Some applications of mathematics in cryptography

Steven Galbraith

Mathematics Department, University of Auckland, New Zealand

Hanoi

Thank you to the organisers for the invitation.

Laws of Nature and Science

- Kepler's Laws of Planetary Motion
- Newton's Laws of Motion
- Darwin's law of natural selection
- Kirchhoff's laws
- Moore's Law: The number of transistors on a microchip doubles about every two years

I thought it would be nice to have a law.

So today, for the first time, I am unveiling my law.

Galbraith's law

1. Every 10 years someone invents some cryptography that is (at least) 10 times slower than the previous ideas, but has some new features.
2. After another 10 years it is fast enough to be practical, and has a totally unexpected application.
3. Both 1 and 2 are driven by Mathematics.

Plan of talk

- Invention of public key cryptography
- Elliptic curve cryptography
- Pairing-based cryptography
- Fully homomorphic encryption
- What's next?

Encryption

- Encryption allows two parties to communicate securely over an insecure channel.
- Traditional methods use a shared secret key between sender and receiver.
- Called symmetric key cryptography.
- Cipher machines were fast and reliable and widely used.
- Phan Duong Hieu and Neal Koblitz, "Cryptography During the French and American Wars in Vietnam", Cryptologia 2017.

Public key cryptography

- The sender only uses public information. The receiver has a secret key.
- Discovered in the 1970s by James Ellis, Clifford Cocks, Malcolm Williamson (GCHQ, UK), Whit Diffie, Martin Hellman, Ralph Merkle, Ron Rivest, Adi Shamir, Len Adleman.
- Uses fairly elementary number theory.
- For example, the RSA cryptosystem works in the multiplicative group of integers modulo N, where $N = pq$ is a product of two large primes.

Public key cryptography in the 1970s

Computers were very slow then. I mean if you think about a VAX back in the day, trying to find a large prime number, it really took quite a long time. Many minutes, maybe half an hour in some cases, depending on how big the prime was. And so the practical aspects of this were daunting at the time.

Secure email, secure phones were the kind of things that... struck us as being the most likely areas of first application.

– Ron Rivest[1]

[1] https://amturing.acm.org/pdf/RivestTuringTranscript.pdf

The Wisdom of Ron Rivest

But then it was an issue, and so we've seen the evolution of the computing power...have its impact on public-key cryptography.
And I think this will continue, we'll have...these schemes now that seem complicated, will become easier to implement as we get more tightly integrated computer systems...there's still lots of gains that can be made in implementing crypto in special purpose ways, for some of these newer ideas that have come on the scene.

Public key cryptography

- Combination of increases in computer power, and also mathematical techniques for improved implementation, mean RSA has been practical and widely used for decades.
- Advances in cryptanalysis (eg factoring algorithms) were also dramatic, and so the integers are much larger than originally envisaged.
- The main "killer" applications of RSA are not the ones foreseen in the 1970s. They are the internet (eg TLS protocol), code-signing (for automatic software updates), etc.


MÌ
QUẢNG
BÚN
MẮM
BÁNH
CANH GIÒ
HEO
HỦ
TIU TÔM
THỊT

Elliptic Curve Cryptography

- Miller and Koblitz proposed elliptic curve crypto in the mid-1980s.
- The idea was that it would resist subexponential index calculus algorithms, and so be more efficient.
- At the time RSA was ok. But subsequent progress in integer factoring (number field sieve) made the advantages of ECC more relevant.
- Elliptic curves were viewed as much more advanced mathematics.

Elliptic Curve Crypto

The security of cryptosystems based on elliptic curves is not well understood, due in large part to the abstruse nature of elliptic curves. Few cryptographers understand elliptic curves... Over time, this may change, but for now trying to get an evaluation of the security of an elliptic-curve cryptosystem is a bit like trying to get an evaluation of some recently discovered Chaldean poetry.

– Ron Rivest (mid-1990s)

Elliptic Curve Crypto

- Through the 1990s and 2000s elliptic curves became much more efficient, due to new discoveries such as Montgomery curves and Edwards curves etc.
- Un-anticipated advantages of elliptic curves include short public keys (important for public key certificates and blockchain applications).
- Elliptic curves are now a very mature field and widely used.

Pairing-based cryptography

- Proposed around 2000 by Sakai-Ohgishi-Kasahara, Joux, Boneh-Franklin.
- Much slower than traditional elliptic curve cryptography, but promised new applications such as identity-based cryptography and 3-party key exchange.
- More advanced mathematics (eg greater use of algebraic geometry).

What is a pairing?

- Let G_1, G_2, G_3 be cyclic groups of prime order r.
- A pairing is a function $e : G_1 \times G_2 \rightarrow G_3$ such that

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$

- (Also, not the constant function!)
- Example: The Weil pairing on elliptic curves.
- (Elliptic curves usually written in additive notation, I'm writing everything in multiplicative notation for simplicity.)

BLS signature scheme

- A public key signature scheme using pairings invented by Boneh, Lynn, Shacham.
- Uses the property $e(g_1^x, g_2) = e(g_1, g_2^x)$.
- Public key is $(g_1, h_1 = g_1^x)$ and the private key is x.
- Given a message m, hash the message to $H(m) \in G_2$.
- Signature on message m is $s = H(m)^x$. To verify signature check that

$$e(h_1, H(m)) = e(g_1, s).$$

- Aggregation: If s_1 and s_2 are signatures on messages m_1, m_2 then

$$e(h_1, H(m_1)H(m_2)) = e(g_1, s_1 s_2).$$

The aggregated signature $s_1 s_2$ is the same size as a single signature.

Pairing-based cryptography

- Lots of research on making pairings faster.
- Originally thought to be useful for identity-based crypto etc, but actually the main application is zero knowledge proofs, e.g., following Groth-Ostrovsky-Sahai (EUROCRYPT 2006, CRYPTO 2006), Groth (ASIACRYPT 2006, ASIACRYPT 2010), Groth-Sahai (EUROCRYPT 2008) etc.
- Hot areas are zk-SNARKs and zk-STARKs, used in blockchain/crypto-currency applications such as ZCash.
- Key idea is non-interactive zero-knowledge arguments for quadratic arithmetic programs: essentially can check quadratic polynomial relations (like $c = ab$) among secret values a, b, c by checking

$$e(g_1^a, g_2^b) = e(g_1, g_2^c).$$

Pairing-based cryptography and Galbraith's law

- Fits my "law" that every 10 years someone invents some cryptography that is (at least) 10 times slower than the previous ideas, but has some new features.
- Based on new mathematics.
- Originally considered rather slow, but nowadays considered very practical (not just due to Moore's law).
- Originally in 2000 thought to be useful for identity-based crypto, but actually used for blockchain (which wasn't a "thing" in 2000).

Fully Homomorphic Encryption (FHE)

- Craig Gentry's PhD thesis in 2008 was a major breakthrough, and solved the major open problem of homomorphic encryption.
- Fully Homomorphic Encryption = Computing on encrypted data.
- Key insight was bootstrapping: Running the decryption algorithm homomorphically to "refresh" ciphertexts.
- Based on algebraic number theory (lattices).
- Ridiculously slow.

Fully Homomorphic Encryption (FHE)

Last month, IBM made some pretty brash claims about homomorphic encryption and the future of security. I hate to be the one to throw cold water on the whole thing — as cool as the new discovery is — but it's important to separate the theoretical from the practical.
Unfortunately... Gentry's scheme is completely impractical. It uses something called an ideal lattice as the basis for the encryption scheme, and both the size of the ciphertext and the complexity of the encryption and decryption operations grow enormously with the number of operations...and converting a computer program, even a simple one, into a Boolean circuit requires an enormous number of operations.

– Bruce Schneier, 9 July 2009.

Fully Homomorphic Encryption (FHE)

These aren't impracticalities that can be solved with some clever optimization techniques and a few turns of Moore's Law; this is an inherent limitation in the algorithm. In one article, Gentry estimates that performing a Google search with encrypted keywords - a perfectly reasonable simple application of this algorithm - would increase the amount of computing time by about a trillion. Moore's law calculates that it would be 40 years before that homomorphic search would be as efficient as a search today, and I think he's being optimistic with even this most simple of examples.

Fully Homomorphic Encryption (FHE)

- A sequence of works over the following 10 years resulted in potentially practical systems.
- Several libraries for FHE, including the HEAAN (Homomorphic Encryption for Arithmetic of Approximate Numbers) open source homomorphic encryption (HE) library, based on:
  J.H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic encryption for arithmetic of approximate numbers, ASIACRYPT 2017.

Fully Homomorphic Encryption (FHE)

Current "killer application" is privacy-preserving machine learning. See:

- Eurocrypt 2021 invited talk by Craig Gentry "A Decade (or so) of fully homomorphic encryption" (youtube)
- Kristin Lauter (Microsoft) "Private AI: Machine Learning on Encrypted Data" (youtube)
- Nigel Smart, CIF Seminar "Homomorphic Encryption" (youtube)

FHE and Galbraith's law

- Fits my "law" that every 10 years someone invents some cryptography that is (at least) 10 times slower than the previous ideas, but has some new features.
- This time much worse than 10 times slower!
- Based on lattices, which had been used in crypto, but many other new mathematical tricks were introduced like modulus switching, SIMD packing using Chinese Remainder Theorem, etc.
- Originally considered impossibly slow, but nowadays considered potentially practical (not due to Moore's law).
- Current applications are not exactly what was originally proposed.


ASIACRYPT 2016
4-8 December, 2016, Hanoi, Vietnam
Microsoft
ECPay
BIDV
intel
CISCO
Google

What's next?

What's the current thing that is not practical and easy to criticise?

Two candidates:

- Post-quantum cryptography
- Indistinguishability Obfuscation (iO)

Quantum computing

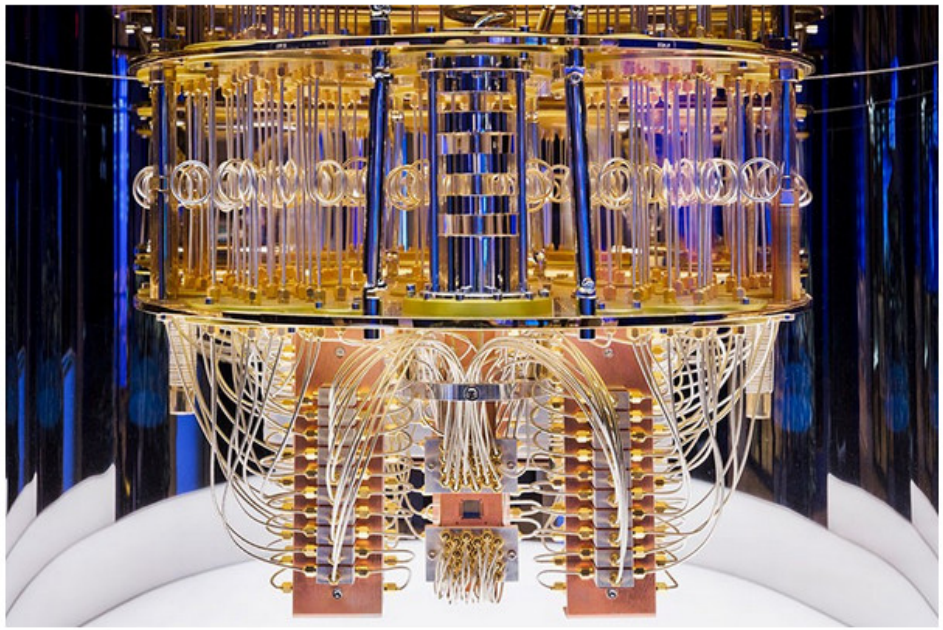
- Quantum computing was proposed by: Paul Benioff (1980), Yuri Manin (1980), Richard Feynman (1982) and David Deutsch (1985).
- Peter Shor (1994): polynomial-time quantum algorithm for integer factorisation and discrete logs.
- Advances in quantum computers pose a serious threat to the security of current public key systems.

NEWS | 19 November 2021

First quantum computer to pack 100 qubits enters crowded race

But IBM's latest quantum chip and its competitors face a long path towards making the machines useful.

Philip Ball



The innards of an IBM quantum computer show the tangle of cables used to control and read out its qubits. Credit: IBM

IBM's newest quantum-computing chip, revealed on 15 November, established a milestone of sorts: it packs in 127 quantum bits (qubits), making it the first such device to reach 3 digits. But the achievement is only one step in an aggressive agenda boosted by billions of

Post-quantum cryptography

- Recent years have shown steady progress in quantum computing.
- Post-quantum crypto (PQC) is systems that run on current hardware and devices, but are secure against an attacker with a quantum computer.
- There are currently several international standardisation processes underway, highest profile being NIST.
- There are many efficient post-quantum cryptosystems, but it is a challenge to fit them into current standards such as TLS. See "Sizing Up Post-Quantum Signatures"[2] Cloudflare blog post.

[2] https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

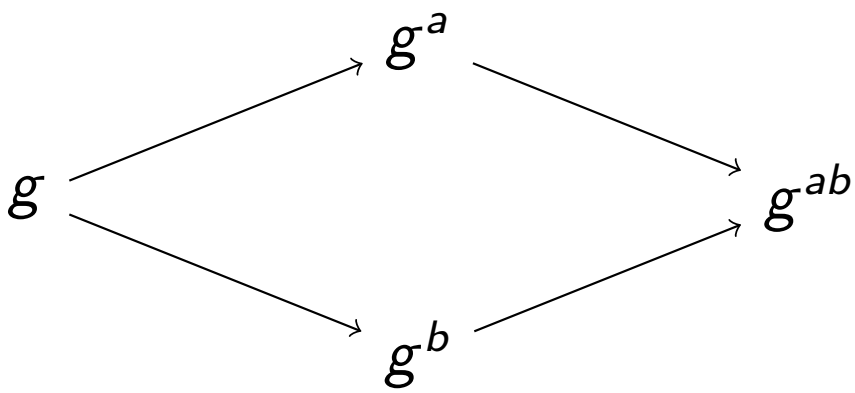
Features of post-quantum cryptography (PQC)

- Based on new mathematics.
- Larger keys/ciphertexts/signatures.
- Different software libraries needed.
- New implementation challenges.
- Less suitable for constrained environments such as IoT, embedded systems, etc.

Why doesn't Shor's algorithm break them?

- **Less algebraic structure**: multivariate polynomial systems, hash trees, non-abelian groups, isogenies.
- **Use of "noise" to disrupt exact algebraic equations**: lattice-based, coding-based.

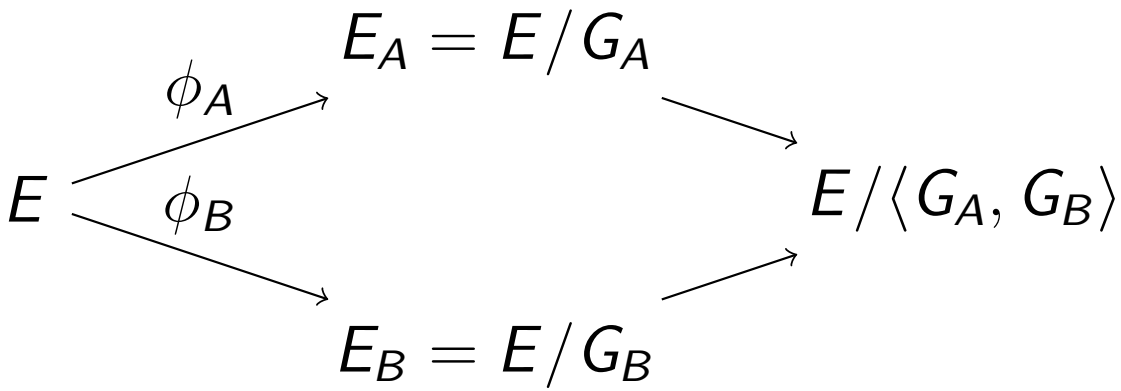
Diffie-Hellman key exchange



Discrete logarithm problem: Given g and g^a, hard to compute a.
Diffie-Hellman problem: Given g, g^a, and g^b, hard to compute g^{ab}.

Generalised Diffie-Hellman: Homomorphisms

Let E be an elliptic curve and G_A, G_B finite subgroups such that $G_A \cap G_B = \{0\}$.



Alice and Bob need to give each other enough information that they can compute $\phi_A(G_B)$ and $\phi_B(G_A)$.
Alice computes $E_B/\phi_B(G_A)$ and Bob computes $E_A/\phi_A(G_B)$.

Isogenies

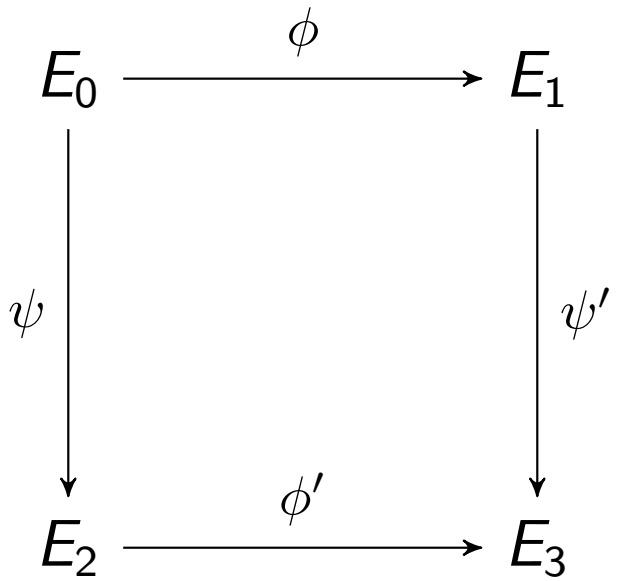
- An **isogeny** $\phi : E_1 \to E_2$ of elliptic curves is a (non-constant) morphism and a group homomorphism.
- An isogeny has finite kernel.
- Given a finite subgroup $G \subseteq E_1(\overline{\mathbb{F}}_q)$ there is a (unique separable) isogeny $\phi_G : E_1 \to E_2$ with kernel G.
  Can compute ϕ_G using Vélu.
- We will write $E_2 = E_1/G$.
- We focus on separable isogenies, in which case $\deg(\phi) = \#\ker(\phi)$.

Main Computational Problem Regarding Isogenies

Given E_1, E_2 elliptic curves over $\mathbb{F}_q$ such that there is an isogeny between them, find an isogeny $\phi : E_1 \to E_2$.

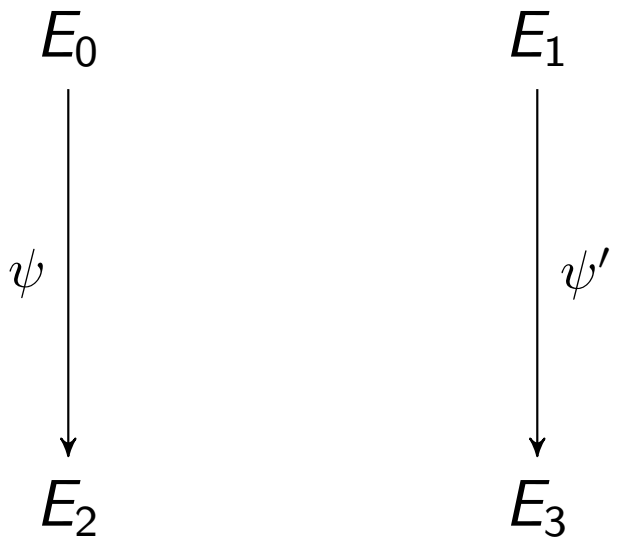
Isogeny-based identification scheme

Public key E_0 and E_1.
Private key $\phi : E_0 \to E_1$.



Isogeny-based identification scheme

Public key E_0 and E_1.
Commit to E_2 and E_3. Receive challenge.
If challenge $= 0$ then



Isogeny-based identification scheme

Public key E_0 and E_1.
Commit to E_2 and E_3. Receive challenge.
If challenge $= 1$ then

$$E_0 \qquad\qquad E_1$$

$$E_2 \xrightarrow{\phi'} E_3$$

and repeat protocol many times...

Soundness of isogeny-based identification scheme

- This works as an identification/signature scheme, but not as a proof of knowledge of an isogeny of specific degree from E_0 to E_1.
- There was an error in soundness proofs of this protocol, going back to Jao, De Feo and Plût in 2014.
- Fixed in this recent paper:
  Luca De Feo, Samuel Dobson, Steven D. Galbraith and Lukas Zobernig, "SIDH Proof of Knowledge", eprint 2021/1023.
- Idea is that need to carry some more information and verifier needs to check more conditions.
- Security proof needs a new assumption.
- Challenge: Find more efficient signature schemes.
  (See SQI-Sign, which is cool.)

Isogeny-based crypto

- Advantages over lattice crypto: very short ciphertexts. CSIDH is even better.
- Disadvantages: relatively slow. Seems difficult to design sophisticated protocols due to lack of algebraic structure.
- What is the future of isogeny crypto? Will it obey my "law" and become more practical and have unforseen applications?

Indistinguishability Obfuscation (iO)

- Barak, Goldreich, Sahai, Impagliazzo, Vadhan, Rudich and Yang. On the (Im)possibility of Obfuscating Programs. CRYPTO 2001.
- An **obfuscator** is like a compiler that turns a circuit (program) C into another program $O(C)$.
- Definition of iO: Given any two equivalent circuits C_0 and C_1 of similar size, the obfuscations $O(C_0)$ and $O(C_1)$ are computationally indistinguishable.
- **Proposition:** (Inefficient) indistinguishability obfuscators exist. Proof: Let $O(C)$ be the lexicographically first circuit of size $|C|$ that computes the same function as C.
- iO looks silly, but is surprisingly useful.

Indistinguishability Obfuscation (iO)

- First iO scheme proposed by Garg, Gentry, Halevi, Raykova and Sahai, Candidate indistinguishability obfuscation and functional encryption for all circuits, FOCS 2013.
- Many attacks and variants.
- Latest work: Jain, Lin and Sahai, Indistinguishability obfuscation from well-founded assumptions, STOC 2021.
- Uses new assumptions such as subexponential hardness of Learning Parity with Noise (LPN), and the existence of a Pseudo-Random Generator (PRG) in NC^0 with stretch $n^{1+\tau}$.

The hype

COMPUTER SECURITY

Computer Scientists Achieve 'Crown Jewel' of Cryptography

By ERICA KLARREICH

November 10, 2020

A cryptographic master tool called indistinguishability obfuscation has for years seemed too good to be true. Three researchers have figured out that it can work.

18 |

—

Kiel Mutschelknaus for Quanta Magazine

In 2018, Aayush Jain, a graduate student at the University of California, Los Angeles, traveled to Japan to give a talk about a powerful cryptographic tool he and his colleagues were developing. As he detailed the team's approach to indistinguishability obfuscation (iO for short), one audience member raised his hand in bewilderment.

"But I thought iO doesn't exist?" he said.

At the time, such skepticism was widespread. Indistinguishability obfuscation, if it could be built, would be able to hide not just collections of data but the inner workings of a computer program itself, creating a sort of cryptographic master tool from which nearly every other cryptographic protocol could be built. It is "one cryptographic primitive to rule them all," said Boaz Barak of Harvard University. But to many computer scientists, this very power made iO seem too good to be true.

Indistinguishability Obfuscation (iO)

Quanta magazine recently published a breathless article on indistinguishability obfuscation - calling it the "crown jewel of cryptography" - and saying that it had finally been achieved, based on a recently published paper. I want to add some caveats to the discussion.

But - and this is a big one - this result is not even remotely close to being practical. We're talking multiple days to perform pretty simple calculations, using massively large blocks of computer code. And this is likely to remain true for a very long time. Unless researchers increase performance by many orders of magnitude, nothing in the real world will make use of this work anytime soon.

– Bruce Schneier, 23 November 2020.

Indistinguishability Obfuscation (iO)

But, consider fully homomorphic encryption. It, too, was initially theoretically interesting and completely impractical. And now, after decades of work, it seems to be almost just-barely maybe approaching practically useful. This could very well be on the same trajectory, and perhaps in twenty to thirty years we will be celebrating this early theoretical result as the beginning of a new theory of cryptography.

Schneier on Security , 23 November 2020.

Indistinguishability Obfuscation (iO)

- Some excellent recent lectures by leaders in the field, at a Simons Institute workshop: https://simons.berkeley.edu/news/new-developments-indistinguishability-obfuscation-io
- Will it become practical? What will it be for?
- Will it obey my "law"?

Summary

- mid-1970s: Public key
- 1985: ECC
- 1999-2001: Pairings
- 2008: FHE
- Now: isogenies and iO
- Each new idea much slower than the previous tools. Each using new mathematics. Each having potential new applications.

Conclusion

- New systems are built on more advanced mathematics.
- New systems start slow and eventually become practical, due to new ideas (not just Moore's law).
- The "killer application" is not the first application that is studied.

Lessons for future research leaders:

- Keep learning advanced mathematics, as you don't know what will turn out to be useful.
- Do not be deterred if a new idea is slow and seems impractical at the beginning.
- Keep an open mind about future applications.

Thank you for your attention.