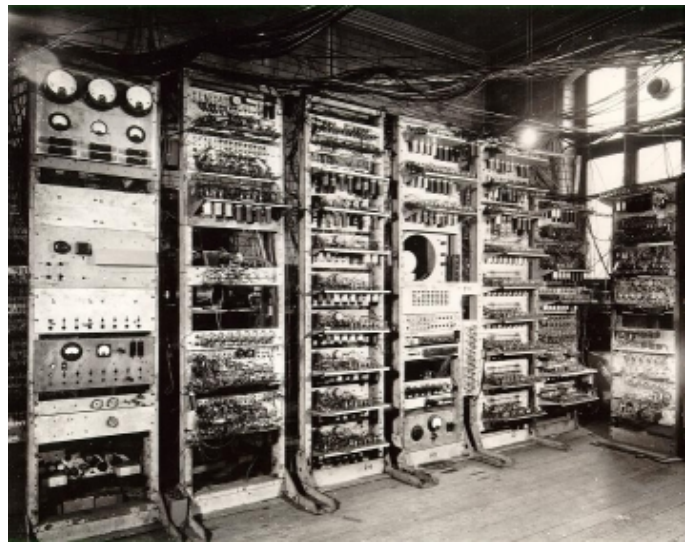
1234567890123456



687519190096677l

Another input and a response

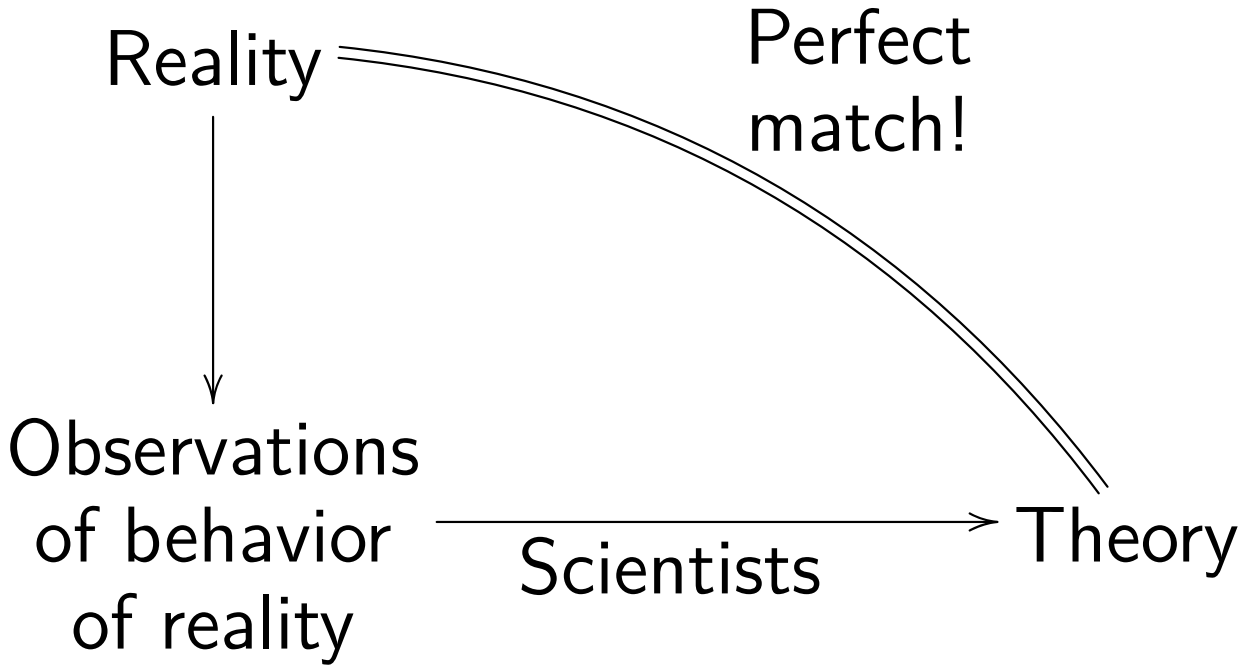
2718281828459045



039645941536756 3

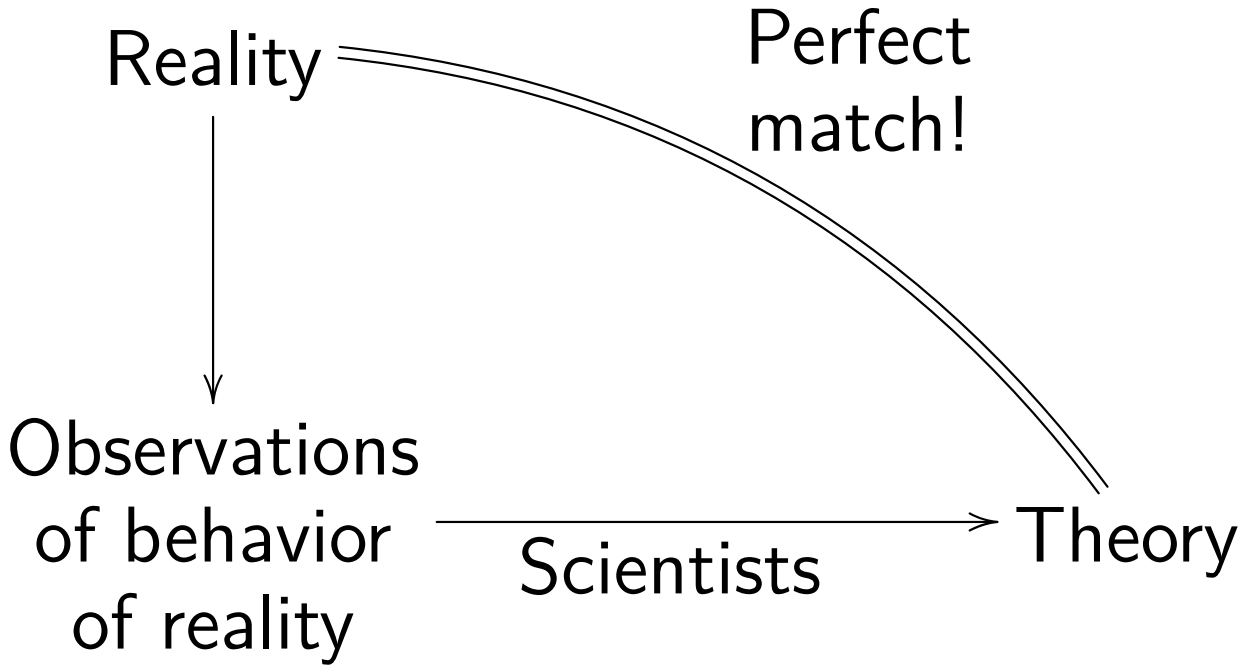
Why is this game important?

Optimistic view of science:



Why is this game important?

Optimistic view of science:



Turing is saying: This doesn't always work.

Why is this game important?

Turing predicts: We will be able to build a computer so that the computer's responses to text messages are indistinguishable from a human's responses.

Why is this game important?

Turing predicts: We will be able to build a computer so that the computer's responses to text messages are indistinguishable from a human's responses.

Objection:
1. We can figure out machines from their behavior.

Why is this game important?

Turing predicts: We will be able to build a computer so that the computer's responses to text messages are indistinguishable from a human's responses.

Objection:

1. We can figure out machines from their behavior.
2. We cannot figure out humans.

Why is this game important?

Turing predicts: We will be able to build a computer so that the computer's responses to text messages are indistinguishable from a human's responses.

Objection:

1. We can figure out machines from their behavior.
2. We cannot figure out humans.
3. Ergo, humans do not behave like machines.

Why is this game important?

Turing predicts: We will be able to build a computer so that the computer's responses to text messages are indistinguishable from a human's responses.

Objection:

1. We can figure out machines from their behavior.
2. We cannot figure out humans.
3. Ergo, humans do not behave like machines.

Turing's response: #1 doesn't always work.

A strategy to beat Turing at his own game

1. Build a computer that imitates a human.

Success! We can't tell the difference.

A strategy to beat Turing at his own game

1. Build a computer that imitates a human.
Success! We can't tell the difference.

2. Build a computer that imitates Turing.
Success! We can't tell the difference.

A strategy to beat Turing at his own game

1. Build a computer that imitates a human.
Success! We can't tell the difference.

2. Build a computer that imitates Turing.
Success! We can't tell the difference.

3. Ask the computer to produce Turing's program.
Success! We now have a copy of Turing's program.

A strategy to beat Turing at his own game

1. Build a computer that imitates a human.
Success! We can't tell the difference.

2. Build a computer that imitates Turing.
Success! We can't tell the difference.

3. Ask the computer to produce Turing's program.
Success! We now have a copy of Turing's program.

4. Run our copy of the program on more inputs.
Success! We've won the game.

This strategy doesn't work

Turing generated a random number.
His program uses that number in the
secret computations producing each response.

This strategy doesn't work

Turing generated a random number.
His program uses that number in the
secret computations producing each response.

If we build a full simulation of the Earth,
including a complete simulation of Turing,
our simulation of Turing's program
will have a **new random number**.

The program I actually used

```
import hashlib, codecs

def hash(seed):
  h = hashlib.sha512()
  h.update(seed.encode('utf8'))
  return h.digest()

def response(input):
  secret = '935022901194106739696580346090'
  h = hash(secret + str(input) + secret)
  i = int(codecs.encode(h,'hex'),16)
  return str(i)[-16:]
```

Why is this game important? (part 2)

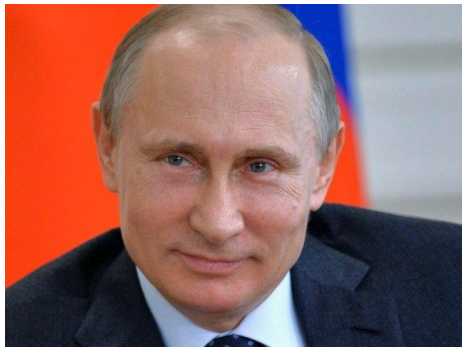
Alice sends vote tally (15117 yes, 42682 no) to Bob:

0001511700042682 →

Why is this game important? (part 2)

Alice sends vote tally (15117 yes, 42682 no) to Bob:

0001511700042682

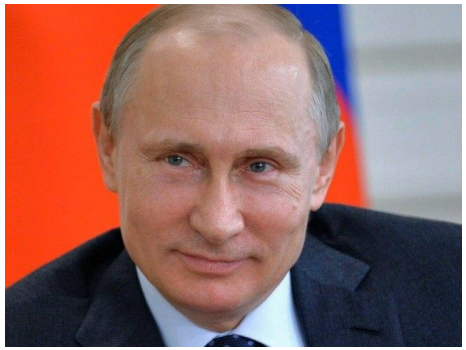


Network between Alice and Bob has been hacked.

Why is this game important? (part 2)

How does Bob know this message is from Alice?

0001511700042682

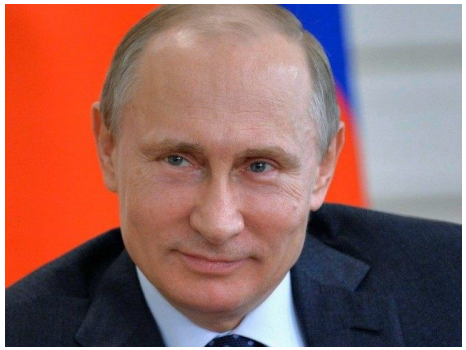


Network between Alice and Bob has been hacked.

Why is this game important? (part 2)

How does Bob know this message is from Alice?
Alice includes an extra number with the message.

881768974780900400015117000426 82



Network between Alice and Bob has been hacked.

Why is this game important? (part 2)

Alice's extra number comes from the program.

000151170042682

8817689747809004

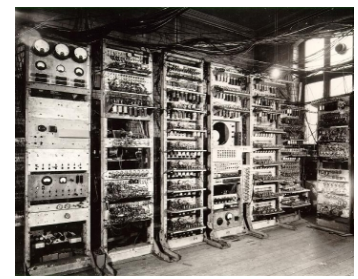
Why is this game important? (part 2)

Alice's extra number comes from the program.

0001511700042682

8817689747809004

0001511700042682



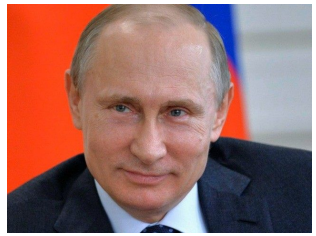
8817689747809004

Post-quantum cryptography

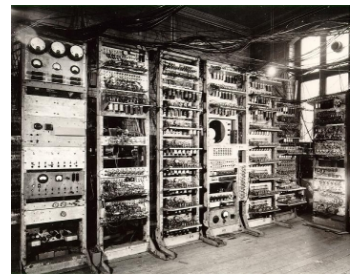
Daniel J. Bernstein

Why is this game important? (part 2)

Alice's extra number comes from the program.



000151170042682 000161170041682 000151170042682



881768974809004 ? 881768974809004

Where do Alice and Bob get the program?

They don't have copies of Turing's program.

Where do Alice and Bob get the program?

They don't have copies of Turing's program.

They have my program, but attacker has it too.

Where do Alice and Bob get the program?

They don't have copies of Turing's program.

They have my program, but attacker has it too.

They could make their own program.
Why do they think the outputs are hard to predict?

Where do Alice and Bob get the program?

They don't have copies of Turing's program.

They have my program, but attacker has it too.

They could make their own program.
Why do they think the outputs are hard to predict?
Need security auditors saying "This is hard".

Where do Alice and Bob get the program?

They don't have copies of Turing's program.

They have my program, but attacker has it too.

They could make their own program.
Why do they think the outputs are hard to predict?
Need security auditors saying "This is hard".

Solution: Alice and Bob share a **secret key.**
Key = random number inserted into my program.
My program is published. Security audits are public.

Are we really worried about forgeries?

Vote tallies are published through many channels.
Surely any discrepancies will be noticed.

Are we really worried about forgeries?

Vote tallies are published through many channels.
Surely any discrepancies will be noticed.

But attackers use false information in other ways:
e.g., hacking into computers via forged email,
forged operating-system updates, etc.

Are we really worried about forgeries?

Vote tallies are published through many channels. *Surely* any discrepancies will be noticed.

But attackers use false information in other ways: e.g., hacking into computers via forged email, forged operating-system updates, etc.

Often false information is corrected too late.

Are we really worried about forgeries?

Vote tallies are published through many channels. *Surely* any discrepancies will be noticed.

But attackers use false information in other ways: e.g., hacking into computers via forged email, forged operating-system updates, etc.

Often false information is corrected too late.

"The Russian government has sought to influence democracy in the United Kingdom through disinformation, cyber hacking, and corruption."

Canada

Rising clarinet star's ex-girlfriend must pay $375,000 for trying to sabotage his career

Ontario court calls case of Eric Abramovitz, whose girlfriend faked a rejection letter from his dream school, 'despicable interference'

Why is this game important? (part 3)

confidential
5572318944361249

Why is this game important? (part 3)

confidential
5572318944361249

random input
4038578500540991

3097310635297394

Why is this game important? (part 3)

confidential
5572318944361249

random input
4038578500540991

3097310635297394

add; keep
last 16 digits

8669629579658643

Why is this game important? (part 3)

confidential
557231894436124 9

random input
403857850054099 1

309731063529739 4

add; keep
last 16 digits

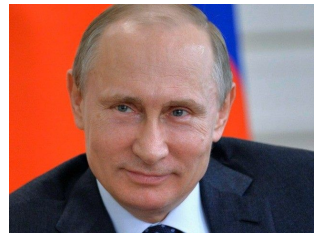
866962957965864 3

Why is this game important? (part 3)

confidential
5572318944361249

random input
4038578500540991

3097310635297394



add; keep
last 16 digits

8669629579658643

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.
Nazis broke secrecy of some Allied communication.

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.
Nazis broke secrecy of some Allied communication.
Many more failures of communication secrecy.

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.
Nazis broke secrecy of some Allied communication.
Many more failures of communication secrecy.

But Turing's program allows secret communication.

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.
Nazis broke secrecy of some Allied communication.
Many more failures of communication secrecy.

But Turing's program allows secret communication.

Or does it? Yes if Turing was right, but
maybe he missed a way to predict the responses.

Wasn't Turing breaking German ciphers?

Turing broke secrecy of some Nazi communication.
Nazis broke secrecy of some Allied communication.
Many more failures of communication secrecy.

But Turing's program allows secret communication.

Or does it? Yes if Turing was right, but
maybe he missed a way to predict the responses.

Turing never published the program.

The public science of cryptography

By late 1970s: Cryptographic research publications included functions that seem totally unpredictable.

The public science of cryptography

By late 1970s: Cryptographic research publications included functions that seem totally unpredictable.

Also included a huge advance in usability: **public-key cryptography**.

Alice and Bob don't need to meet to share a secret. Instead share secret through public communication.

Billions of cryptographic users today

amazon

Good cryptography takes time to build

Many stages of research from design to deployment:

- Explore space of cryptosystems.
- Study algorithms for the attackers.
- Focus on secure cryptosystems.

Good cryptography takes time to build

Many stages of research from design to deployment:

- Explore space of cryptosystems.
- Study algorithms for the attackers.
- Focus on secure cryptosystems.
- Study algorithms for the users.
- Study implementations on real hardware.
- Study side-channel attacks, fault attacks, etc.
- Focus on secure, reliable implementations.
- Focus on implementations meeting performance requirements.
- Integrate securely into real-world applications.

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We thus give the first examples of quantum cryptanalysis.)

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical computer scientists generally classify algorithms as efficient when the number of steps of the algorithms grows as a polynomial in the size of the input. The class of prob-

The quantum apocalypse

Today: Massive usage of RSA-2048 and ECC-256 to protect against espionage and sabotage.

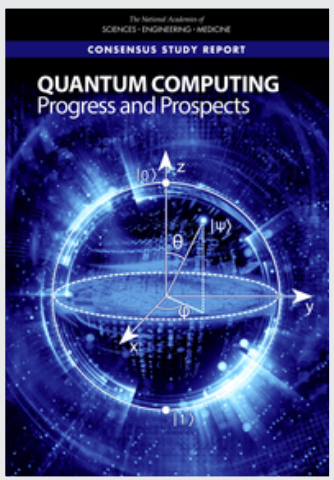
But RSA-2048 and ECC-256 will be broken by any attacker who builds a quantum computer.

The quantum apocalypse

Today: Massive usage of RSA-2048 and ECC-256
to protect against espionage and sabotage.

But RSA-2048 and ECC-256 will be broken
by any attacker who builds a quantum computer.

Attackers are recording encrypted data today.
Will decrypt once they have a quantum computer.
("Perfect forward secrecy" does not prevent this.)



Quantum Computing: Progress and Prospects (2018)

DETAILS

202 pages | 6 x 9 | PAPERBACK
ISBN 978-0-309-47969-1 | DOI 10.17226/25196

CONTRIBUTORS

Emily Grumbling and Mark Horowitz, Editors; Committee on Technical Assessment of the Feasibility and Implications of Quantum Computing; Computer Science and Telecommunications Board; Intelligence Community Studies Board; Division on Engineering and Physical Sciences; National Academies of Sciences, Engineering, and Medicine

nap.edu report on quantum computing

Don't panic. "Key Finding 1: Given the current state of quantum computing and recent rates of progress, it is highly unexpected that a quantum computer that can compromise RSA 2048 or comparable discrete logarithm-based public key cryptosystems will be built within the next decade."

nap.edu report on quantum computing

Panic. "Key Finding 10: Even if a quantum computer that can decrypt current cryptographic ciphers is more than a decade off, the hazard of such a machine is high enough—and the time frame for transitioning to a new security protocol is sufficiently long and uncertain—that prioritization of the development, standardization, and deployment of post-quantum cryptography is critical for minimizing the chance of a potential security and privacy disaster."

Post-quantum cryptography

Cryptography designed under the assumption that the attacker has a large quantum computer.

Cryptographic researchers plan ahead

PQCrypto 2006: International Workshop
on Post-Quantum Cryptography.

Cryptographic researchers plan ahead

PQCrypto 2006: International Workshop on Post-Quantum Cryptography.

PQCrypto 2008.

Cryptographic researchers plan ahead

PQCrypto 2006: International Workshop on Post-Quantum Cryptography.

PQCrypto 2008.

PQCrypto 2010.

Cryptographic researchers plan ahead

PQCrypto 2006: International Workshop on Post-Quantum Cryptography.

PQCrypto 2008.

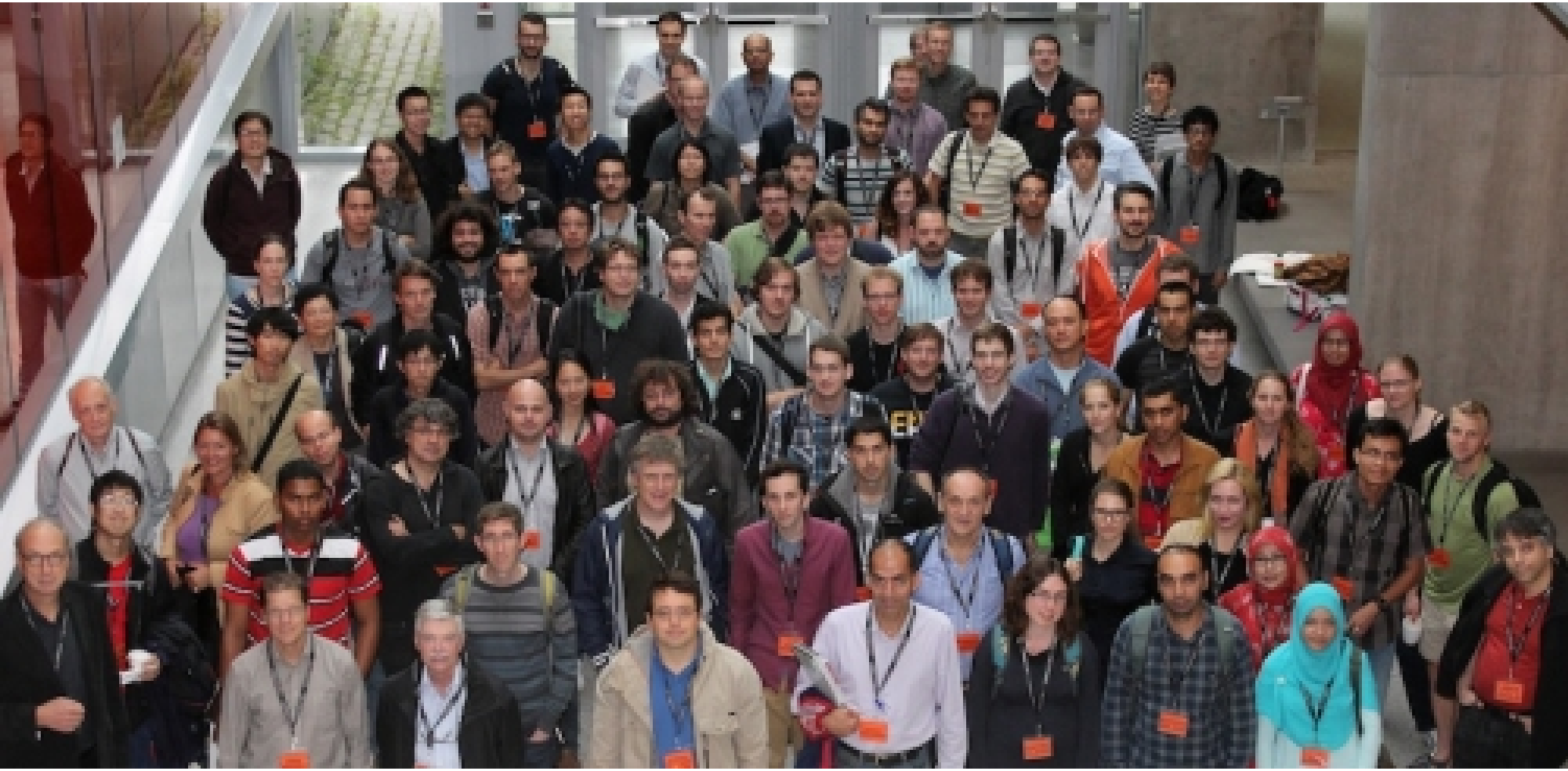
PQCrypto 2010.

PQCrypto 2011.

PQCrypto 2013.

PQCrypto 2014.

PQCrypto 2014 participants



Activity heats up

EU funds three-year PQCRYPTO project.

NSA issues a statement.

PQCrypto 2016.

Google starts a post-quantum experiment.

NCSC UK issues a statement.

NIST calls for submissions to "Post-Quantum Cryptography Standardization Project".

PQCrypto 2017.

PQCrypto 2018 + NIST conference.

PQCrypto 2016 participants

PQCrypto 2018 participants

In December 2017 . . .

NIST posts 69 submissions from 260 people.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

In December 2017 ... there were attacks

By end of 2017: 8 out of 69 submissions attacked.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

What is going on here?

By end of 2018: 22 out of 69 submissions attacked.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

An attempt to explain the situation

People often categorize submissions. e.g.:

- Code-based encryption and signatures.
- Hash-based signatures.
- Isogeny-based encryption.
- Lattice-based encryption and signatures.
- Multivariate-quadratic encryption and signatures.

An attempt to explain the situation

"What's safe is lattice-based cryptography."
— Are you sure about that?

An attempt to explain the situation

"What's safe is lattice-based cryptography."
— Are you sure about that?

Lattice-based submissions: Compact LWE.
CRYSTALS-DILITHIUM. CRYSTALS-KYBER.
Ding Key Exchange. DRS. EMBLEM and R.EMBLEM. FALCON.
FrodoKEM. HILA5. KINDI. LAC. LIMA. Lizard. LOTUS.
NewHope. NTRUEncrypt. NTRU-HRSS-KEM. NTRU Prime.
Odd Manhattan. OKCN/AKCN/CNKE. pqNTRUSign. qTESLA.
Round2. SABER. Titanium.

An attempt to explain the situation

"What's safe is lattice-based cryptography."
— Are you sure about that?

Lattice-based submissions: Compact LWE.
CRYSTALS-DILITHIUM. CRYSTALS-KYBER.
Ding Key Exchange. DRS. EMBLEM and R.EMBLEM. FALCON.
FrodoKEM. HILA5. KINDI. LAC. LIMA. Lizard. LOTUS.
NewHope. NTRUEncrypt. NTRU-HRSS-KEM. NTRU Prime.
Odd Manhattan. OKCN/AKCN/CNKE. pqNTRUSign. qTESLA.
Round2. SABER. Titanium.

Important progress in lattice attacks this decade—
even in the past year. Maybe none of these are safe.

Details matter

4 August 2018: Round5 merges HILA5 and Round2.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

Details matter

4 August 2018: Round5 merges HILA5 and Round2.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

24 August: Security failure announced in Round5.

Details matter

4 August 2018: Round5 merges HILA5 and Round2.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

24 August: Security failure announced in Round5.

Round5 response: "proposed fix" ... "looking at the security proof adjustments" ... "The actual Round5 proposal to NIST is still months away."

Another attempt to explain the situation

"What's safe is using the portfolio from the European PQCRYPTO project."
— Are you sure about that?

Another attempt to explain the situation

"What's safe is using the portfolio from the European PQCRYPTO project."
— Are you sure about that?

The portfolio: BIG QUAKE. BIKE. Classic McEliece. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. FrodoKEM. Gui. KINDI. LUOV. MQDSS. NewHope. NTRU-HRSS-KEM. NTRU Prime. Picnic. qTESLA. Rainbow. Ramstake. SABER. SPHINCS+.

Security auditors are overloaded

69 submissions = **denial-of-service attack against security auditing**.

Maybe the auditors have been focusing on submissions from outside the PQCRYPTO project.

Computer Security Resource Center

Due to the lapse in government funding, csrc.nist.gov and all associated online activities will be unavailable until further notice. Learn more.

30 Jan 2019: NIST announces round 2

Code enc: BIKE. Classic McEliece. HQC. LEDAcrypt (LEDAkem + LEDApkc). NTS-KEM. ROLLO (LAKE + LOCKER + Ouroboros-R). RQC.

Lattice enc: FrodoKEM. KYBER. LAC. NewHope. NTRU (NTRUEncrypt + NTRU-HRSS-KEM). NTRU Prime. Round5 (HILA5 + Round2). SABER.

Other encryption: SIKE. Three Bears.

Lattice sig: DILITHIUM. FALCON. qTESLA.

MQ sig: GeMSS. LUOV. MQDSS. Rainbow.

Other signatures: Picnic. SPHINCS+.