

# Who Moved my Rock?

Post-Quantum Cryptography and its Impact on Higher  
Education

We live in an age of misinformation, fear, uncertainty, and doubt. I put together this talk to discuss where we are with Post-Quantum Cryptography and whether it is time to panic or not.

# Who am I?

Brian Epstein (he/him) - [bepstein@ias.edu](mailto:bepstein@ias.edu)

Institute for Advanced Study - [ias.edu/security](https://ias.edu/security)

IT Manager, Network and Security  
Chief Information Security Officer

Mastodon: [infosec.exchange/@ep](https://infosec.exchange/@ep)

X: [@epepepep](https://twitter.com/epepepep)



My name is Brian Epstein, my pronouns are he him. I'm not a cryptographer, but I have been a practitioner of cryptography for over two decades in my career in IT. I currently work at the Institute for Advanced Study in Princeton, NJ and serve the role as IT Manager for Network and Security and the Chief Information Security Officer. The Institute for Advanced Study promotes the disinterested pursuit of knowledge unburdened from distraction for our scholars.



I always try to tie in the main tenets of Information Security into my talks. Today we'll be focusing mainly on Confidentiality and Integrity.

Quantum Comput... x China's new quan... x Researcher Claim... x Debunking The To... x NASA Just Shut D... x Guest Post: Harve... x +

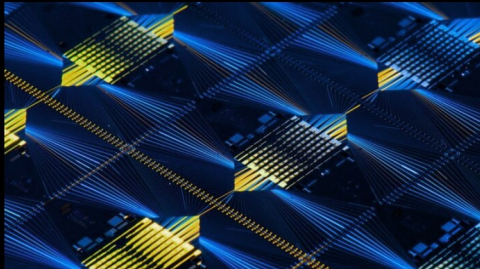
singularityhub.com/2023/10/02/quantum-computers-could-crack-encryption-sooner-than-ex...

singularity group singularity community f @ X


Singularity hub TOPICS EXPERTS EVENTS VIDEOS

# Quantum Computers Could Crack Encryption Sooner Than Expected With New Algorithm

By Edd Gent > October 2, 2023



FEATURED



Mice Just Passed the Mirror Test. Here's What That Says About Our Sense of Self

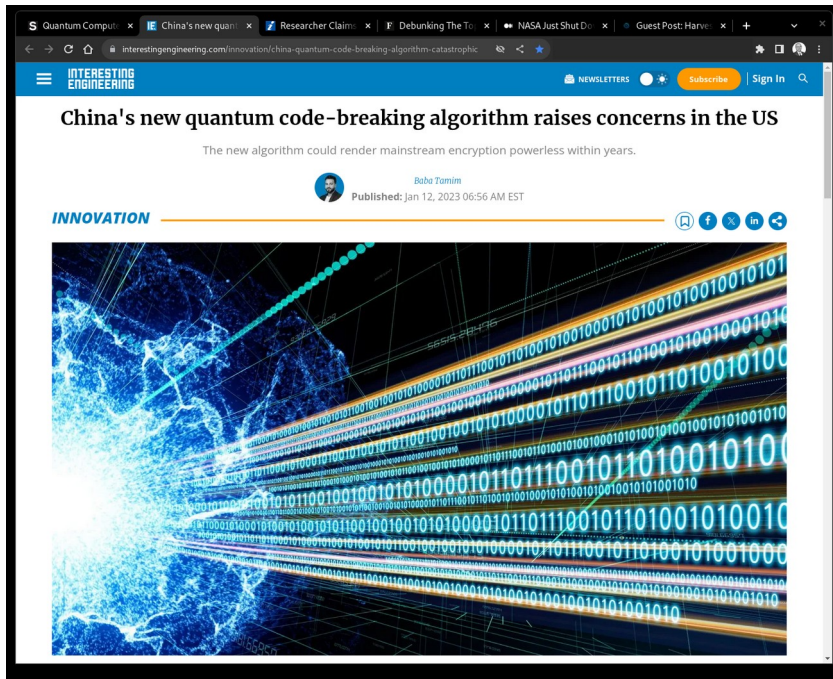
December 7, 2023

Load more >

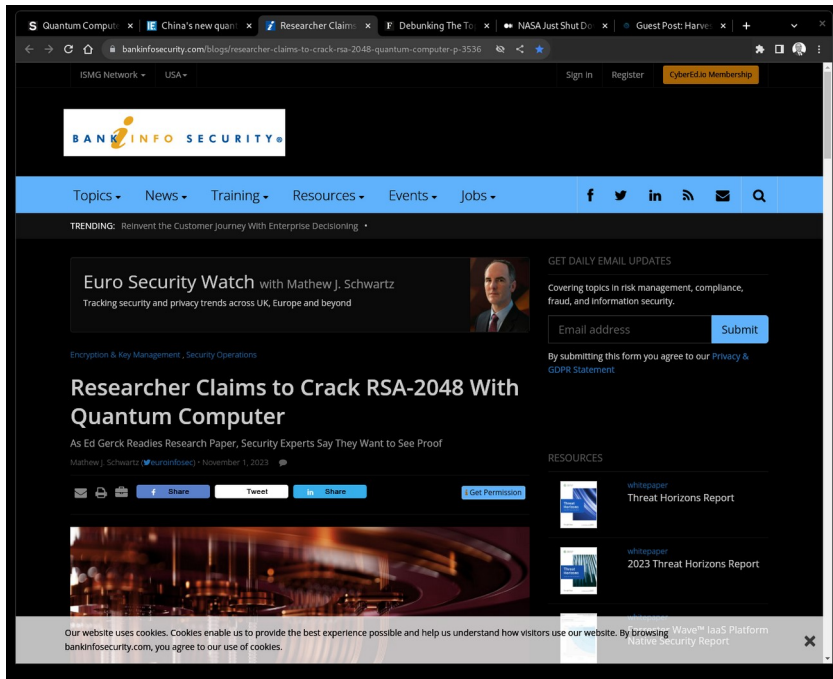
One of the most well-established and disruptive uses for a future quantum computer is the ability to crack encryption. A new algorithm could significantly lower the barrier to achieving this.

Despite all the hype around quantum computing, there are still significant question marks around **what quantum computers will actually be useful for**. There are hopes they could accelerate everything from

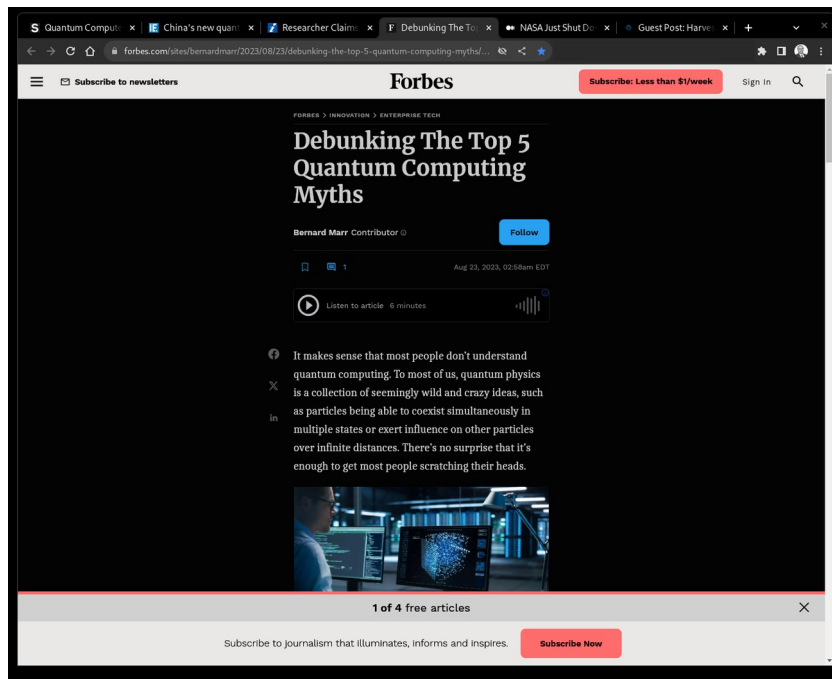
It's hard to determine what is truth and what is fiction in today's quantum news.



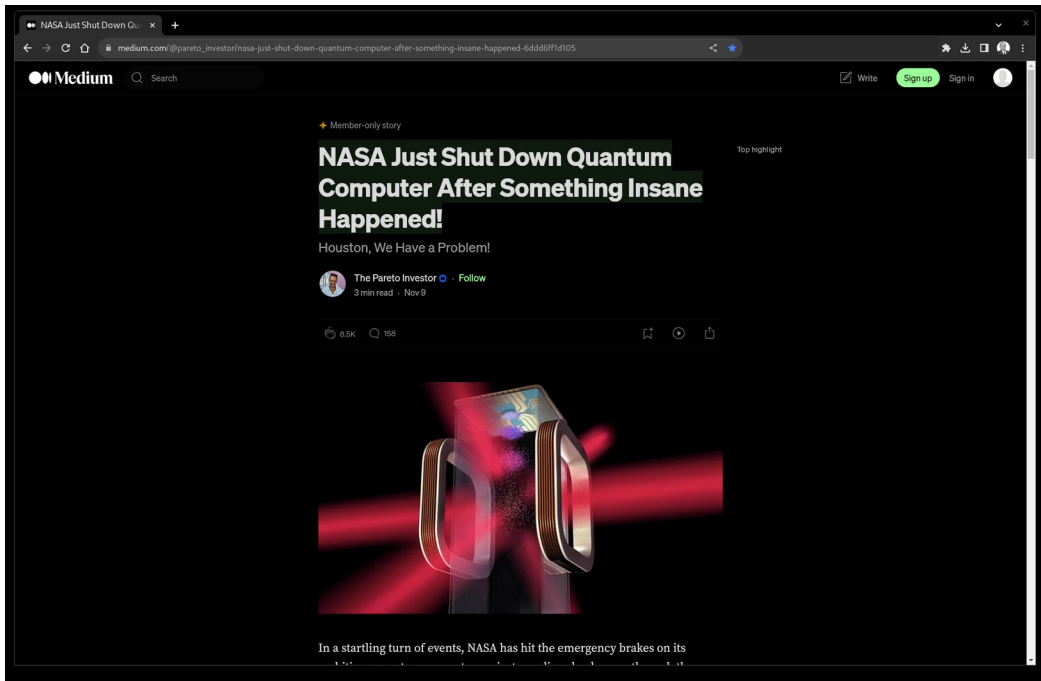
Headlines like these come out on a daily basis.



Sometimes they point to stories that are difficult to believe.



Other times, you see helpful articles that try to dispel the myths.



And sometimes you just see articles that are plain fake.




Quantum Computing | China's new quantum | Researcher Claims | Debunking The Truth | NASA Just Shut Down | Guest Post: Harvest Now, Decrypt Later? The Truth Behind This Common Quantum Theory

thequantuminsider.com/2023/02/07/guest-post-harvest-now-decrypt-later-the-truth-behind-...

NEWS | EXCLUSIVES | ABOUT US | MARKETING | REPORTS | NEWSLETTER | QUANTUM CHALLENGE

## Guest Post: Harvest Now, Decrypt Later? The Truth Behind This Common Quantum Theory

Insights | Jeffrey Duran • February 7, 2023



Harvest Now, Decrypt Later? The Truth Behind This Common Quantum Theory  
Is the Sky Really Falling?

To many, the term "quantum computing" equates to a world of new possibilities. What started as a theoretical curiosity is now touted as the future of IT. With quantum, there will be lightning fast processing of information and computers will be able to answer problems previously thought of as unsolvable.

But like the children's fable *Chicken Little*, once you get past the hype, there is a significant level of trepidation.

*One day Henny-penny was picking up corn in the rickyard when—whack!—an acorn hit her upon the head. "Goodness gracious me!" said Henny-penny, "the sky's a-going to fall! I must go and tell the King."*

<https://americanliterature.com/childrens-stories/henny-penny-the-sky-is-falling>

Relevant [More >](#)

- IonQ Announces \$2.8 Million in Revenue for 2022 Q3  
Matt Swayne • November 15, 2022
- Quantum Technology in Science Fiction & Popular Culture  
Kenna Hughes-Castiberry • July 9, 2021
- La Monde: France Pledges 1.8 Billion Euros for Quantum Technologies  
Matt Swayne • January 22, 2021
- The U.S. Department of Energy Unveiled a Blueprint Strategy to Develop a National Quantum Internet...  
Matt Swayne • July 26, 2020
- Silicon Quantum Computers Part of RIKEN and Intel's Joint Research Project  
Matt Swayne • May 24, 2023

Recommended [More >](#)

**Quantum Machine Learning**  
The Next Big Thing

Quantum Machine Learning Is The Next Big Thing

So what should we be doing to protect ourselves and our institutions?

## Characters

Hem - angry, impatient, unwilling to move

Haw - scared stiff, but thinks

Sniff - always using their tools to find something new

Scurry - always on the move to discover new things



For this talk we'll follow a cast of characters and their adventures through history. We borrow them from the popular book by Spencer Johnson, "Who Moved My Cheese".

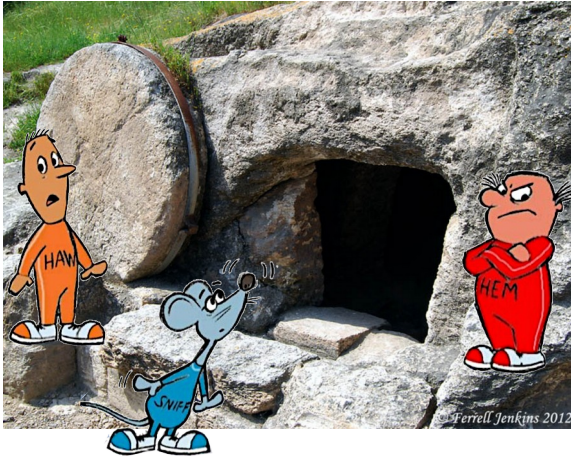
Hem - angry, impatient, unwilling to move

Haw - scared stiff, but thinks

Sniff - always using their tools to find something new

Scurry - always on the move to discover new things

# Protecting things



## First Attempt at Security

- Put large boulders in front of our caves
- Someone figured out how to move them
- Repetitive journey - find the better lock



Our first attempts <Click> at security involved hiding our stuff by moving a big boulder in front of our caves <Click>. This took a lot of work to roll the boulder and get just right.

However <Click>, someone figured out how to move our rock and get to our stuff anyway.

<Click> This scared Haw who said, "What will we do now?"

<Click> Hem was mad and said "We'll move the rock back of course. They won't figure it out again!"

<Click> Sniff knew this was a mistake, though, and

<Click> Scurry said, "Let's find a new way to protect our stuff!"

So, this is the story we have repeated throughout history <Click>. We create a lock, someone breaks it, so we make a new lock.

Let's visit some of these innovations over time.

## Protectors



We first utilized protectors to guard against a thief. Human guards were error prone and unreliable, so we started with <Click> man's best friends trained to attack intruders.

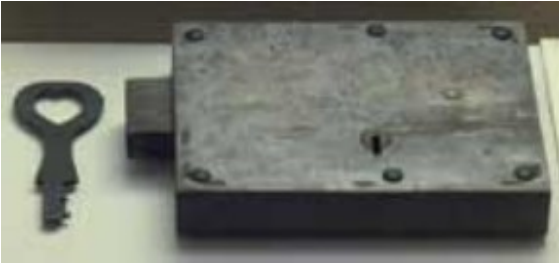
<Click> The story goes that in India, they placed their valuables in pools filled with partially starved crocodiles. In both cases, drugging or killing the animals was an effective means of bypassing the lock.

## Locks and Lockpicks



Moving to a non-living protection method was the next evolution. At first we used <Click> seal locks that would be able to show evidence of tampering. Although a deterrent, it didn't actually prevent someone from gaining access. <Click> In the 8th century BC in Persia one of the first wooden locks was used in a method very similar to a tumbler lock that we use today.

## Locks and Lockpicks



As materials improved, we moved to <Click> metallic tumbler locks and <Click> rim locks. Both proved tricky for criminals to bypass, but not impossible. Today, we see many hobbyists picking locks like these for fun.

# Value of things and information

- Information became valuable
  - War plans
  - Hidden treasure
  - Hunting / farming grounds
- Necessary to transmit
  - Get strategy to front lines
  - Remote allies
- Easy to intercept



At some point in history, we started to realize that in many circumstances, information can be much more valuable than possessions.

Information also needs to be transferred in some way or another for it to be useful.

This lead to problems with information being intercepted and used for malicious purposes.

## Hiding info in plain sight

- Novel schemes
- Useless when discovered
- Secrets can be bought



Which lead us to invent methods to hide information in plain sight.

<Click>

Many of these were novel schemes. They worked, but

<Click>

Once discovered how they worked, they were useless.

And of course, <Click> secrets can be bought.

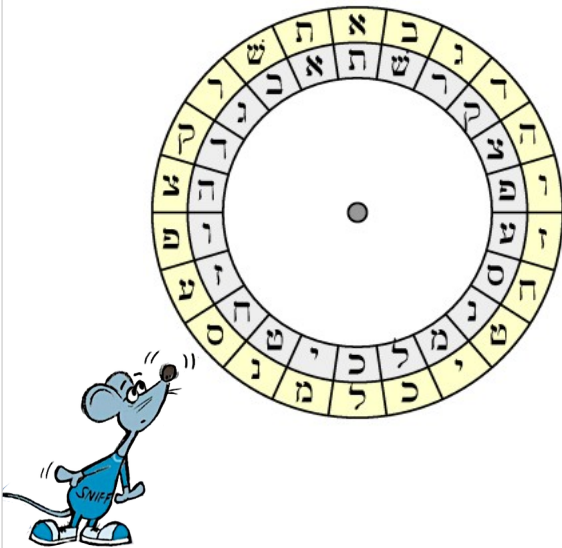


## Hiding info in plain sight



During the Classical Era <Click>, the Spartans were known to use Scytale (skit-a-lee) Cipher's to transmit message by a leather belt wrapped around a pole. There are even classical writers <Click> who indicate that in 500 BC, Histiaeus tattooed secret messages onto his slaves shaven heads, waited until their hair grew back, and then sent them to deliver the messages in secret.

## Encoding



In other parts of the world, they were coming up with their own ways of encoding secret messages, <Click> like Atbash in Israel in 500 BC, and the <Click> Polybius Square in 200 BC.

All of these methods were trivial to break once discovered how they worked. So, our heroes had to move on to something new.

# Symmetric Key Cryptography

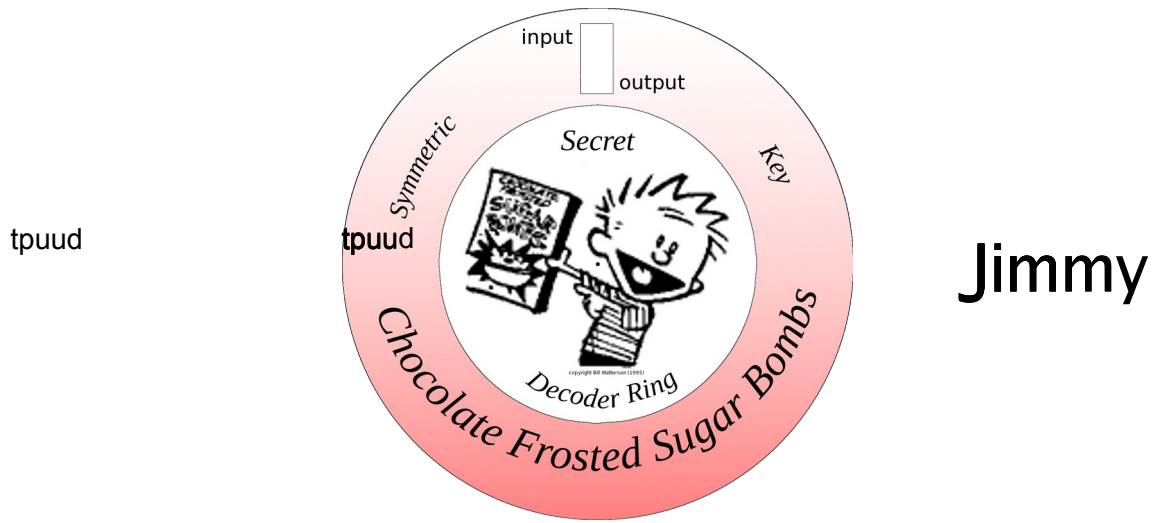
- Single key for both encryption and decryption
- Simple
- Powerful
- Key distribution issues
- Number of keys difficult

How can we hide information in a way that is known without it being discoverable? Much like a lock, we need a key that is kept secret and only the keyholders can <Click> encrypt and decrypt the message.

Having a system like this gives us many desirable <Click> properties <Click> that can help transfer secret messages.

However, how do we keep those <Click> keys safe? How do we make sure only those that should have the keys. And how to you <Click> keep track of all those keys? Every conversation you have with a different person needs a new key!

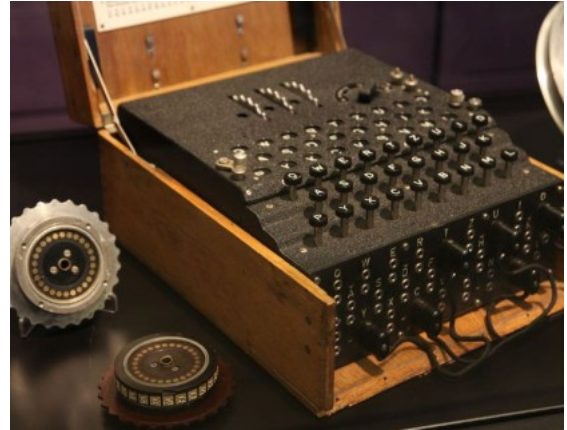
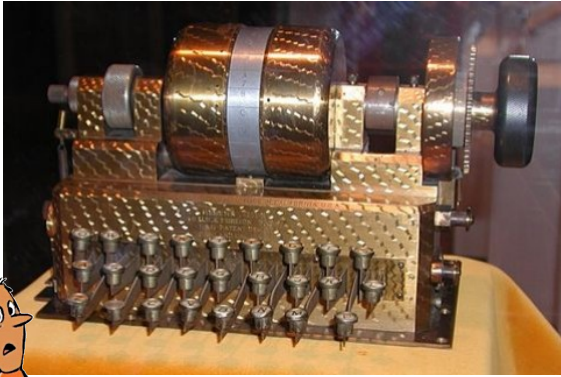
# Symmetric Key Cryptography



Here we've created a simple substitution cipher where we decrypt our simple message by rotating the dial and writing down each letter.

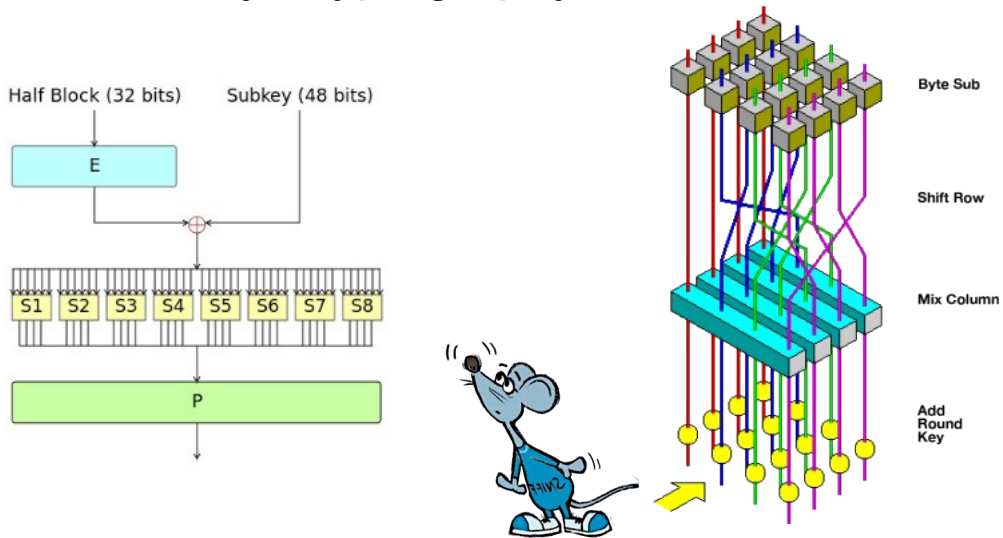
The nice thing about this type of cipher is that the same key can be used for encrypting as well. As you can see when I reverse the process.

## Symmetric Key Cryptography



Before modern computers, many types of machines were created in the early 20th century. <Click> the Hebern Rotor Machine in 1921 failed as product, only selling a few machines to the US Navy in 1931. A more famous rotor machine was invented by the Germans in 1923. <Click> The Enigma Machine was actively used for transmitting secrets during WWII. In 1932, the Enigma machine was defeated by a Polish mathematician.

# Symmetric Key Cryptography

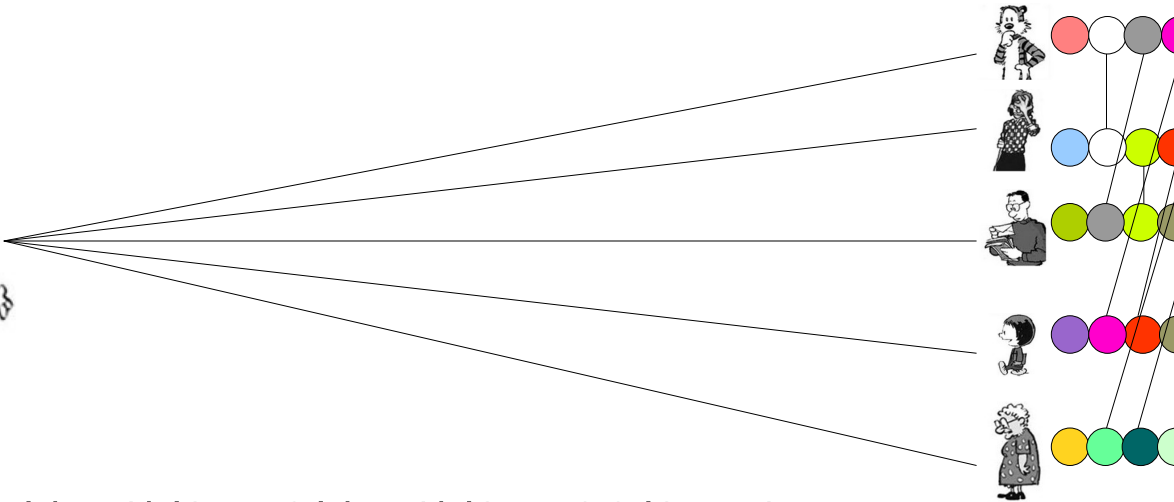
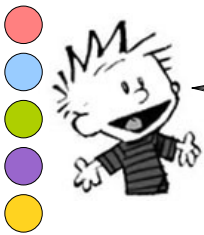


The Data Encryption Standard (DES) <Click> was created in 1975 and utilized mathematics for the scheme to encrypt and decrypt based on a private key. The NSA approved DES, but with a small key size that they could brute force if necessary. By 1999, public organizations showed that they too could break DES leading us to the use of triple DES (3DES), which is vulnerable to some theoretical attacks.

This brought us to the adoption <Click> of the Advanced Encryption Standard (AES) in 2001. AES-256 is considered to be quantum resistant as well.

Either way, the key distribution remains an issue even with AES-256, which leads us to our next topic.

# Symmetric Key Cryptography



$$n*(n-1)/2 = 6*(5-1)/2 = 30/2 = 15 \text{ unique keys}$$

May not seem significant, but remember, for 100 people, you'll need almost 5000 keys (4,950)

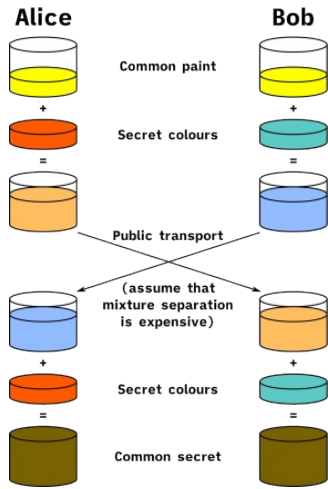
# Key Exchange

- Generate a unique secret key
- Never transfer it over the wire




The problem with Symmetric Key Cryptography is how to share the key securely. One method is to utilize a key exchange protocol. It allows you to generate a one time use, random secret key. And then transfer enough information to so that both parties can know the secret key without ever having transferred it over the wire.



# Key Exchange (Diffie-Hellman)



### Public Key Cryptography/ECDH

 <b>Bob picks</b> $\beta = 7$ <i>Bob computes</i> $B = 7G = (4, 7)$ <i>Bob receives</i> $A = (6, 7)$ <i>Bob computes</i> $\beta A = 7(5G) = 35G = 17G = (7, 10)$	 <b>Common parameters</b> $y^2 = (x^3 + 2 \cdot x + 3) \pmod{13}$ $N = 18$ $G = (7, 3)$ $B = (4, 7)$ $A = (6, 7)$	 <b>Alice picks</b> $\alpha = 5$ <i>Alice computes</i> $A = 5G = (6, 7)$ <i>Alice receives</i> $B = (4, 7)$ <i>Alice computes</i> $\alpha B = 5(7G) = 35G = 17G = (7, 10)$
---	--	---

Whitfield Diffie and Martin Hellman came up with such a <Click> scheme in 1976 utilizing discrete logarithms. In 1985 <Click>, a method of using Elliptic Curves along with Diffie-Hellman was created as well. Both are popular today. However, there are times when you want to encrypt and decrypt more than just a secret key. In those cases, Asymmetric Cryptography is used.

## Asymmetric Key Cryptography

- Split public and private keys
- Key distribution easier
- Number of keys manageable
- Slower
- Relies on a one-way function
  - Theoretical Math
  - Authentication issues

Using the idea of splitting a key into a public half shared by all, and a private half kept secret corrected a lot of issues surrounding key distribution. Now you only needed one private key and you could share your public key with the world. Unfortunately, this type of encryption is slower and is typically only used to share a secret key that is used for symmetric key encryption.

The beauty of Asymmetric Key cryptography is the use of a one-way function. This is a mathematical construct much like frying an egg. It is easy to scramble an egg, but very hard to do the opposite. Notice we never say impossible, though. There is also an issue of verifying the authenticity of a key holder, which is a problem we solve with a Public Key Infrastructure, or PKI.

# One way function



Anyone here from University of California, Irvine? Don't spoil my analogy! \*

Easy to make scrambled eggs out of eggs, but it is practically impossible to de-scramble eggs without some sort of a magic de-scrambler.

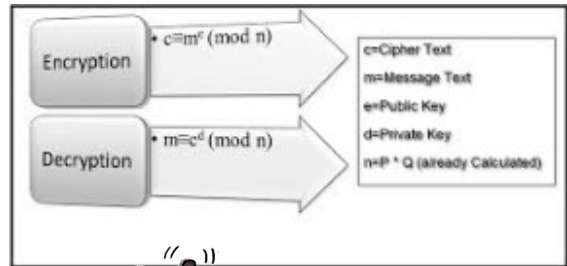
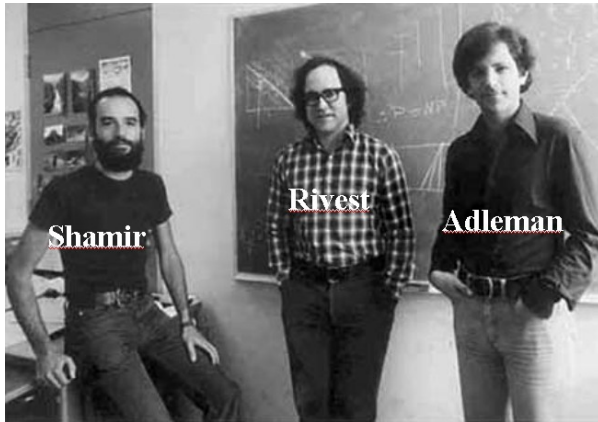
We want something similar, where it is easy to go one way, but nearly impossible to go back the other way without some other piece of knowledge.

This is exactly what Rivest, Shamir and Adleman did by using prime numbers, exponents and modulus.

We'll review these three concepts before continuing.

\* <http://news.uci.edu/press-releases/uci-fellow-chemists-find-a-way-to-unboil-eggs/>

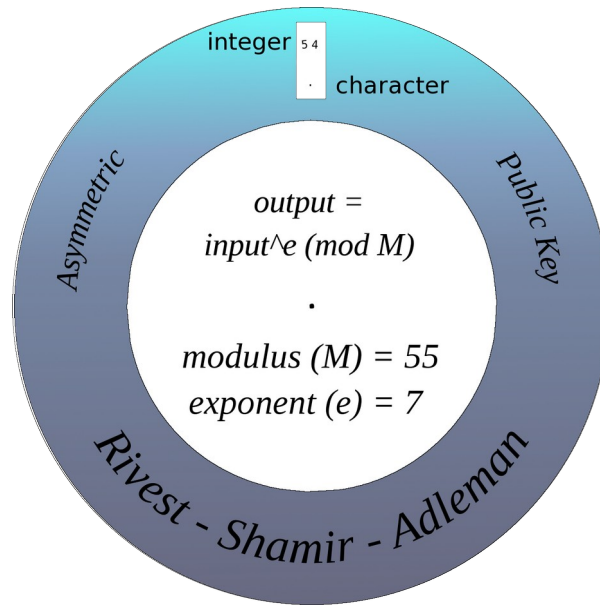
# RSA



In 1977, <Click> Ron Rivest, Adi Shamir and Leonard Adleman created the RSA algorithm. <Click> It relied on the one-way function of factoring large numbers created by multiplying two primes together, also know as a semiprime. Multiplying is easy, factoring is hard.

# One way function for RSA

Jimmy



14 13 18 18

## One way function for RSA

14 13 18 18 36

14 13 18 18

## One way function for RSA

~~14~~ 13 18 18 36

13

18

18

36

## One way function for RSA

$$14^7 \bmod 55 = 105413504 \bmod 55 = 9 =$$

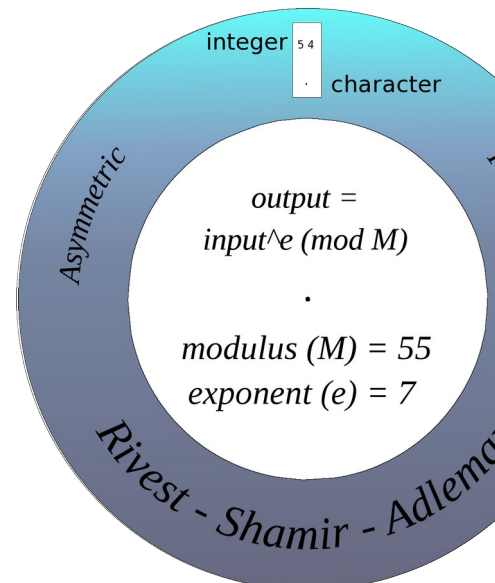
$$13^7 \bmod 55 = 7 = \text{"e"}$$

$$18^7 \bmod 55 = 17 = \text{"l"}$$

$$18^7 \bmod 55 = 17 = \text{"l"}$$

$$36^7 \bmod 55 = 31 = \text{"v"}$$

$$E(\text{Jimmy}) = \text{gellv}$$





## One way function for RSA

$$9^{23} \bmod 55 = 14 = \text{"J"}$$

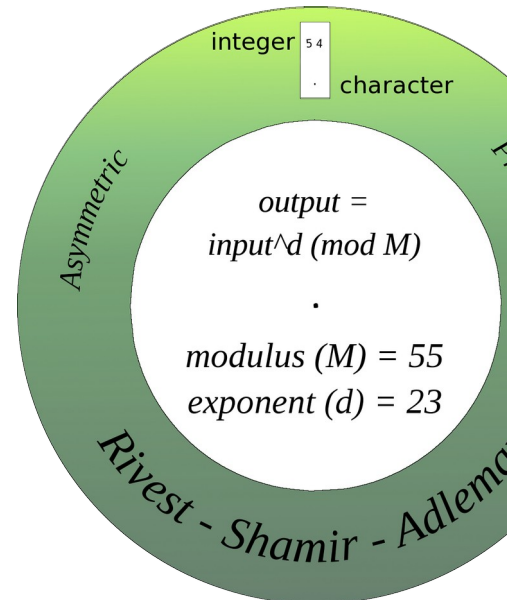
$$7^{23} \bmod 55 = 13 = \text{"i"}$$

$$17^{23} \bmod 55 = 18 = \text{"m"}$$

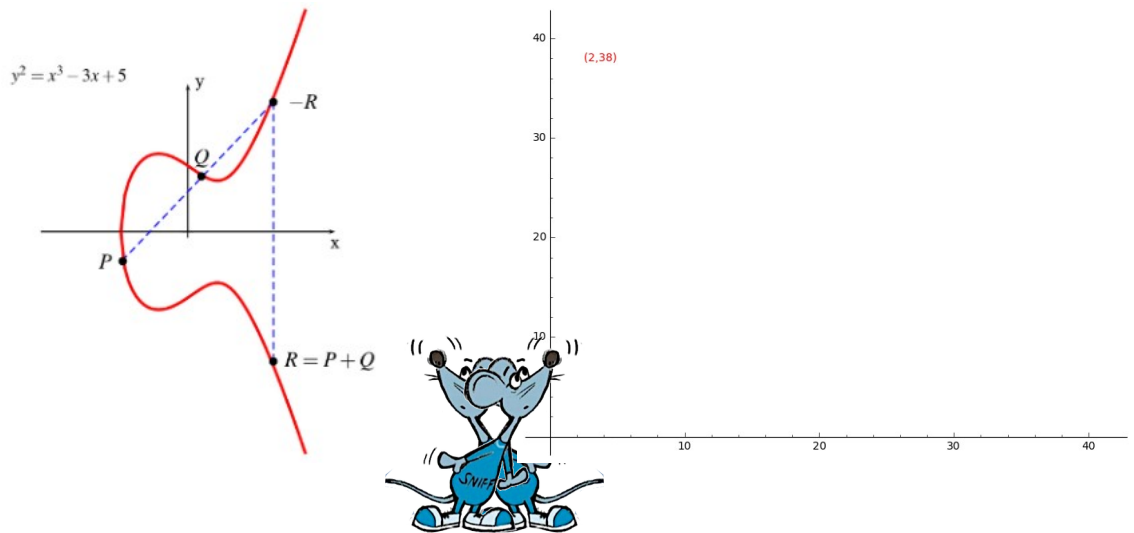
$$17^{23} \bmod 55 = 18 = \text{"m"}$$

$$31^{23} \bmod 55 = 36 = \text{"y"}$$

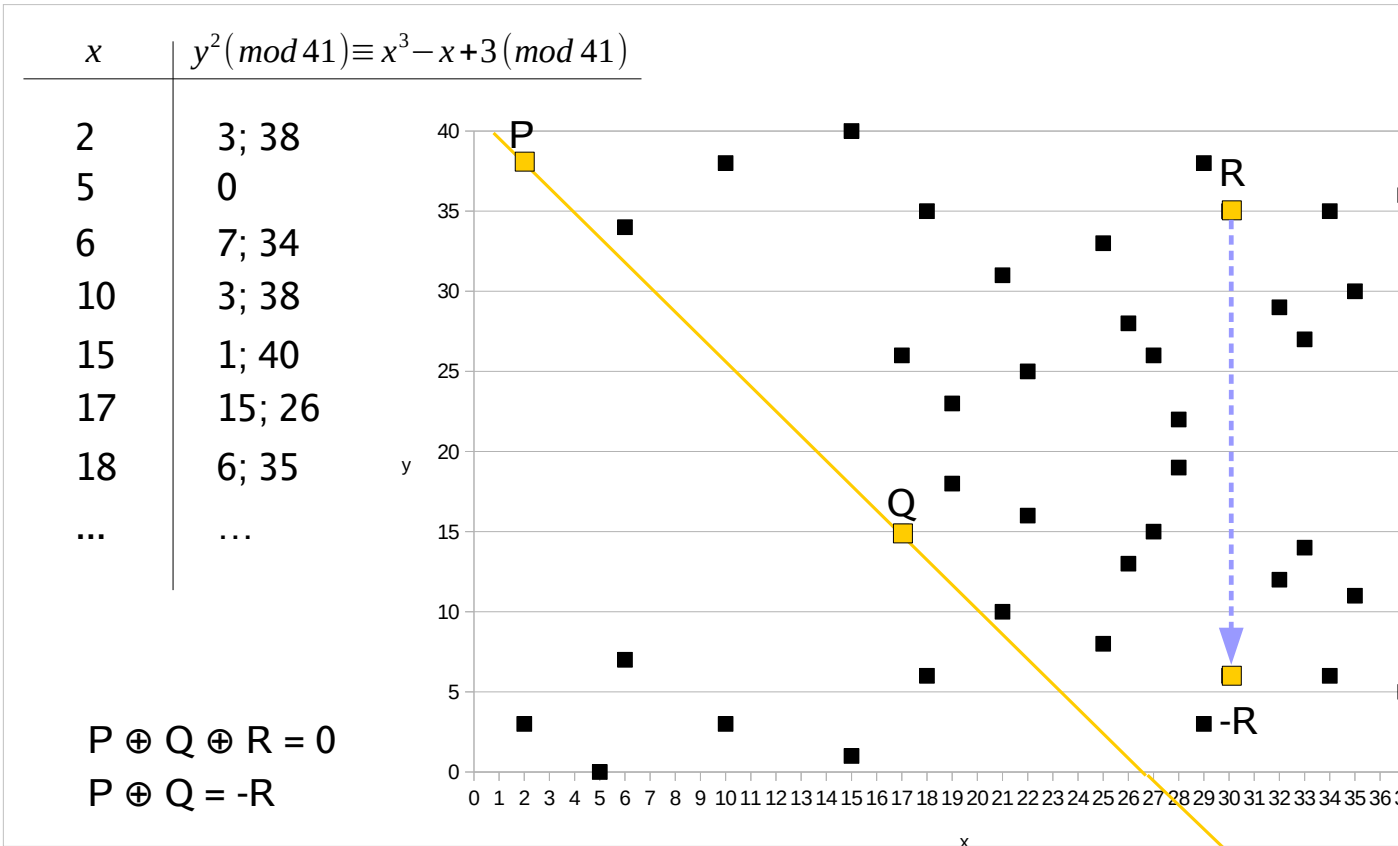
D(gellv) = Jimmy



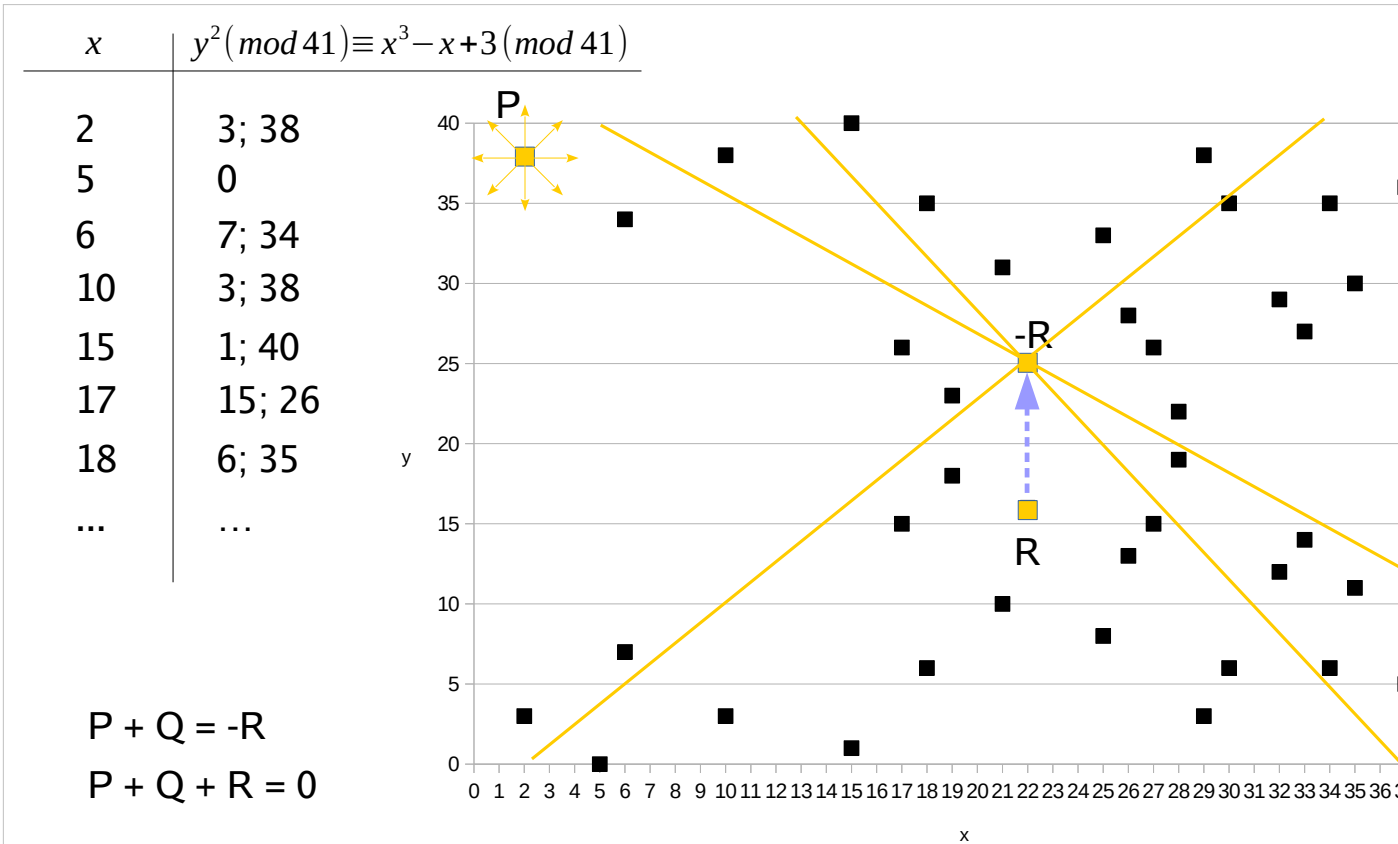
# Elliptic Curve Cryptography



In 1985, <Click> Elliptic Curve Cryptography was created which relies on the mathematical properties of an Elliptic Curve. It's smaller keys and faster mathematics make it ideal for use on small devices without a lot of computing power.

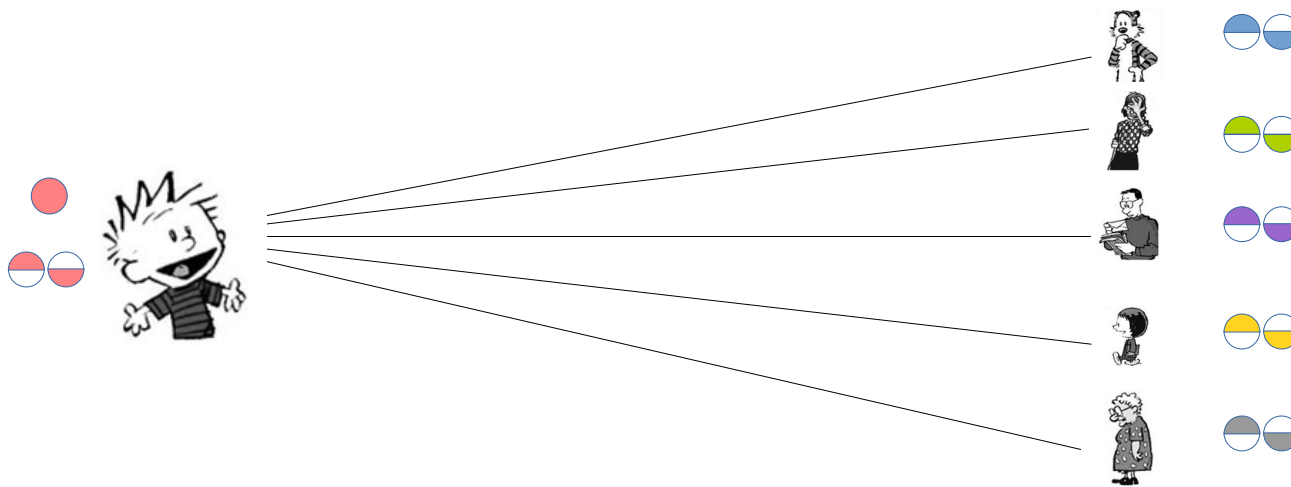


The one way function for ECC involves plotting the graph in a finite field and then doing what is called point addition. By drawing a line through points P&Q you can jump back to the top of the graph and eventually draw through point R. Then you can flip across the center axis to find the negative R value which is the answer.



Reversing this flow, however is difficult. Even if you know point  $P$  and point  $R$ , you don't know which direction to draw in order to find your point  $Q$ . This is the one way function of ECC.

# Asymmetric Key Cryptography



$$2 * n = 2 * 6 = 12 \text{ unique keys}$$

Remember, for 100 people Symmetric crypto would require about 5000 keys (4950).

For Asymmetric crypto, we only need 200.

# Quantum Computers

- Use qubits instead of classical bits
- Qubits can be in more than one state at the same time
- The state of a qubit is unknown until you observe it
- Qubits are fragile and interference can put them into an error state
- Error rates slow down the development of quantum computers

Quantum computing uses quantum mechanical properties such as superposition and entanglement to solve problems. Much like a bit in classical computers, quantum computers use qubits to represent data. Qubits have the special property of being able to exist in multiple states at the same time. Once observed the qubits reveal their final state in a probabilistic manner. Qubits are prone to errors due to many factors, which slows down the efficacy of quantum computers.

# Shor's Algorithm

- Makes factoring large semiprimes obtainable
- Another algorithm for solving the discrete logarithm problem
- And yet another for the period finding problem
- With proper quantum computer, RSA, DH, ECC, ECDH all become obsolete

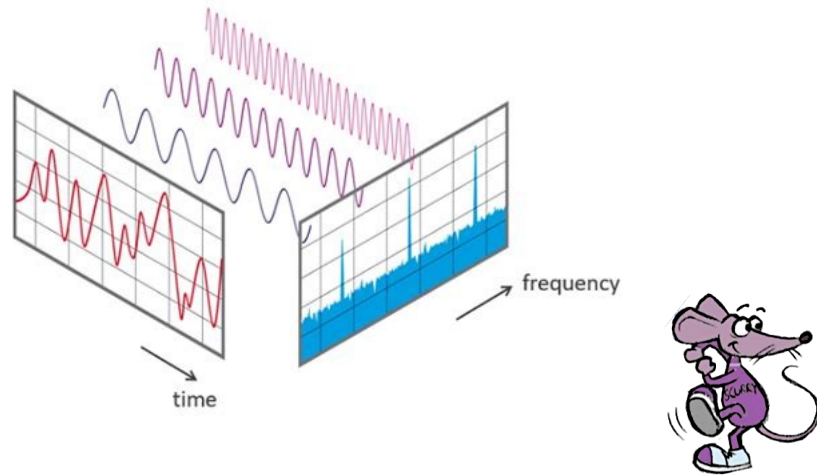
Peter Shor created an algorithm in 1994 that used a theoretical quantum computer to <Click> drastically reduce the time needed to factor semiprimes back to their original primes.

He created other algorithms to solve the <Click> discrete logarithm and <Click> period finding problem in a shorter amount of time as well.

With the correct hardware, <Click> these algorithms could render all of the previously discussed key exchange and asymmetric key cryptography system unusable.

The concepts of Shor's algorithms are complex, but here is a brief overview of how they work.

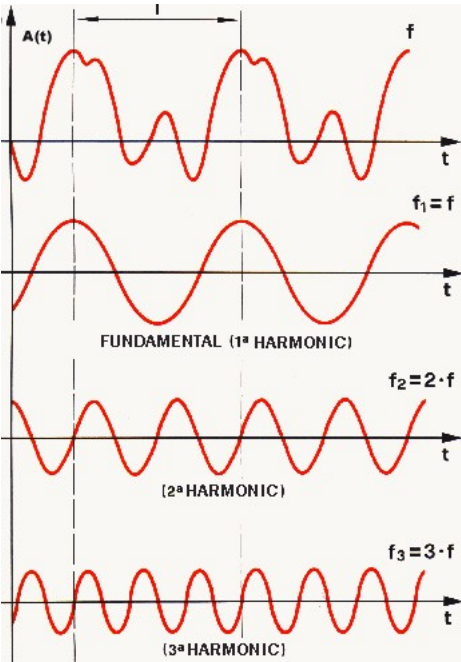
# Fourier Transform



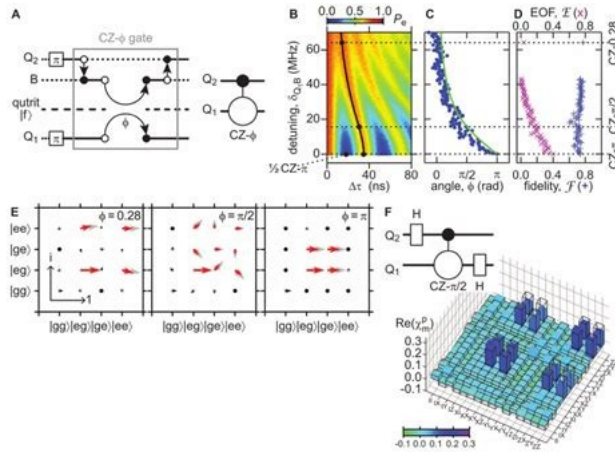
A Fourier Transform is a way to convert a complex wave into its component waves. On the left of this slide we see a complex wave over time being transformed into a frequency graph on the right.



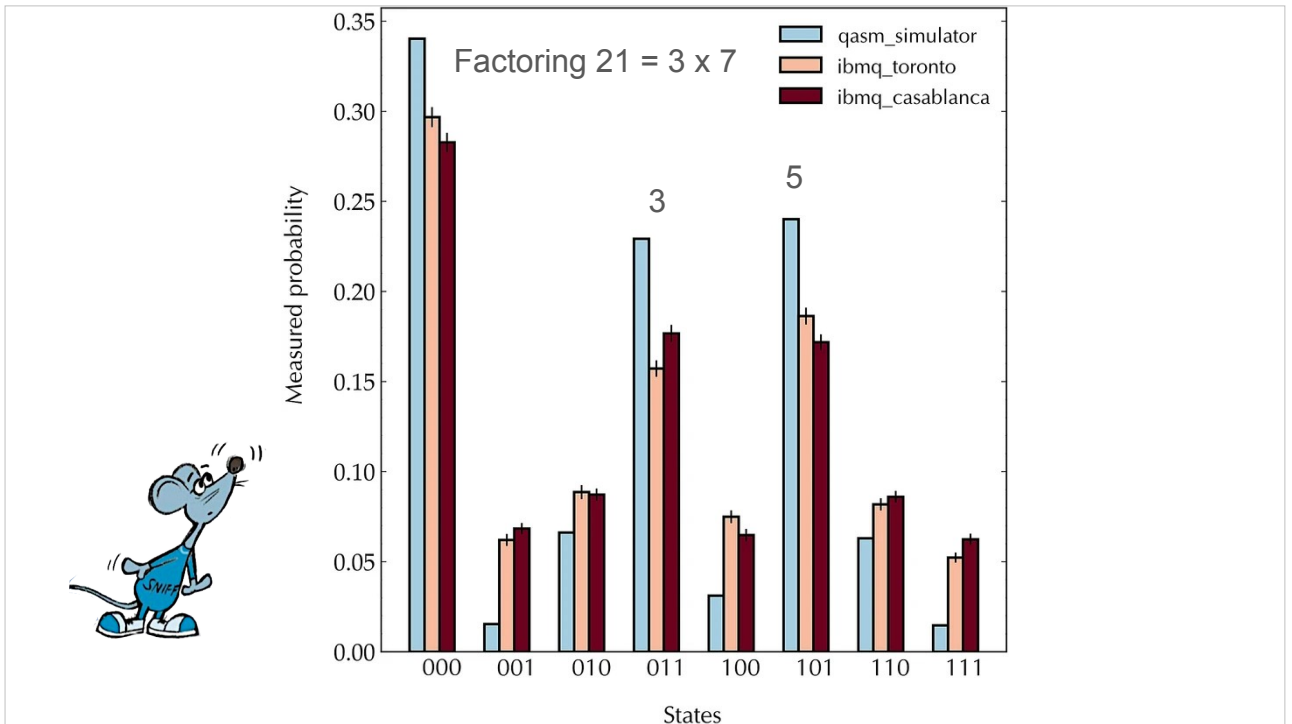
Fourier Transform



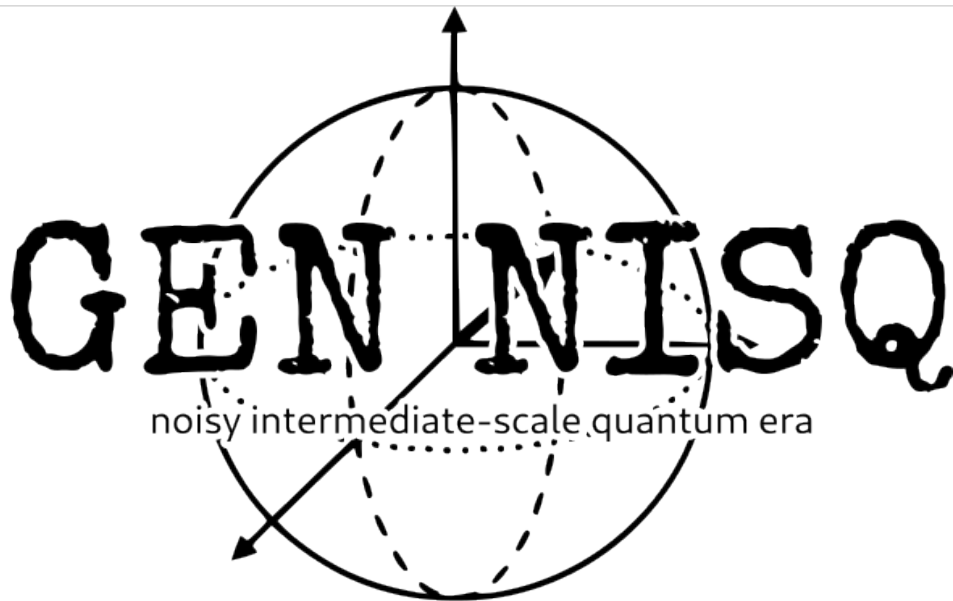
This is useful in breaking down waves of any kind, like sound waves to determine how they are formed.



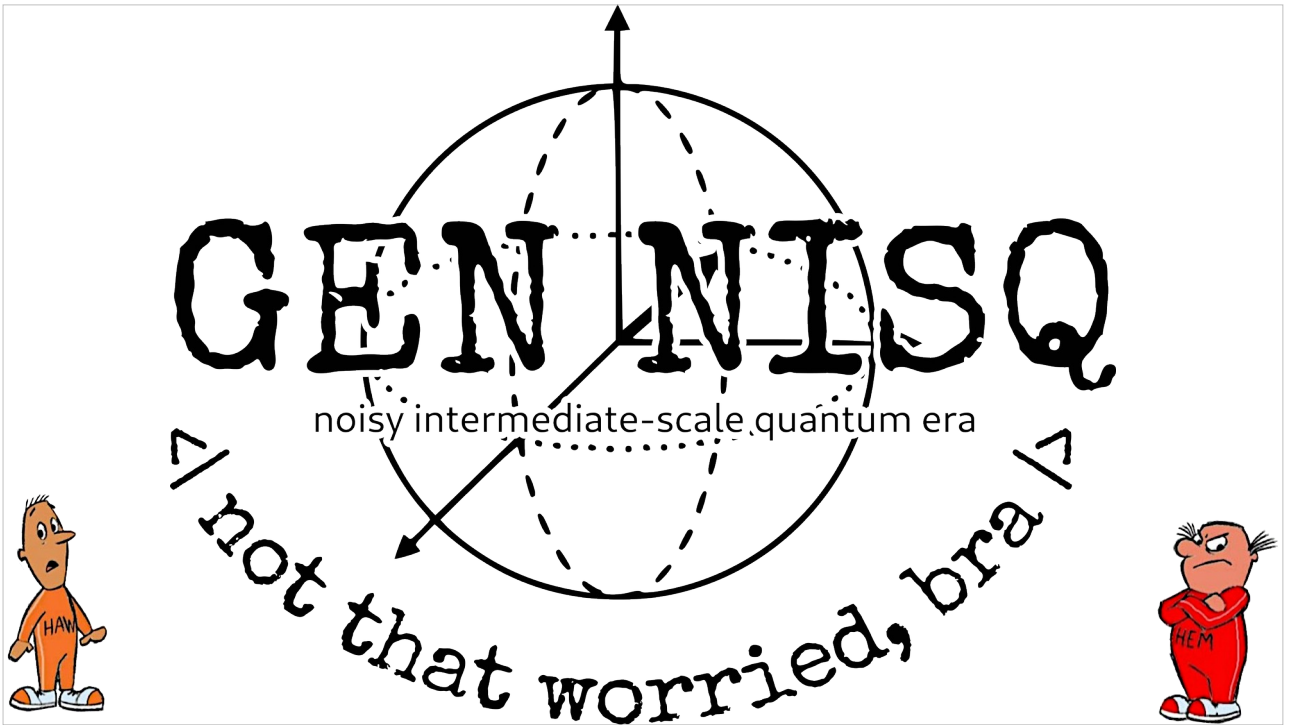
Utilizing Quantum Fourier Transform, we can sequence the probability amplitudes for all the possible outcomes upon measurement. This ability to test everything with a single measurement shows us the power of quantum computers. Notice, that unlike discrete systems, quantum computers and qubits give us a probable answer which may change over multiple runs. The number of runs is very small, though, compared to the processing that a classical computer would need.



Let's talk about the errors that happen with quantum computers. Here is the results of a 5 qubit quantum computer attempting to factor 21 into its component primes 3 and 7. As you can see, the measured probability of 0 is the highest given, with 3 and 5 coming in close second and third. The 0 result is an error state and should be ignored. Given 4 as the initial guess, the IBM casablanca test came out favoring 3 as one of the primes, which was the desired result. This is considered a successful factoring of the semiprime 21 even though there are lots of error conditions to consider.

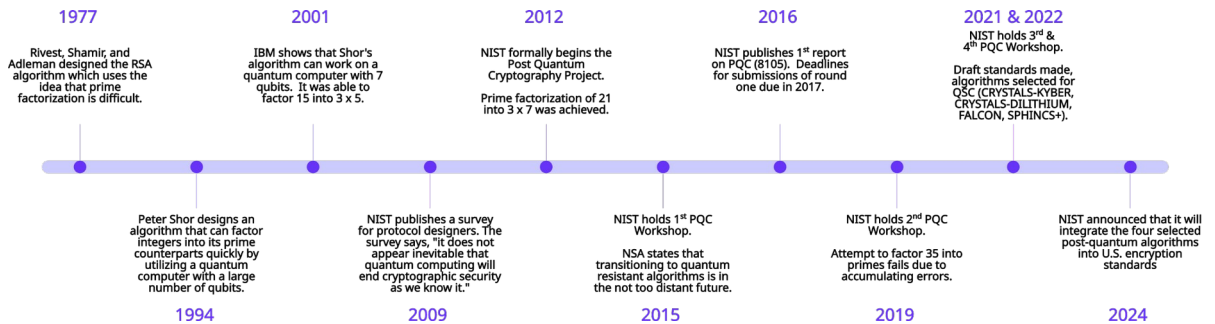


So that leads us to the question, are quantum computers ready to start breaking the Internet? It has been described that we are in the <Click> Noisy Intermediate-Scale Quantum era. This is a time of innovation and investment into building better and more stable quantum computers. But, we still have time before quantum computers will be breaking the encryption that keeps us all safe today.



So why aren't I more worried about Quantum computers? To explain, we'll let's take a look at our timeline and what quantum computers have accomplished so far.

# Timeline



As you can see from this timeline, the RSA algorithm was created nearly 47 years ago. 17 years later, Peter Shor developed the algorithm that a quantum computer could use to break RSA. Only 7 years later, IBM was able to use a 7 qubit computer to factor a 3 bit number, 15, into its prime factors 3 and 5. 11 years after that a 5 bit number, 21, was factored into 3 x 7. Another 7 years goes by and in 2019 an attempt to factor a 6 bit number, 35, failed. There is a number of larger, specially chosen numbers that have been factored by a process called quantum annealing. This has not been proven as a general solution to the problem, though.

In the meantime, in 2009, NIST starts a survey to begin developing the next generation of quantum safe encryption algorithms. Fast forward to 2023, and NIST has completed several rounds of discovery and testing of various quantum safe algorithms and chosen its candidates. In 2024, we expect those algorithms to be fully ratified and that vendors will start releasing them in their products.

## Quantum Factored

4 bit semiprime (2001): 15

5 bit semiprime (2012): 21

6 bit semiprime (2019 - failed): 35

So, to review, this is what we've accomplished. <Click><Click><Click>Ok, this last one was a failure.

## Need to factor

1024 bit semiprime:

14858031842529041742675223662020065615490358969220662  
89933239251279096441074379066840500275518447762785965  
12160601382625659982180758999544221868254722043619979  
84526745656869867322663775380667065617182042243040564  
49791211661181323805868000257522596407301217381568222  
96246476504443847811940638639921190244907229

Here you can see the size of the numbers that need to be factored for us to be truly at risk.



## Need to factor

2048 bit semiprime:

23042155144807033264822777505847352979234760665383921887127664352  
78228085219412095936558643453832808918618527312570608206987897806  
77826686707683634384826161371591122821396878931674234574664588025  
31248068491920362991080654721527620276216893955001892785769536572  
672674398163063893122113035627939070111414300285284659709274691084  
07139947940617113946753664637237729732757037649805390977042965495  
91252017823586478606798826383864169874982487262704648043916843571  
89386521326968154447830980281514623449772865907495447946085856235  
82763434406818849621131943279250579658150209579438838292900340862  
9452106790235491702341492875849

Roll of 2048bit semiprime example.

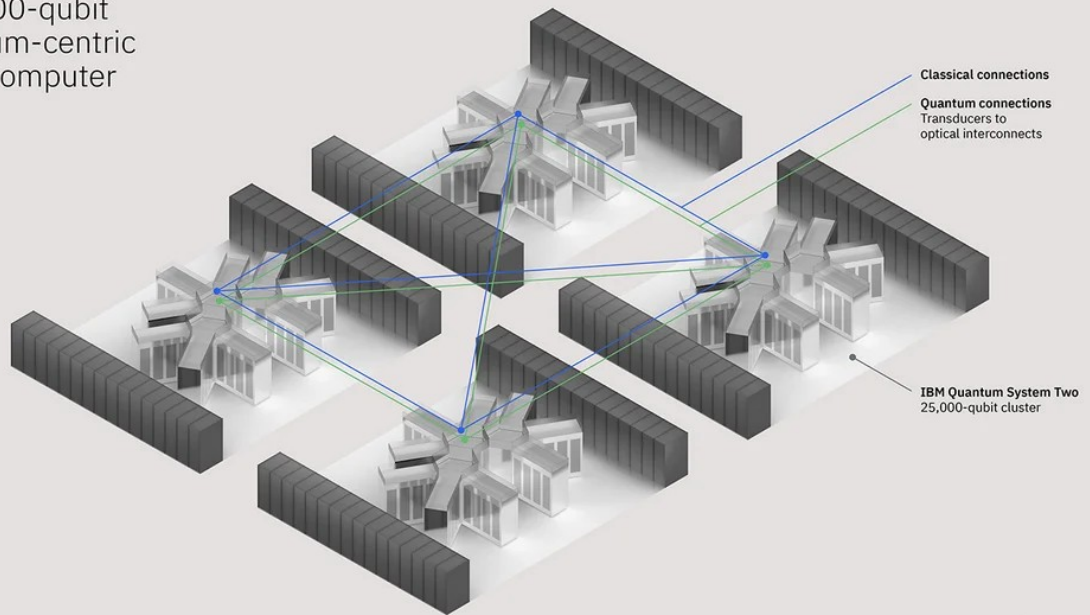
# Need to factor

4096 bit semiprime:

89504296934857829698191591033616367873245312475953314558655760495289576031777631656915867615082748  
61242821078925479671303329140277742438396447452567534005959267759634549536772718687907851138335766  
406911679216800465068765481022358269511066011043214241154792284849342476806301462838794988489104497  
97060157700850496337813043477561293777835401086614295601519110057733873752161156353028368807727894  
92709650652236081507200318618246430210732924080165186444733243222748080612117654666016347796120790  
87301378180468550758603979708720992301281518897296205081608907825465936077824761157221996661909831  
6259052074129972637775088716806070106496493062874472415479023496763767874917569948669544950361493  
26421416454157731258373198330407522928786552983597105324467054328794519859700723042119413859876220  
09399852416597534382289605254398863710160238544899455274270168012833272387402646230414545438569105  
45747681154748200516832481353745896101528766394364192199682267695484131145194250945616529523978521  
59009146621352752470666177843775024703907473664179948628184940624223676248454542922575918188677874  
33314406963341724404237652561568658515029294655137151461974536474688780183636286034894244240308027  
69700952150094244596873055650438391281230331589881854277

4096 bit semiprimes are not the largest number I've seen used for modern cryptography, either. I believe we will someday have quantum computers capable of factoring large numbers like this. So, what about those Post-Quantum Cryptography algorithms that NIST has been working on?

100,000-qubit  
quantum-centric  
supercomputer  
—  
2033



IBM Quantum

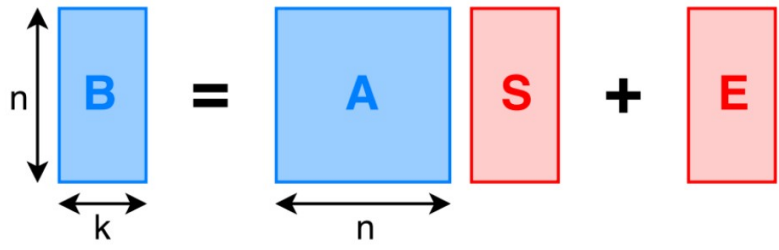
IBM plans on having a quantum-centric supercomputer in 2033 with 100,000 qubits. Is that enough to break RSA? It depends on how error prone those qubits are.

## Post-Quantum Cryptography (PQC) - where are we?

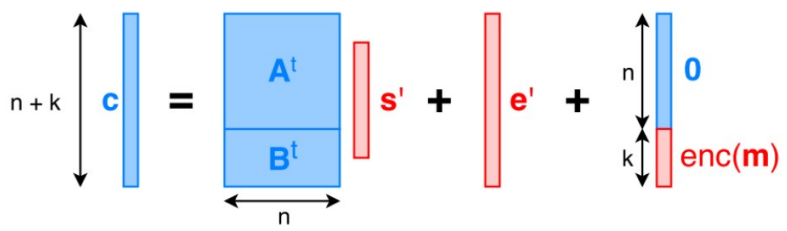
- 2022 NIST approves PQC encryption and signature algorithms
  - CRYSTALS-Kyber (general encryption)
  - CRYSTALS-Dilithium (signature)
  - FALCON (signatures for smaller applications)
  - SPHINCS+ (backup based on a different mathematical model)

<Click> NIST made its approvals of the Quantum-Safe Algorithms <Click> CRYSTALS-Kyber for encryption, <Click> CRYSTALS-Dilithium, FALCON, and SPHINCS+ for signatures.

Public parameter  $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n})$   
*KeyGen()*



*Encrypt<sub>b</sub>*( $m \in \{0, 1\}$ )

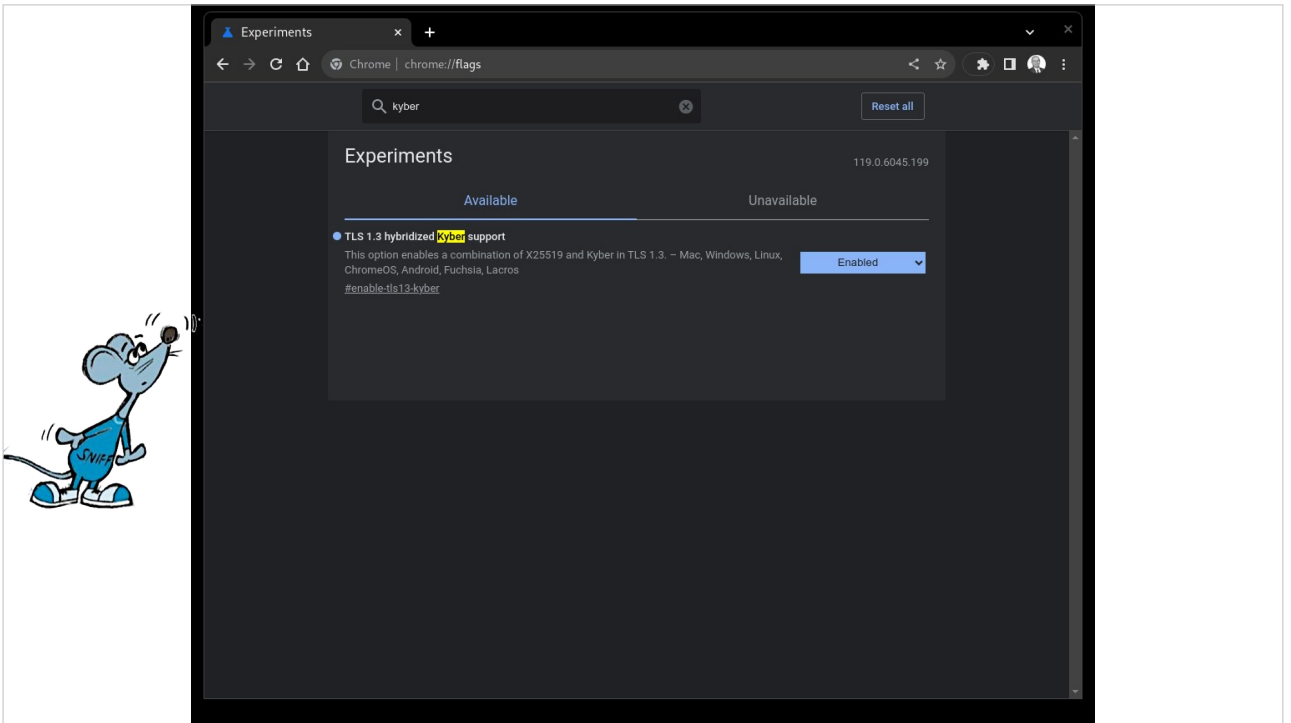


CRYSTALS-Kyber for example, uses techniques created in 2005 called Learning with Errors. Instead of being based on factoring semiprimes, discrete logarithms, or elliptic curves, it uses lattices and introduces errors to make discovery difficult mathematically, even for large scale quantum computers without interference issues.

## Post-Quantum Cryptography (PQC) - where are we?

- X25519Kyber768Draft00 readily available to test
  - Google Chrome
  - Firefox
  - BoringSSL
  - nginx
- Cloudflare Research
  - <https://pq.cloudflare.com/research/>

These algorithms have been encoded for use in various applications for testing. And Cloudflare has released a tool where you can test.



Enabling Kyber support in the latest versions of Google Chrome (Microsoft Edge, Samsung Internet, Opera, Brave, etc), for example, is easy to do.

Experiments Post-Quantum Key Agree x +

← → ↻ 🏠 🔒 pq.cloudflareresearch.com ⌵ ☆ ⚙️ 🗺️ 👤 ⋮

## Cloudflare Research: Post-Quantum Key Agreement

```

graph LR
    Visitor[Visitor] -- 1 --> Cloudflare[Cloudflare]
    Cloudflare -- 2 --> Cloudflare
    Cloudflare -- 3 --> Origin[Origin]
  
```

On essentially all domains served (1) through **Cloudflare**, including this one, **we have enabled** hybrid post-quantum key agreement. We are also **rolling out support** for post-quantum key agreement for connection from Cloudflare to origins (3).

You are using `X25519Kyber768Draft00` which is **post-quantum secure**.

### Deployed key agreements

Available with TLSv1.3 including HTTP/3 (QUIC)

Key agreement	TLS identifier
<code>X25519Kyber768Draft00</code>	<code>0x6399</code> (recommended) and <code>0xfe31</code> (obsolete)
<code>X25519Kyber512Draft00</code>	<code>0xfe30</code>
<code>X25519Kyber{xy}Draft00</code> is a <b>hybrid</b> of <code>X25519</code> and <code>Kyber{xy}Draft00</code> (in that order).	

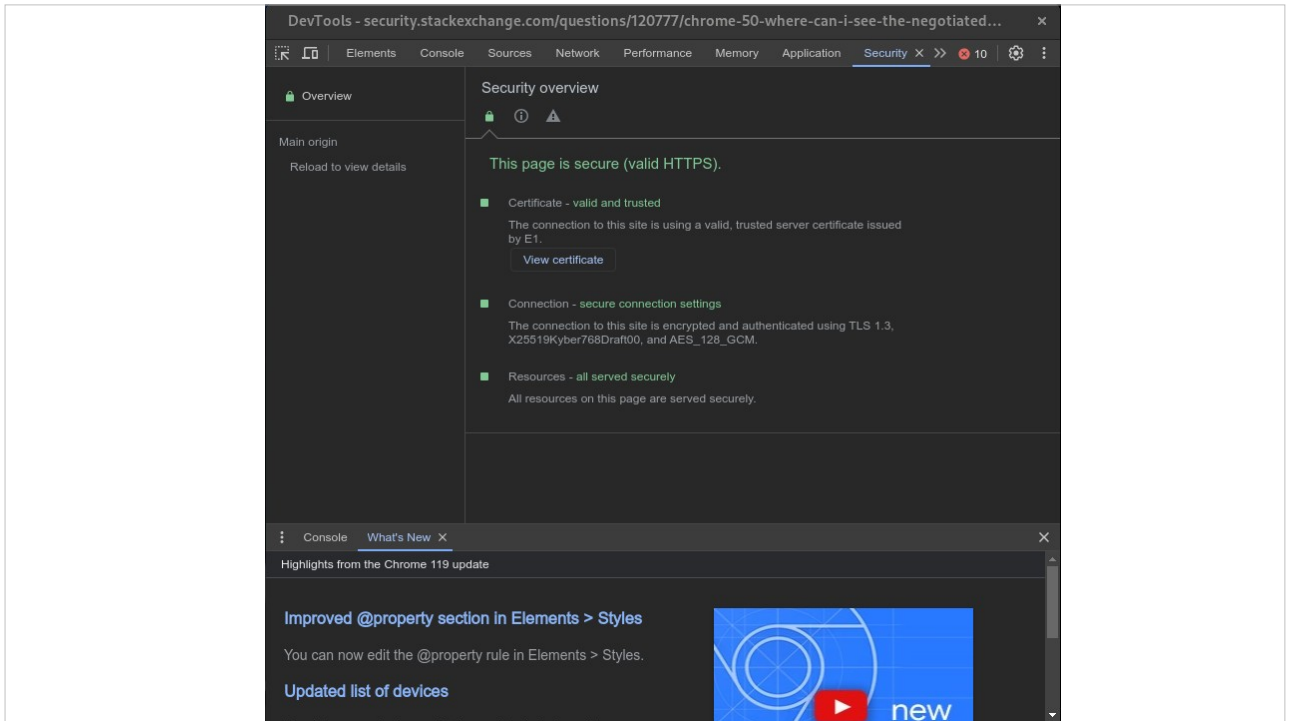
### Software support

- **Chrome 116+** if you turn on *TLS 1.3 hybridized Kyber support* (`enable-tls13-kyber`) in `chrome://flags`.



Once enabled, you can test your browser against Cloudflare's test site. Quantum-Safe Cryptography is here!





Here we have Stack Exchange which is using Kyber as well! You can access developer tools via <CTRL><SHIFT>-I in chrome and then look at the security tab to see what cipher you are using.

Sadly, this information is not available via the Chrome extension API, so we won't get a nice extension to tell us if we are quantum safe or not. You'll have to use the developer's tools.

# What do we need to do?

1. Don't panic.
2. Educate yourself and your company on the risks.
3. Understand your environment.
  - a. Where do you use encryption?
  - b. What type of encryption do you use?
  - c. Can you update? Are your vendors working on implementing PQC?
  - d. Realize that you should be auditing your encryption usage anyway.



<Click> As with all things, don't panic, and don't let your company panic. Good leadership will show that calm, collective forward movement will prevail.

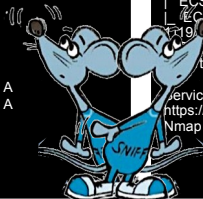
<Click> Educate yourself and your leadership on the risks you face. Make sure this is treated as a priority to plan for now. You have time, but don't become complacent.

<Click> Start auditing your environment now to understand your exposure. <Click> You may be surprised to find out where you use encryption, <Click> and the type of encryption you use. Are your certificates already expired? <Click> Are you still using certificates with SHA-1 signatures. <Click>

Nothing on this list should be new to a seasoned IT professional. Technology changes all the time and we have to constantly balance upgrading and updating against risk and productivity.

There are easy to use tools to scan your environment to see what cryptographic algorithms you are using and the health of your certificates.

```
[*]ep:~$ nmap -sV --script ssl-enum-ciphers --script ssl-cert www.example.com
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-12 20:54 EST
Nmap scan report for www.example.com (93.184.216.34)
Host is up (0.0055s latency).
Other addresses for www.example.com (not scanned):
2606:2800:220:1:248:1893:25c8:1946
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Edgecast CDN httpd (nyb/1DCD)
|_ http-server-header: ECS (nyb/1DCD)
443/tcp   open  ssl/http  Edgecast CDN httpd (nyb/1DCD)
|_ ssl-enum-ciphers:
|_ TLSv1.0:
|_ | ciphers:
|_ | compressors:
|_ |   NULL
|_ | cipher preference: server
|_ TLSv1.1:
|_ | ciphers:
|_ |   TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|_ |   TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|_ |   TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
|_ |   TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|_ |   TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (dh 2048) - A
|_ |   TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (dh 2048) - A
|_ |   TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|_ |   TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 2048) - A
|_ |   TLS_DHE_RSA_WITH_SEED_CBC_SHA (dh 2048) - A
|_ | compressors:
|_ |   NULL
|_ | cipher preference: server
|_ TLSv1.2:
|_ | ciphers:
|_ |   TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|_ |   TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|_ |   TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (dh 2048) - A
|_ |   TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|_ | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
|_ | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|_ | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
|_ | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|_ | compressors:
|_ |   NULL
|_ | cipher preference: server
|_ TLSv1.3:
|_ | ciphers:
|_ |   TLS_AKE_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|_ |   TLS_AKE_WITH_CHACHA20_POLY1305_SHA256 (secp256r1) - A
|_ |   TLS_AKE_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|_ | cipher preference: server
|_ | least strength: A
|_ | ssl-cert: Subject: commonName=www.example.org/organizationName=InternetxC2I
|_ | xA0CorporationxC2xA0forxC2xA0AssignedxC2xA0NamesxC2xA0andxC2
|_ | xA0Numbers/stateOrProvinceName=California/countryName=US
|_ | Subject Alternative Name: DNS:www.example.org, DNS:example.net, DNS:example.edu,
|_ | DNS:example.com, DNS:example.org, DNS:www.example.com, DNS:www.example.edu,
|_ | DNS:www.example.net
|_ | Issuer: commonName=DigiCert TLS RSA SHA256 2020
|_ | CA1/organizationName=DigiCert Inc/countryName=US
|_ | Public Key type: rsa
|_ | Public Key bits: 2048
|_ | Signature Algorithm: sha256WithRSAEncryption
|_ | Not valid before: 2023-01-13T00:00:00
|_ | Not valid after: 2024-02-13T23:59:59
|_ | MD5: 749b8bbcb4a6cb23c205c9850b35bed6a
|_ | SHA-1: f2aad73d32683b716d2a7d61b51c6d5764ab3899
|_ | http-server-header:
|_ | | ECS (nyb/1D2E)
|_ | | ECS (nyb/1DCD)
|_ | |_ tcp closed bnetgame
|_ | |_ tcp open h323q931?
|_ | |_ tcp closed rtmp
|_ | service detection performed. Please report any incorrect results at
|_ | https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 189.80 seconds
```



Here is an example of a simple scan with nmap of a host. You can easily expand this to your entire environment and the script automatically scans around 1000 known ports that serve SSL and TLS. It supports the majority of protocols, too.

# What do we need to do?

## 1. Plan

- a. What can you upgrade?
- b. What can't you upgrade?
- c. Risk Analysis

## 2. Execute your Plan

## 3. Automation

- a. ACME - Automatic Certificate Management Environment
- b. SCEP - Simple Certificate Enrollment Protocol
- c. REST/API - Check with your certificate provider, InCommon supports this

Based on your encryption audit, <Click>make a plan. Find out <Click> what can be upgraded, and <Click> what can't. Do a <Click> risk analysis of the things that are too old to be updated, or just can't. Do you have a replacement plan for those items? <Click> then execute your plan when the technology becomes available.

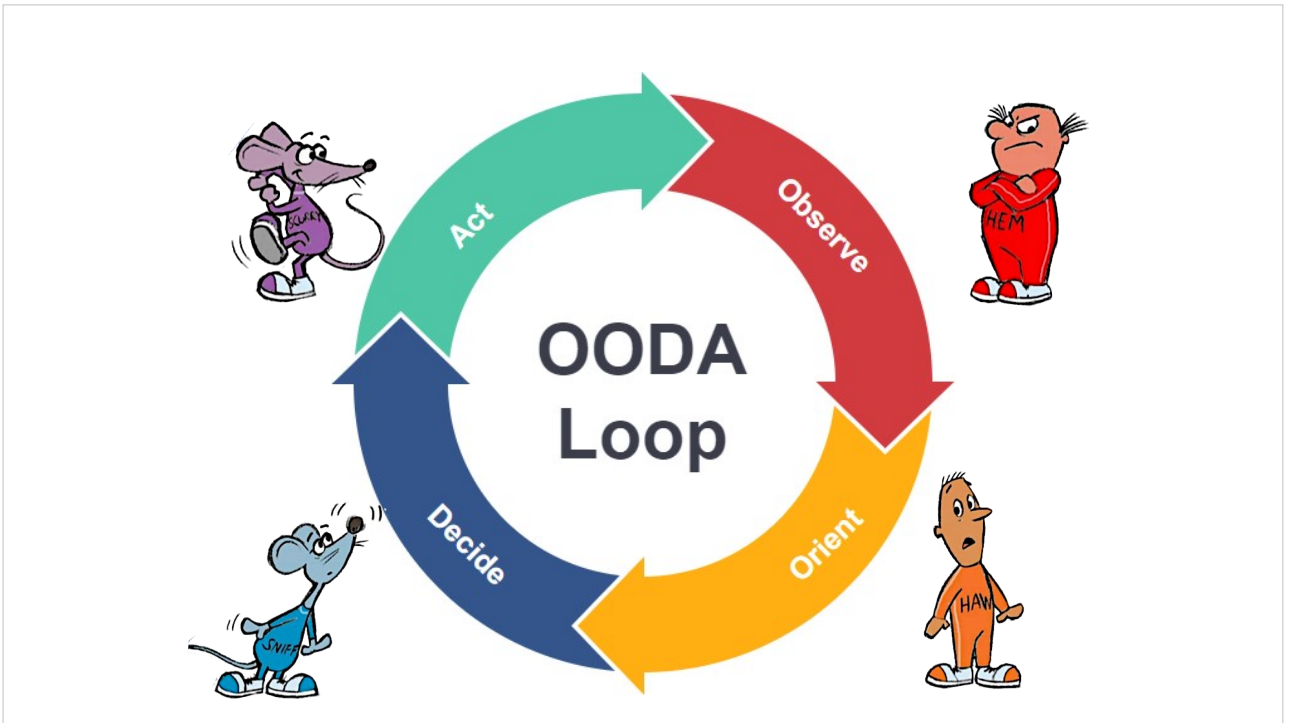
<Click> Also, remember that automation to certificate deployment is important to consider. Chromium, the backend engine to most modern browsers (with the notable exception of Firefox) is looking to push for 90 day certificates. Being able to deploy these in a large environment will really need to rely on automation. Setting this up now will mean it is all that much easier to deploy PQC when it is made available.

## Module-Lattice-based Key-Encapsulation Mechanism FIPS-203

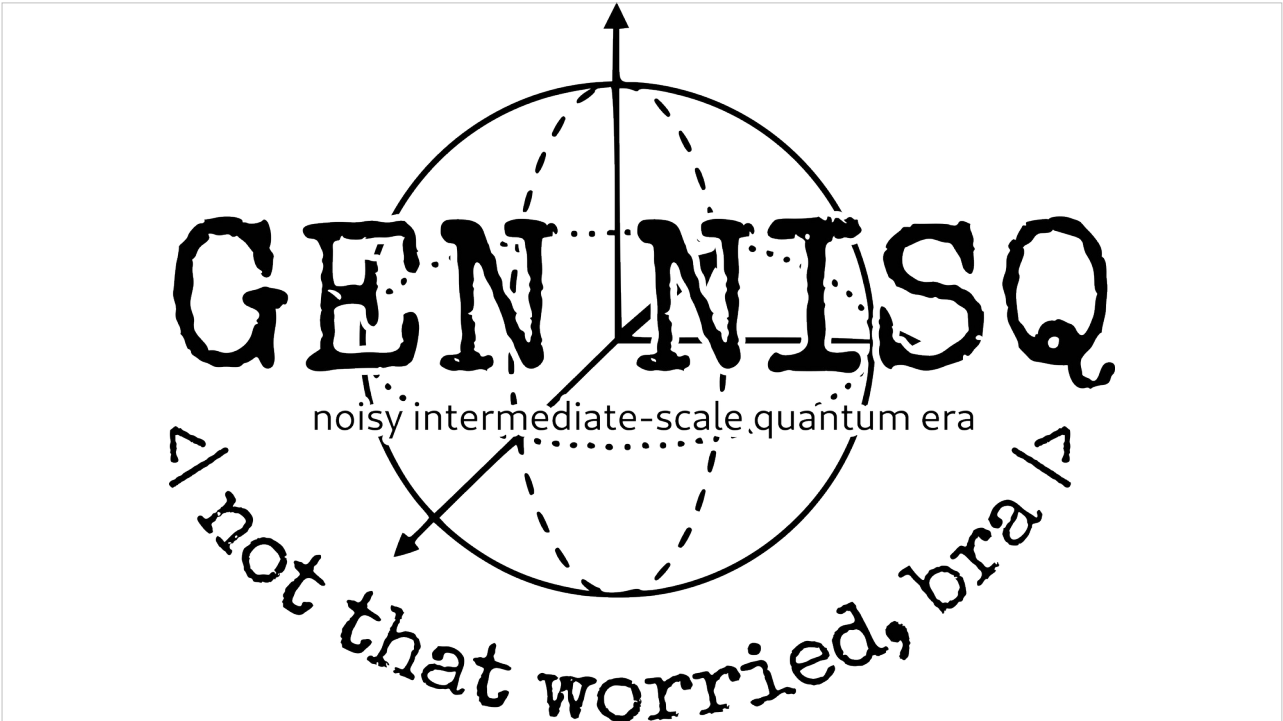
13. Qualifications. In applications, the security guarantees of a KEM only hold under certain conditions (see NIST SP 800-227 [1]). One such condition is the secrecy of several values, including the randomness used by the two parties, the decapsulation key, and the shared secret key itself. Users shall, therefore, guard against the disclosure of these values. While it is the intent of this standard to specify general requirements for implementing ML-KEM algorithms, **conformance to this standard does not ensure that a particular implementation is secure**. It is the responsibility of the implementer to ensure that any module that implements a key establishment capability is designed and built in a secure manner. Similarly, the use of a product containing an implementation that conforms to this standard does not guarantee the security of the overall system in which the product is used. The responsible authority in each agency or department shall ensure that an overall implementation provides an acceptable level of security. NIST will continue to follow developments in the analysis of the ML-KEM algorithm. As with its other cryptographic algorithm standards, **NIST will formally reevaluate this standard every 5 years**. Both this standard and possible threats that reduce the security provided through the use of this standard will undergo review by NIST as appropriate, taking into account newly available analysis and technology. In addition, the awareness of **any breakthrough in technology or any mathematical weakness of the algorithm will cause NIST to reevaluate** this standard and provide necessary revisions.



And remember, we are all in this boat together. NIST describes Module-Lattice-based Key-Encapsulation Mechanism in FIPS-203. Note that NIST already plans to re-evaluate the protocol every 5 years or if needed sooner. Once again, repeating the cyclical nature of information security



The OODA loop (observe, orient, decide, act) was developed by military strategist and United States Air Force Colonel John Boyd. Remaining agile and following the steps of Observe <Click>, Orient <Click>, Decide <Click>, and Act <Click> will keep us always moving in the right direction.



Thank you!