Lecture 7: Privacy-Enhancing Technologies-1

-Post-quantum Cryptography and Fully Homomorphic Encryption

COMP 6712 Advanced Security and Privacy

Haiyang Xue

haiyang.xue@polyu.edu.hk

2024/3/4

Part of the slides are based on those of Huelsing, Gentry, and Jacobsen, Paillier, Viand, etc

Topic 1: Post-quantum Cryptography

- What is post-quantum cryptography?
- Why do we need post-quantum cryptography now?
- What is the status of post-quantum cryptography?
- What can we do further?

Topic 2: Fully Homomorphic Encryption

- What is fully homomorphic encryption (FHE)?

- What can we do with FHE?

- How could we achieve FHE?

- What is the status of FHE?

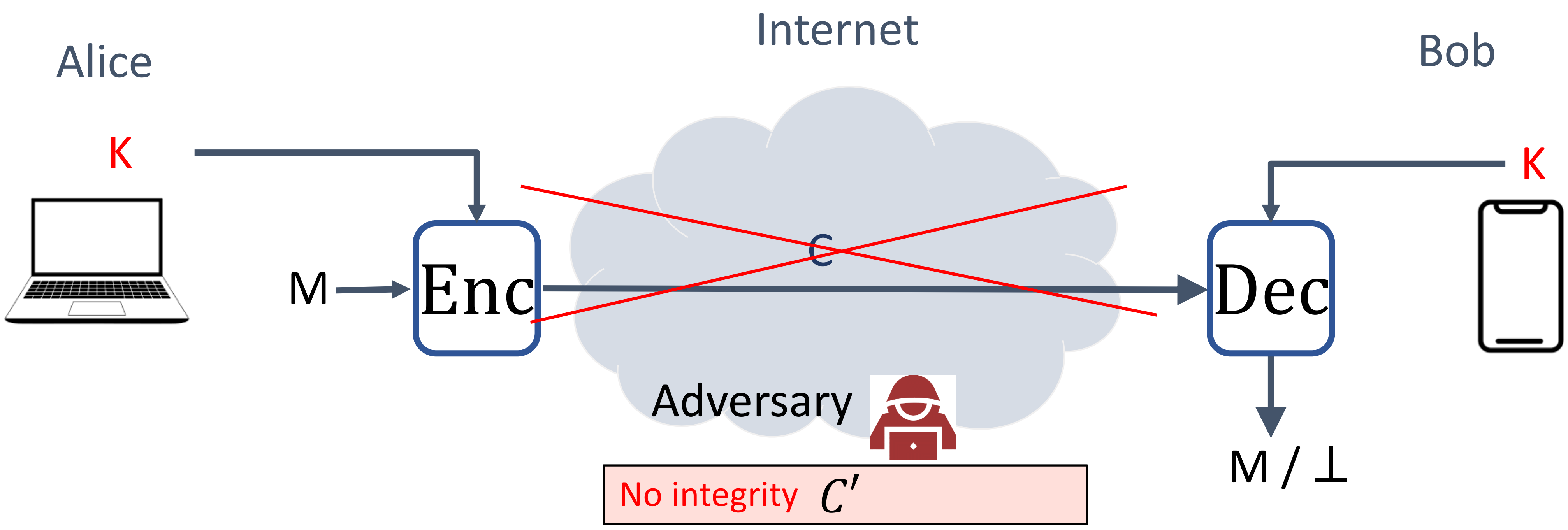
Our aim

- We do not aim to know all the constructions, and security proof of these topic.
- We try to understand their status and recent development.

- Each of them is a BIG topic and deserves an individual course.
- Due to their importance, I can not ignore them in this lecture.

Post-quantum Cryptography

Symmetric-key cryptography



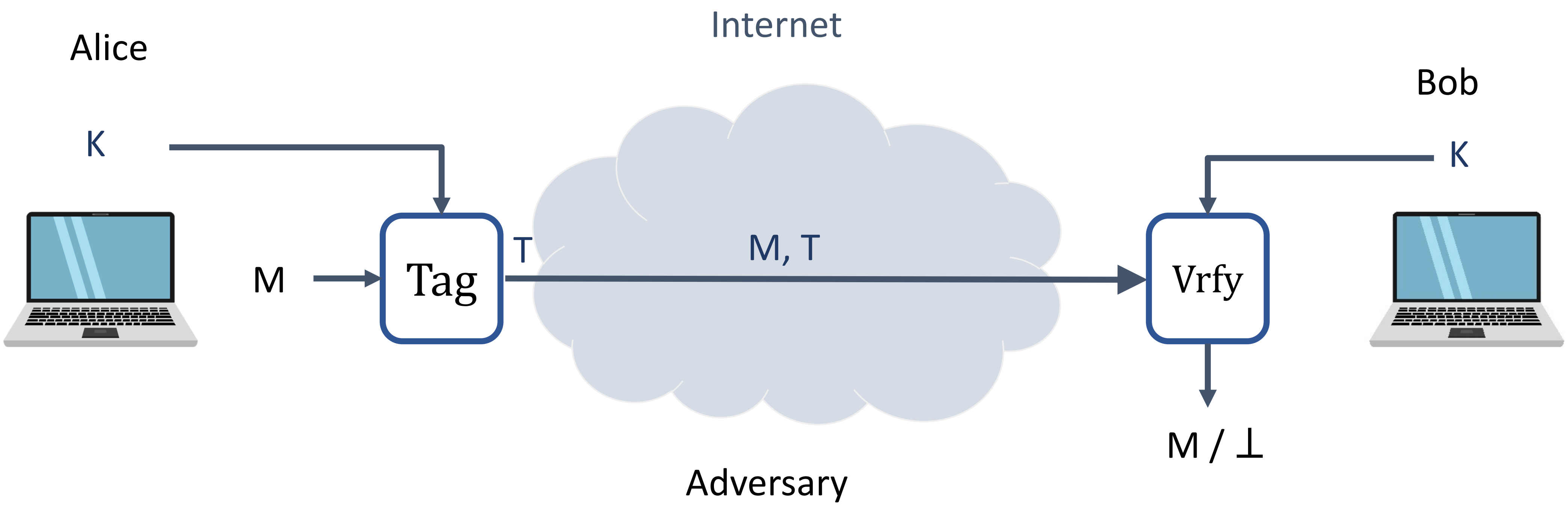
Enc : encryption algorithm (public)

Dec : decryption algorithm (public)

K : shared key between Alice and Bob

How to achieve this??

Achieving integrity: MACs

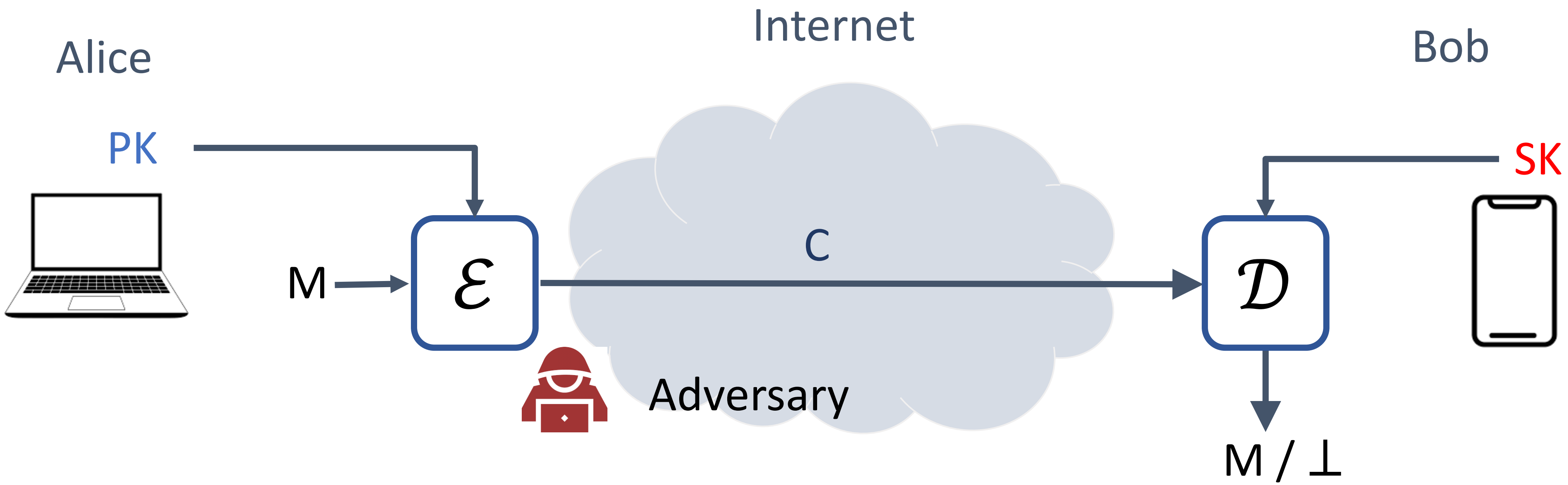


Tag : tagging algorithm (public)

K : tagging / verification key (secret)

Vrfy: verification algorithm (public)

Public-key Encryption



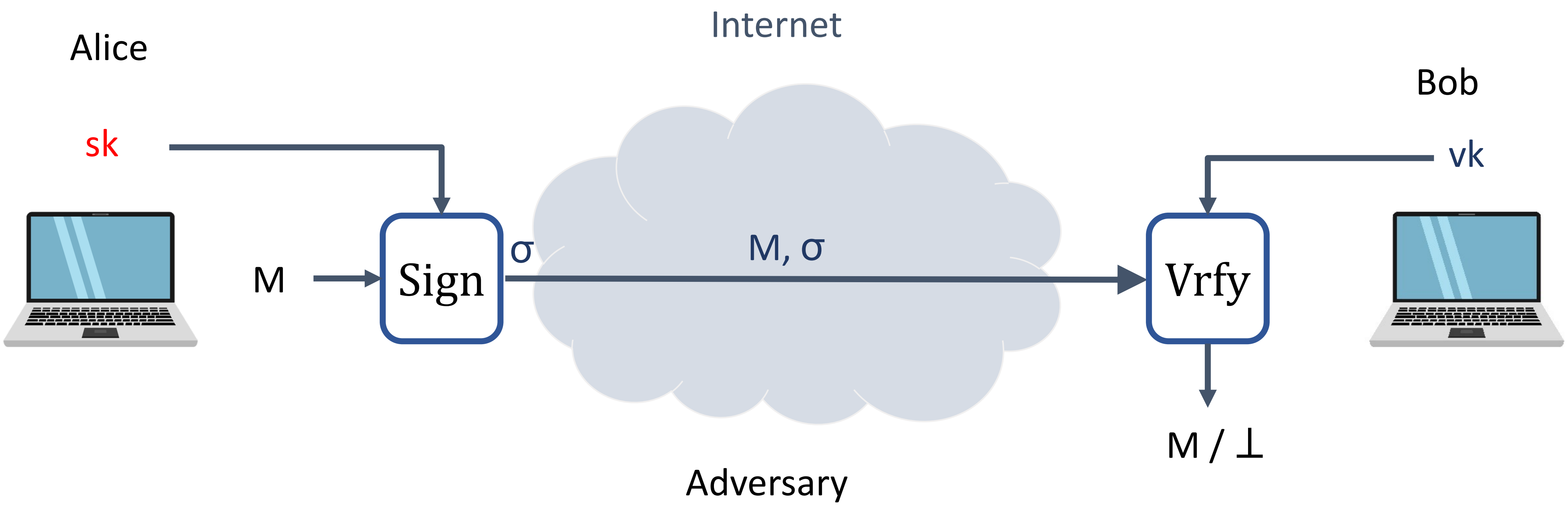
Enc: encryption algorithm (public)

Dec : decryption algorithm (public)

PK : public key of Bob (public)

SK : secret key (secret)

Achieving integrity: digital signatures



Sign : tagging algorithm (public)

sk : signing key (secret)

Vrfy : verification algorithm (public)

vk : verification key (public)

Basic goals of cryptography

	IND-eva, IND-CPA IND-CCA	UF-CMA
AE	**Message privacy**	**Message integrity / authentication**
Symmetric keys	Symmetric encryption	Message authentication codes (MAC)
Public keys	public-key encryption (Key exchange)	Digital signatures
	IND-CPA, IND-CCA	UF-CMA

Unkeyed primitives Hash functions

Collision resistance, one-wayness

Basic goals of cryptography

Encrypt-then-MAC
AES-GCM
AES-CCM
AES-OCB

	AES-CTR, AES-CTR\$ AES-CBC\$	PRF, CBC-MAC, HMAC
	Message privacy	**Message integrity / authentication**
Symmetric keys	Symmetric encryption	Message authentication codes (MAC)
Public keys	Padded RSA, ElGamal public-key encryption (Diffie-Hellman Key exchange)	Digital signatures Hashed RSA, Schnorr, ECDSA

Unkeyed primitives Hash functions SHA2-256, SHA2-512 SHA3-256, SHA3-512

Security in practice based on cryptography

- Communication security (Signature, PKE / Key Exchange)
  - TLS, SSH, IPSec, ...
  - eCommerce, online banking, eGovernment, ...
  - Private online communication

- Code signing (Signature)
  - Software updates
  - Software distribution

Recall how to build PKC and Symmetric key cryptography

(Computationally)
hard problem

FACT
RSA
DL DDH

→

PKC Scheme

RSA Diffie-Hellman ECDSA

FACT: $N = pq \rightarrow$ p, and q

DL: $g^x \rightarrow x$

RSA : $x^e \; mod \; N \rightarrow x$

DDH: $g^a, g^b, g^c \rightarrow c =? \, ab$

Symmetric key

AES SHA-2 SHA-3 HMAC

The Quantum Threat-Shor‘s algorithm (1994)

- Quantum computers can do "Fast Fourier Transform" (FFT) very efficiently

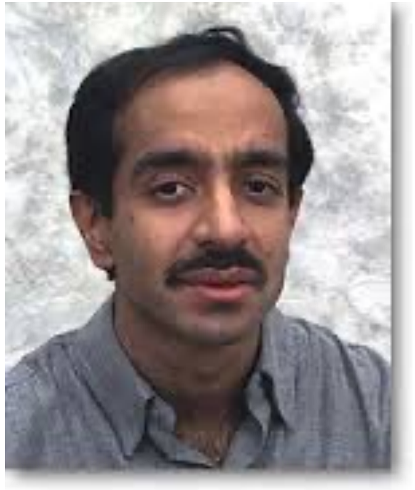
- Can be used to find period of a function

- This can be exploited to factor efficiently (RSA)
- Shor also shows how to solve discrete log efficiently (Diffie-Hellman, ECDSA, ECDH)

Shor, P.W. (1994). "Algorithms for quantum computation: discrete logarithms and factoring"

The Quantum Threat-Grover‘s algorithm (1996)

- Quantum computers can search N entry Date Base in $\Theta(\sqrt{N})$



- Application to symmetric cryptography

- Nice: Grover is provably optimal
- Then, double security parameter.

Grover, Lov K. (1996-07-01). "A fast quantum mechanical algorithm for database search"

Take RSA as example: Factoring to order-finding

$N = pq$

$|\langle a \rangle| = r$

$$a^1, a^2, a^3, \dots, \underbrace{a^r}_{}\overset{=1}{}, a^1, a^2 \dots \pmod{N}$$

order of a = the smallest positive r such that $a^r = 1 \pmod{N}$

Euler's theorem: for all $a \in \boldsymbol{Z}_N^*$

$$a^{\phi(N)} = a^{(p-1)(q-1)} = 1 \pmod{N}$$

Fact: r must divide $(p-1)(q-1)$

This is where the quantum magic happens!
Shor's algorithm

Conclusion: learn $r \implies$ we learn a factor of $(p-1)(q-1)$

repeat with a different $a \implies$ learn another factor of $(p-1)(q-1)$ (with high prob.)

eventually we learn full $(p-1)(q-1) \implies$ can find p and q

The Quantum Threat

- When will a quantum computer be built that breaks current crypto?
  - 15 years, $1 billion USD (to break RSA-2048)
  - (PQCrypto 2014, Matteo Mariantoni)

- Impact:
- Public key crypto
  - RSA
  - Elliptic Curve Cryptography (ECDSA)
  - Finite Field Cryptography (DSA)
  - Diffie-Hellman key exchange
- Symmetric key crypto
  - AES
  - Triple DES

- Hash functions
  - SHA-1, SHA-2 and SHA-3

The Quantum Threat

- When will a quantum computer be built that breaks current crypto?
  - 15 years, $1 billion USD (to break RSA-2048)
  - (PQCrypto 2014, Matteo Mariantoni)

- Impact:
- Public key crypto

  Post-quantum Cryptography
  - ~~RSA~~
  - ~~Elliptic Curve Cryptography (ECDSA)~~
  - ~~Finite Field Cryptography (DSA)~~
  - ~~Diffie-Hellman key exchange~~
- Symmetric key crypto
  - AES
  - Triple DES

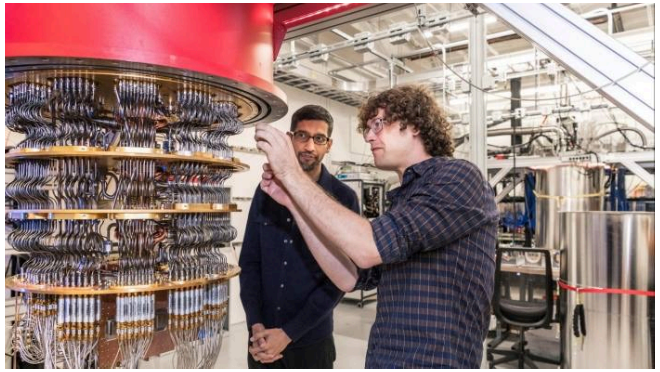
  Needs larger keys

- Hash functions
  - SHA-1, SHA-2 and SHA-3

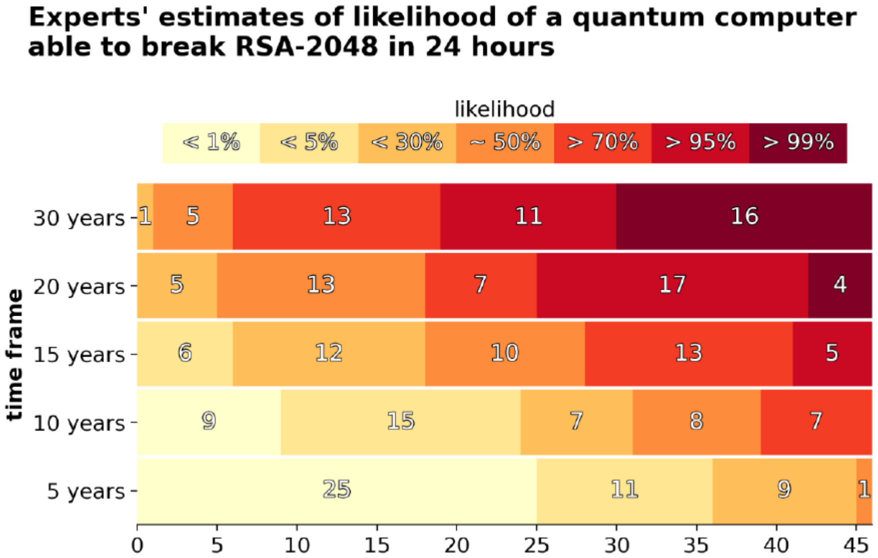
  Needs longer outputs

Why care today??

- Now, we only have quantum computer <100 qubits



- It is not known for certain when
- a large scale quantum computer could be
- built, although experts said it may be
- possible within the next two decades



https://globalriskinstitute.org/publications/2021-quantum-threat-timeline-report/

Why care today??

- Some one store all encrypted data traffic
- Wait for the large-scale quantum computer

- Development time easily 10+ years
- Lifetime easily 10+ years

What about quantum key distribution (QKD)

- The problem solved by QKD

“Key growing”

(≠ “Key establishment“)

Given

- a shared classical secret,
- a physical channel between
- compatible QKD devices on both ends of the channel

It is possible to

- generate a longer shared classical secret.

- In August 2016, China launched world's first quantum communications satellite Mozi (墨子号)

- Quantum cryptography (QKD)
  - Use quantum mechanics to build cryptography

- Post-quantum cryptography
  - Classical algorithms believed to withstand quantum attacks

Post-quantum cryptography

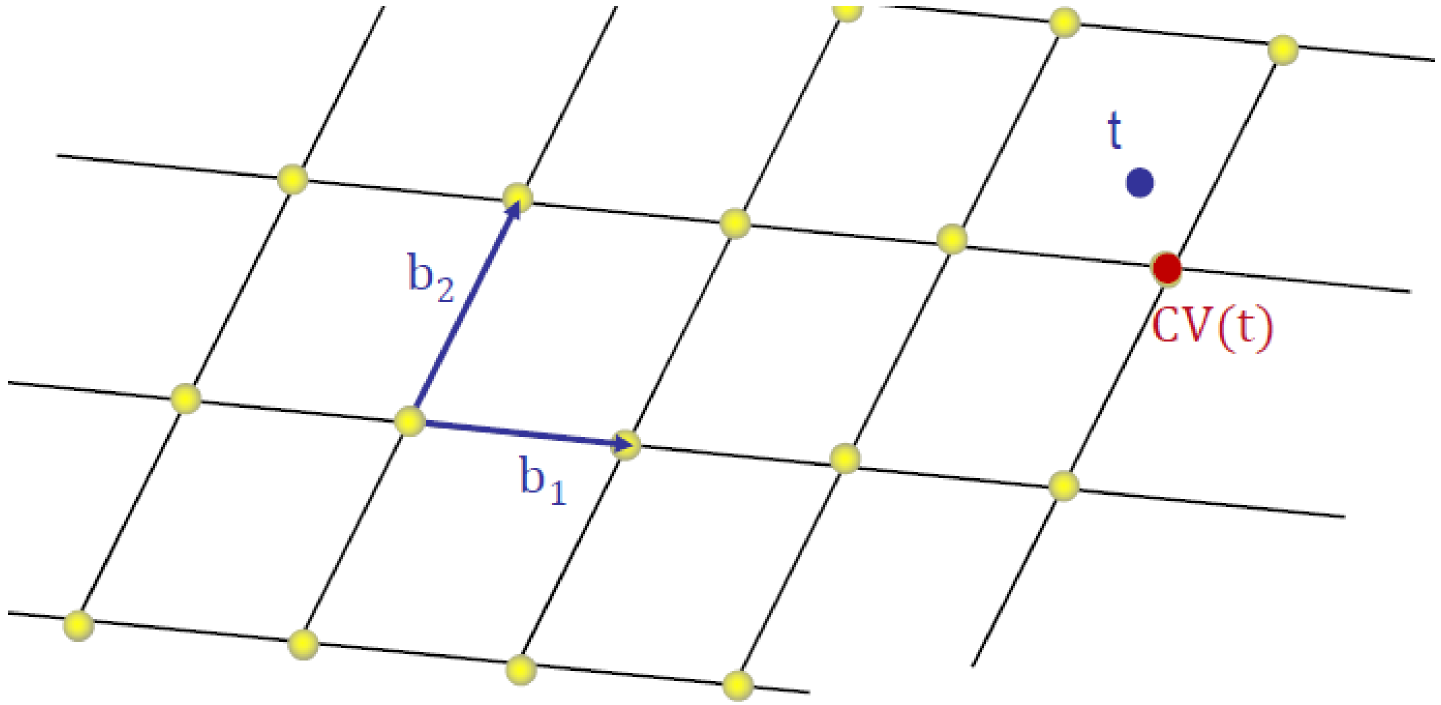
- **Top candidates:**
  - Lattice-based cryptography
  - Code-based cryptography
  - Multivariate cryptography
  - Hash-based cryptography
  - Isogeny-based cryptography

Cryptographic Standards in the Post-Quantum Era
https://ieeexplore.ieee.org/document/9935575

Lattice-based cryptography

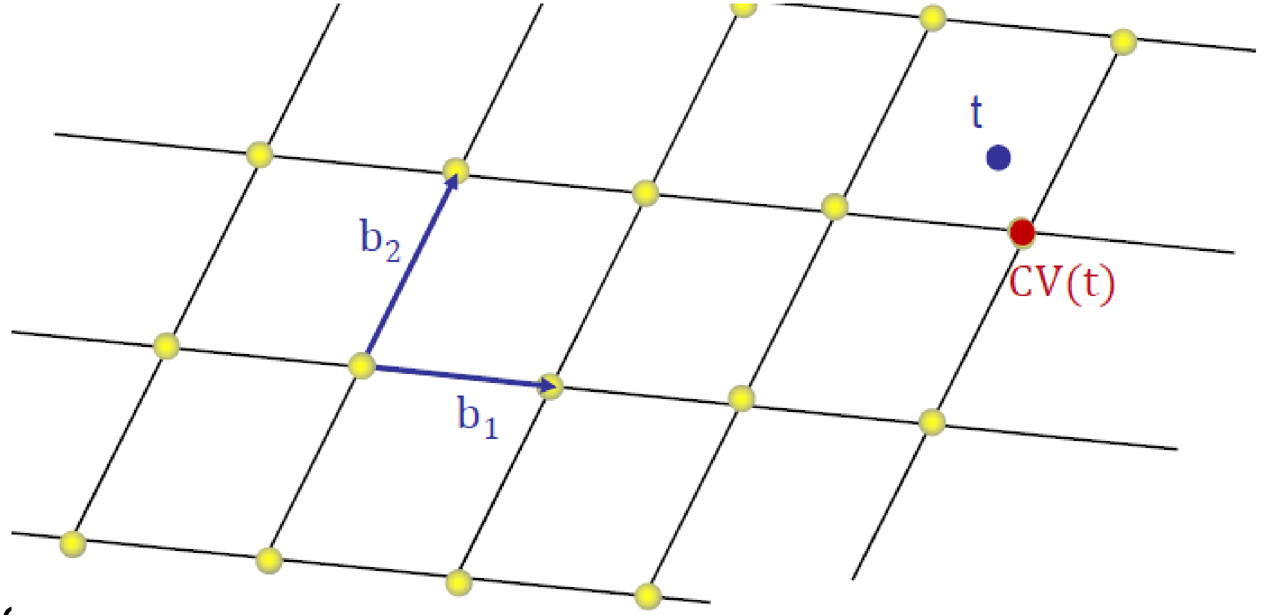
- Expected to resist quantum computer attacks

- Permits fully homomorphic encryption



Lattice-based cryptography

- Hard problems
- $n \in \mathbb{N}$, base $B = (b_1, b_2, \cdots, b_n)$
- Lattice $L = \{\mathbb{Z}\, b_1 + \cdots + \mathbb{Z} b_n\}$



- Shortest vector problem (SVP)
  - Given B
  - find the shortest nonzero $v \in L$

- Closest vector Problem (CVP)
  - Given B and $t \in \mathbb{R}^n$
  - Find $CV(t) \in L$ such that $|CV(t) - t|$ is the sho

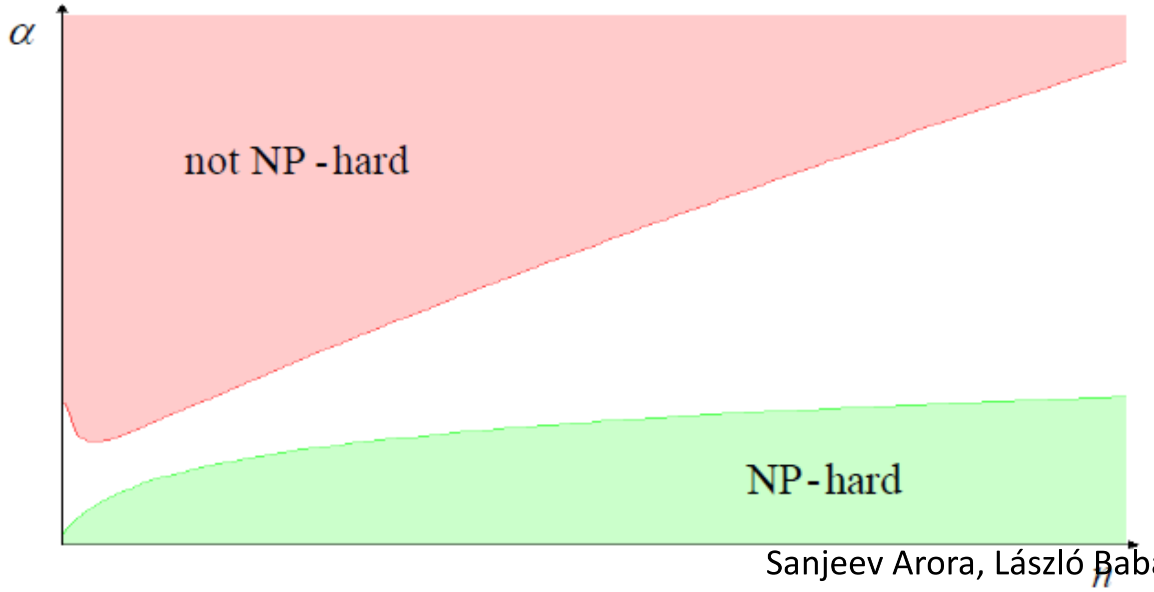
- For 2-dimension case, SVP and CVP are easy.
- For general large n, SVP and CVP are NP-hard

Lattice-based cryptography

- Hard problems
- Approximate Closest vector Problem (α CVP)
  - Given B and $t \in \mathbb{R}^n$
  - Find $CV(t) \in L$ such that $|CV(t) - t| < \alpha \min_{v \in L} |v - t|$

$\log(n)^c$ - CVP is NP - hard for all c

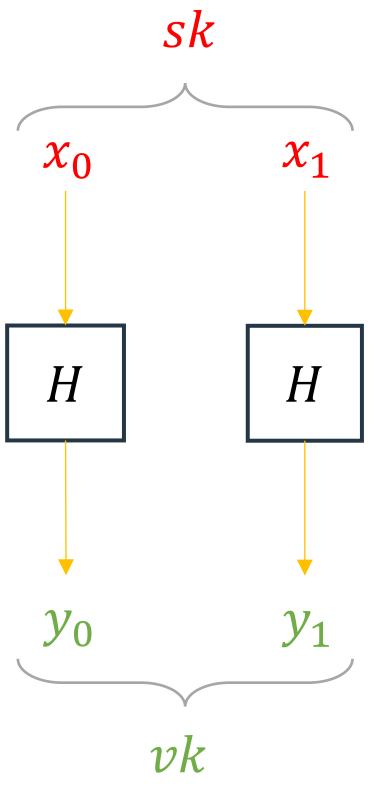
- Arora et al.



Sanjeev Arora, László Babai, Jacques Stern, Z. Sweedyk: The Hardness of Approximate Optimia in Lattices, Codes, and Systems of Linear Equations. FOCS 1993: 724-733

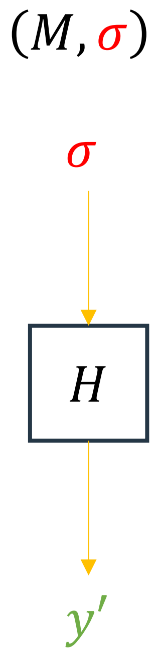
Hash-based signatures – Lamport's 1-time signature

$\mathcal{SK} = \{0,1\}^{512}$ $\mathcal{VK} = \{0,1\}^{512}$ $\mathcal{M} = \{0,1\}$ $\mathcal{S} = \{0,1\}^{256}$



$\text{Sign}(sk, 0) = x_0$

$\text{Sign}(sk, 1) = x_1$

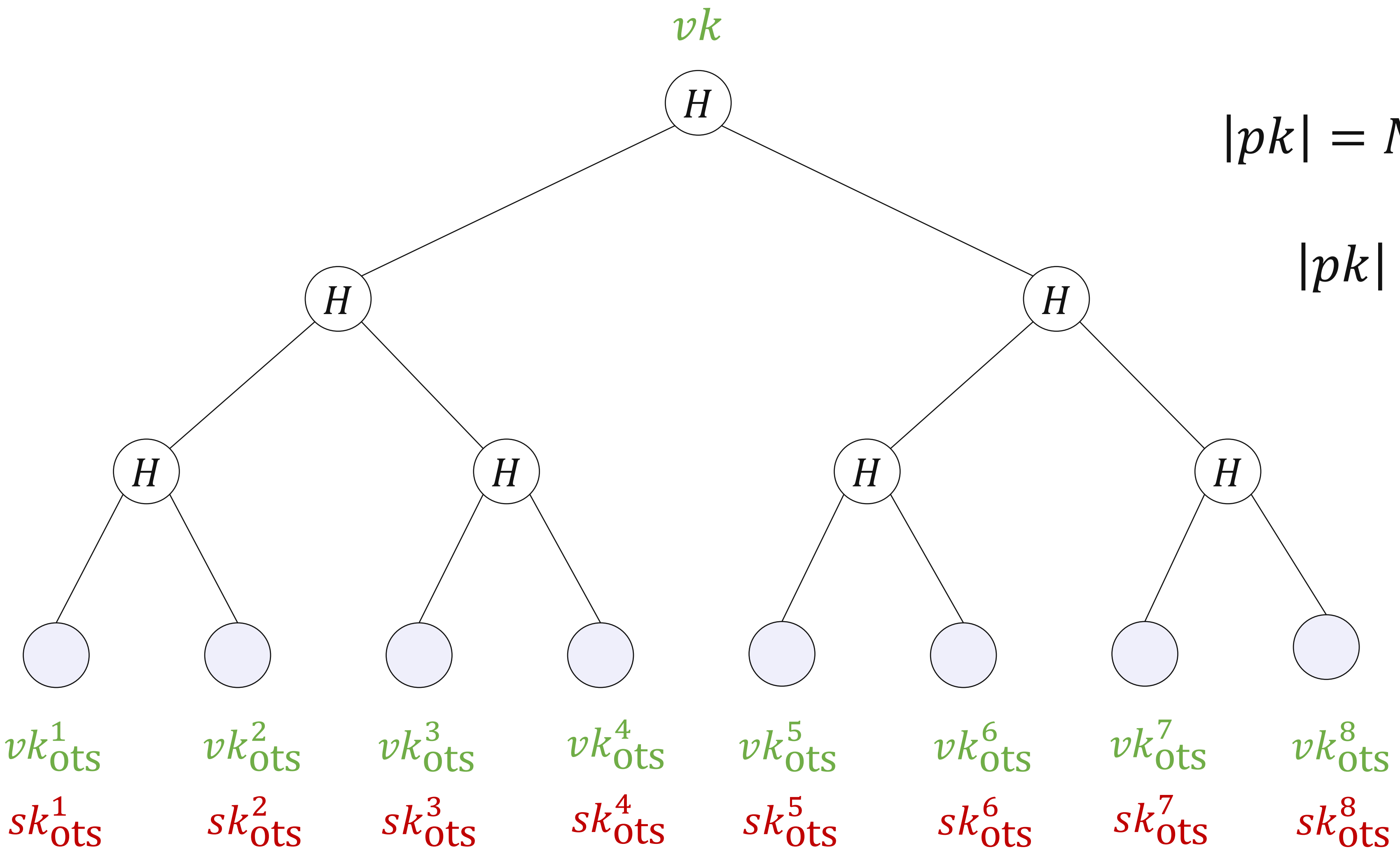


$\text{Vrfy}(vk, 0, \sigma) = (H(\sigma) \stackrel{?}{=} y_0)$

$\text{Vrfy}(vk, 1, \sigma) = (H(\sigma) \stackrel{?}{=} y_1)$

$y' \stackrel{?}{=} y_M$

Hash-based signatures–multitime signature Merkle tree

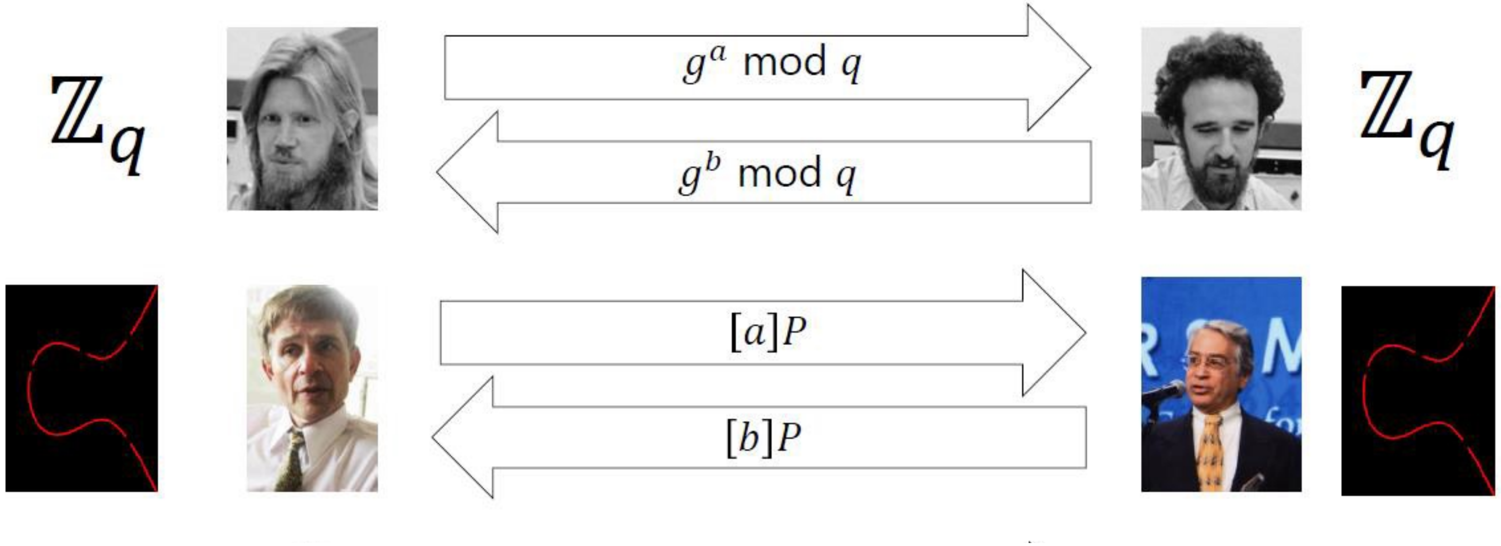


$$|pk| = N \cdot |pk_{\text{ots}}|$$

$$|pk| = |H|$$

Isogeny-based Cryptography

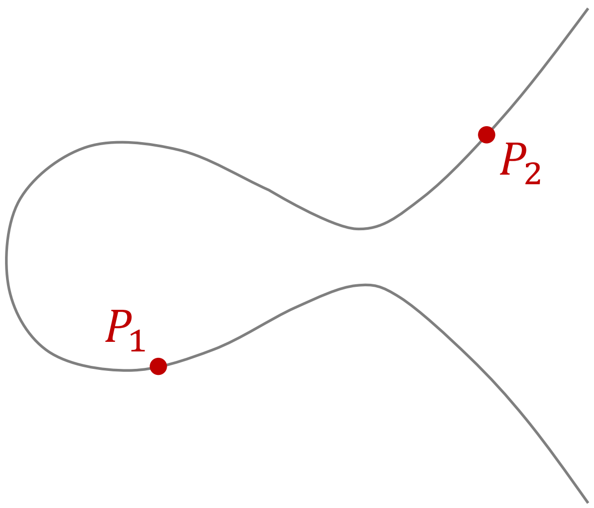
- Over a class of Super-singular Elliptic curves
- Isogeny: maps between (supersingular) elliptic curves that respect their group structure



Isogeny-based Cryptography

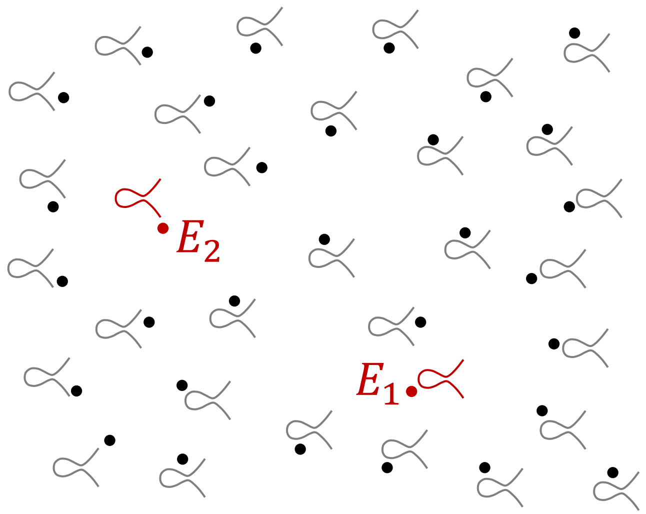
Isogeny-based cryptography = new kind of elliptic curve cryptography,
supposed to be resistant against quantum computers

classical elliptic curve cryptography



based on hidden relation between two points on an elliptic curve

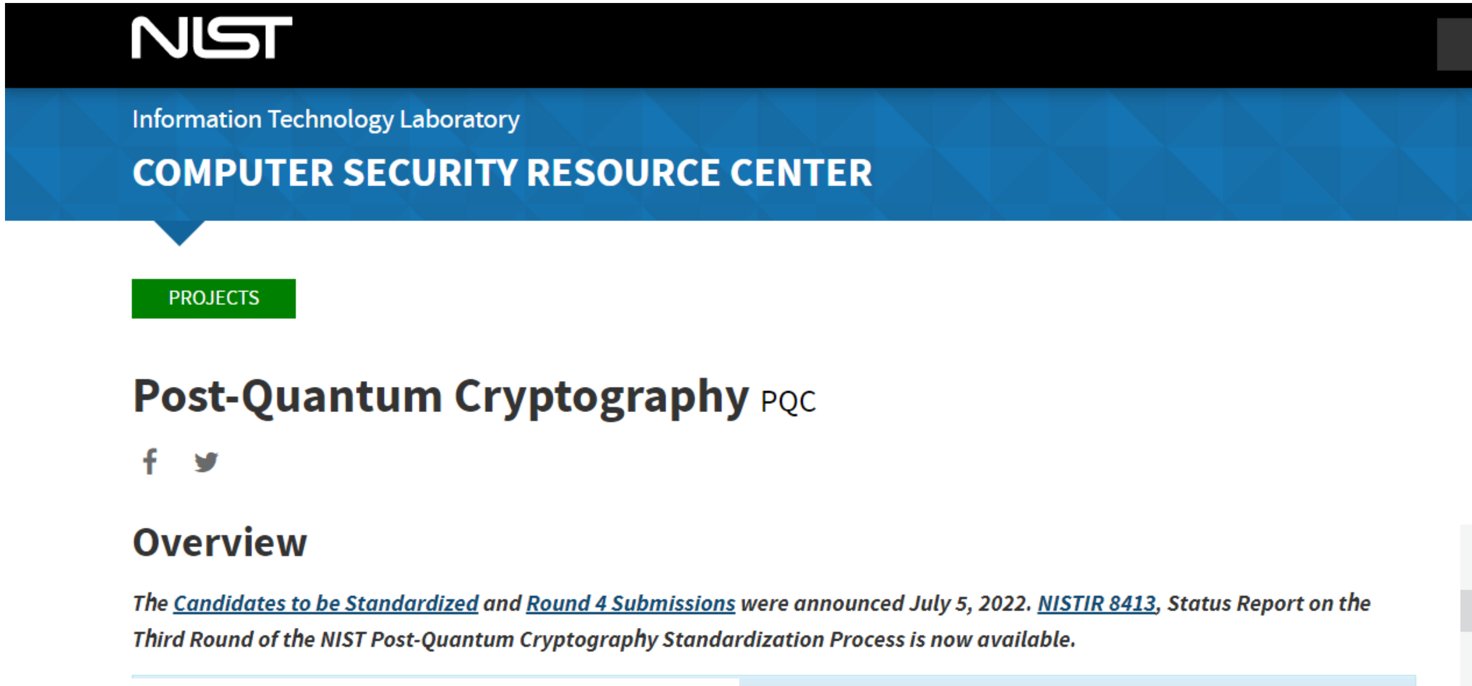
isogeny-based cryptography

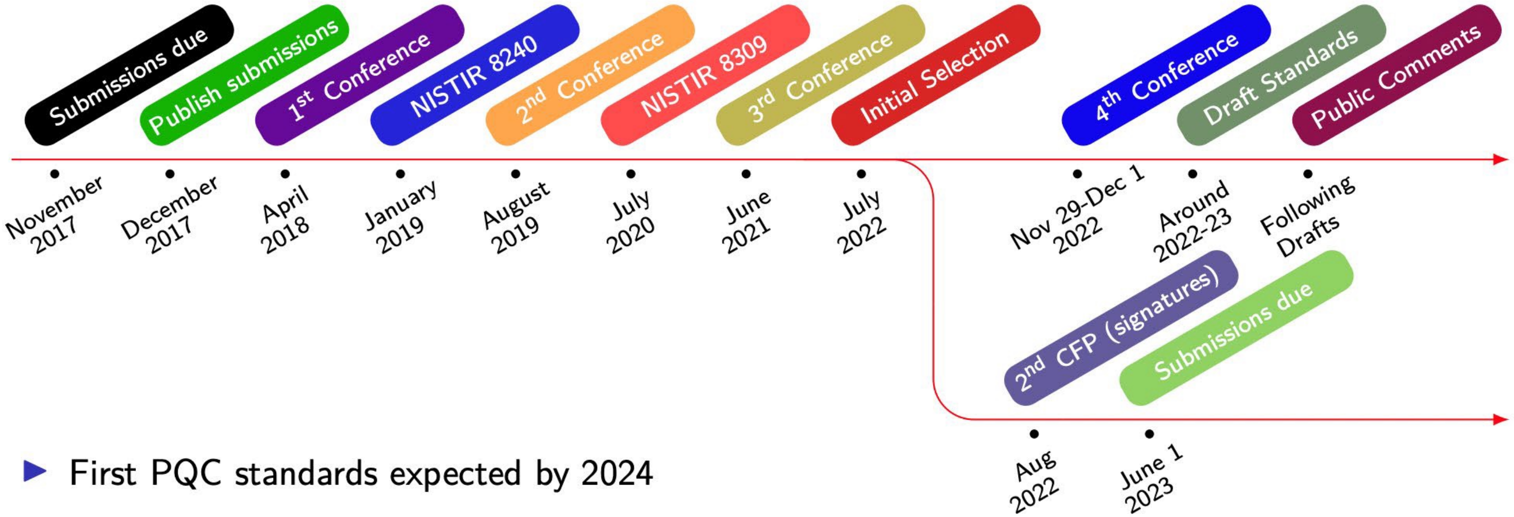


based on hidden relation between two elliptic curves in an isogeny class

The NIST post-quantum competition

- The NIST PQC Standardization Process began in 2016 with a call for proposals for post-quantum digital signatures and post-quantum PKE





▶ First PQC standards expected by 2024

https://csrc.nist.gov/projects/pos quantum-cryptography

The NIST post-quantum competition

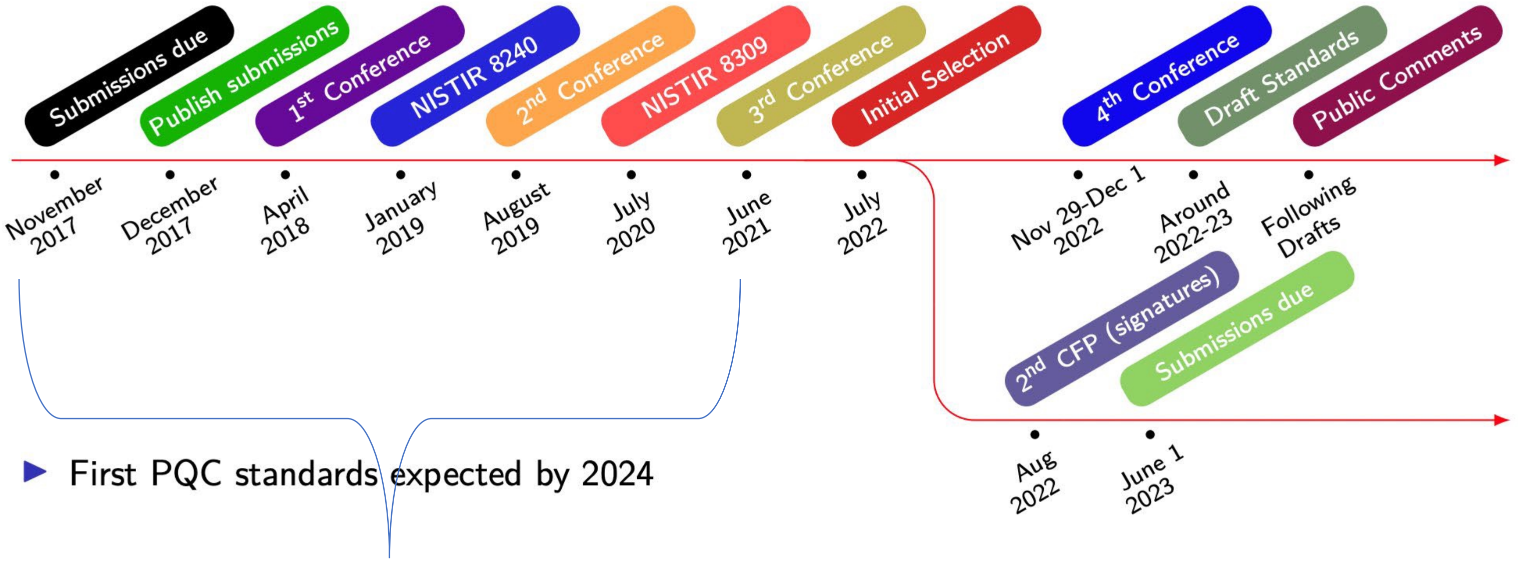
- Public competition to standardize post-quantum schemes
  - Public-key encryption
  - Digital signatures
- Started in 2017
  - Round 1: 69 submissions
  - Round 2: 26 candidates selected
  - Round 3: 15 candidates selected
  - Round 3 winner
  - Round 4 Candidates and Call for proposals
- Round 3 winners: Kyber, Dilithium, Falcon

SPHINCS+

Algorithm (public-key encryption)	Problem
Classic McEliece	Code-based
CRYSTALS-KYBER	Lattice-based
NTRU	Lattice-based
SABER	Lattice-based
BIKE	Code-based
FrodoKEM	Lattice-based
HQC	Code-based
NTRU Prime	Lattice-based
SIKE	Isogeny-based

Algorithm (digital signatures)	Problem
CRYSTALS-DILITHIUM	Lattice-based
FALCON	Lattice-based
Rainbow	Multivariate-based
GeMSS	Multivariate-based
Picnic	ZKP
SPHINCS+	Hash-based

The NIST post-quantum competition



We design a lattice-based scheme, **LAC**. It advances to Round 2.

https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions

What can we do further?

- Deployment of PQC in TLS, SSL, etc.
  - Test of PQC in chrome TLS, chrome://flags/
- More post-quantum signature candidates?
  - Current PQ signatures are either slow or has a large signature size
- More cryptographic algorithms, such as authenticated key exchange.

PKE ---------- > PQC

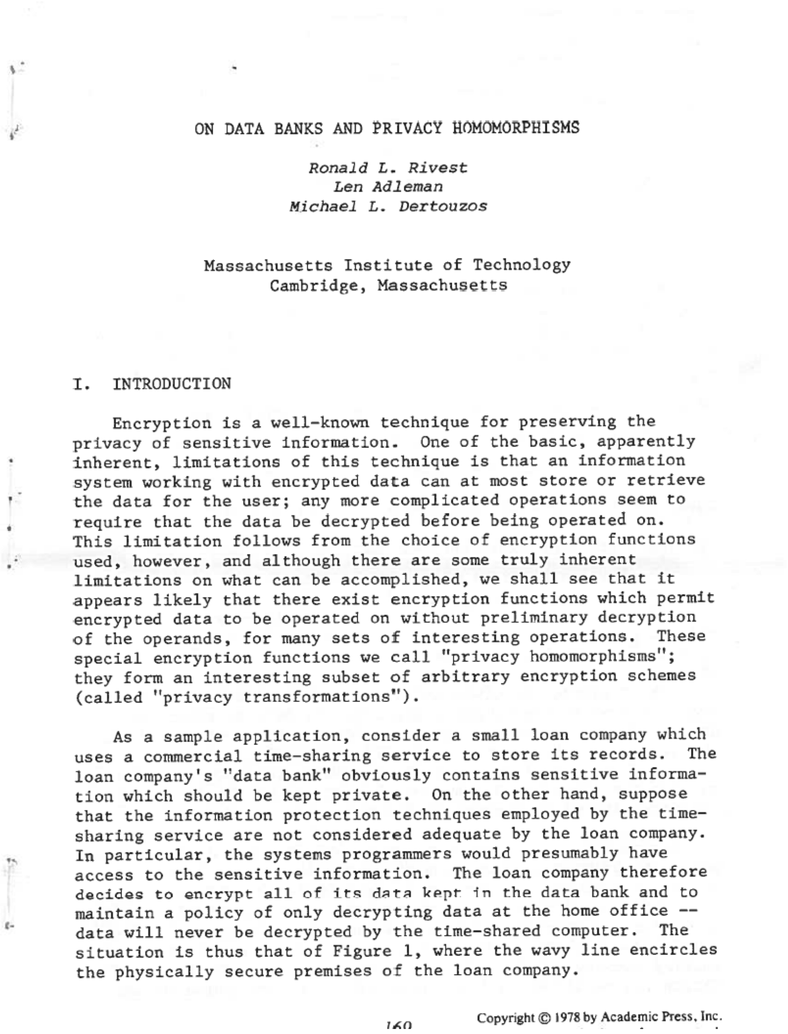
Diffie-Hellman RSA-OAEP ElGamal	--------- >	Kyber	https://pq-crystals.org/kyber/
Hash-RSA ECDSA Schnorr	--------- >	Dilithium Falcon SPHINCS+ etc.	https://pq-crystals.org/dilithium https://falcon-sign.info/ https://sphincs.org/

In summary

- Post-quantum cryptography aims to design alternatives of classical public key encryption (such as RSA, ECDSA), such that they are still secure against quantum computer.
- We need to do this right now.
- NIST has done a great effort. From lattice, hash, code, isogeny based cryptography.
- but we still need to do more.
- Or you can work on designing large scale quantum computer.

Fully Homomorphic Encryption

Homomorphic Encryption



ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

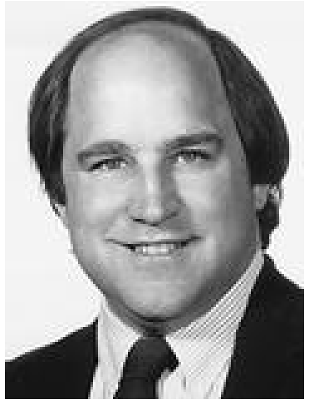
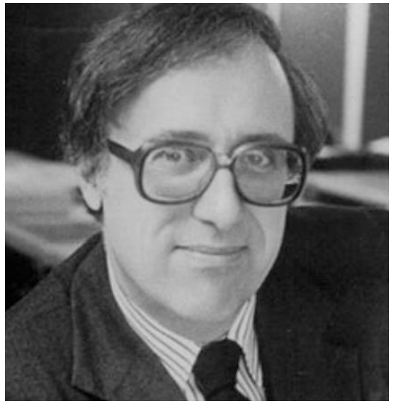
Massachusetts Institute of Technology
Cambridge, Massachusetts

I. INTRODUCTION

Encryption is a well-known technique for preserving the privacy of sensitive information. One of the basic, apparently inherent, limitations of this technique is that an information system working with encrypted data can at most store or retrieve the data for the user; any more complicated operations seem to require that the data be decrypted before being operated on. This limitation follows from the choice of encryption functions used, however, and although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call "privacy homomorphisms"; they form an interesting subset of arbitrary encryption schemes (called "privacy transformations").

As a sample application, consider a small loan company which uses a commercial time-sharing service to store its records. The loan company's "data bank" obviously contains sensitive information which should be kept private. On the other hand, suppose that the information protection techniques employed by the time-sharing service are not considered adequate by the loan company. In particular, the systems programmers would presumably have access to the sensitive information. The loan company therefore decides to encrypt all of its data kept in the data bank and to maintain a policy of only decrypting data at the home office -- data will never be decrypted by the time-shared computer. The situation is thus that of Figure 1, where the wavy line encircles the physically secure premises of the loan company.

160 Copyright © 1978 by Academic Press, Inc.

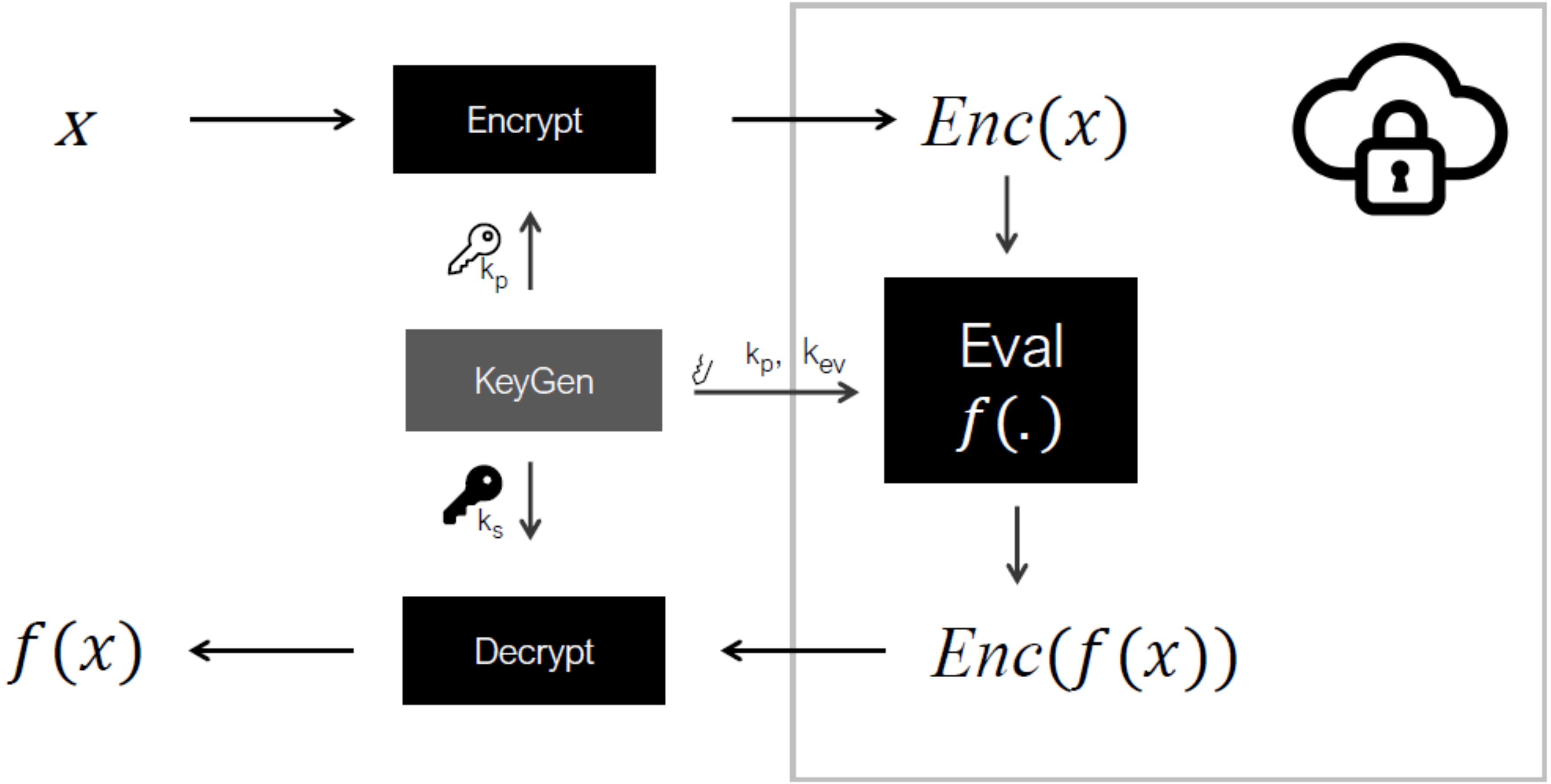
  

1978: Rivest, Adleman, Dertouzos,
"On Data Banks and $\boxed{\text{Privacy Homomorphisms}}$"

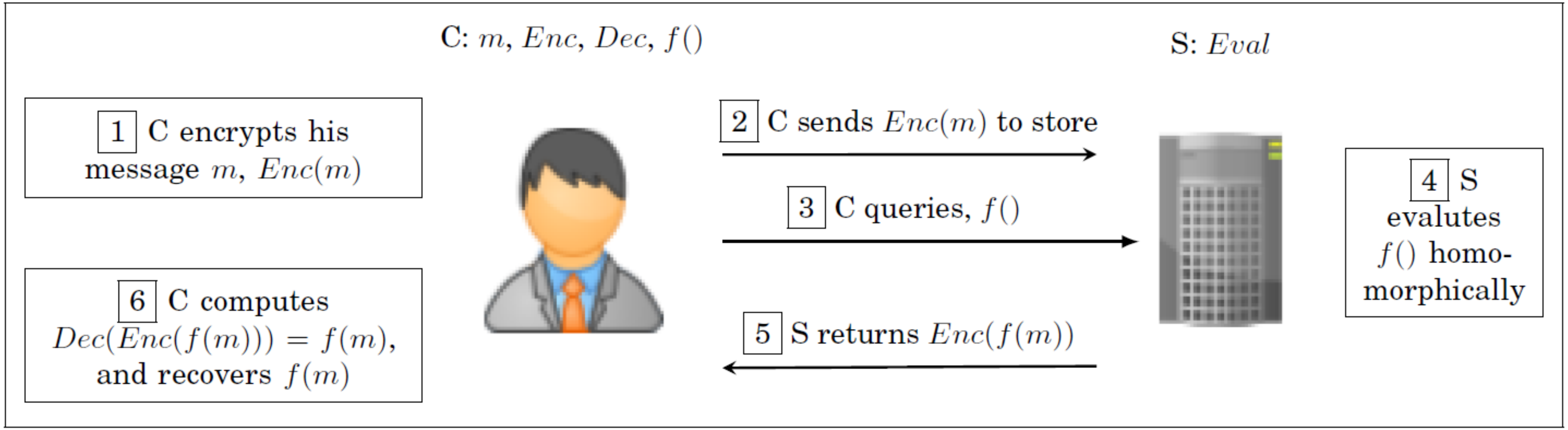
Homomorphic Encryption

Can we delegate the processing of data without giving away access to it

If we have an ideal encryption



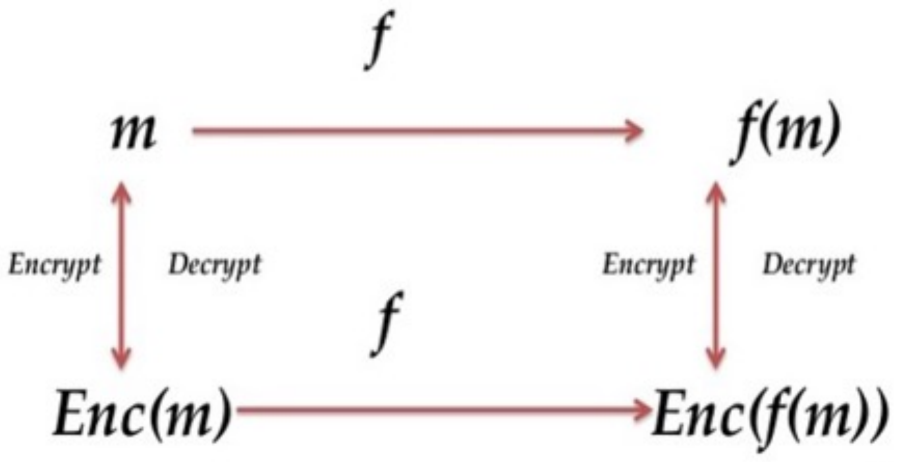
Cloud Computing on Encrypted Data



Abbas A , Hidayet A , Selcuk U A , et al (2017) A Survey on Homomorphic Encryption Schemes: Theory and Implementation

Homomorphic encryption(HE)

- A homomorphic encryption(HE) scheme allows
- computations on the ciphertext without knowing the secret key,
- meanwhile ensures that the decryption of the resulting ciphertext is exactly the same as the computations over the plaintext.



Formally, we define FHE

- A homomorphic encryption scheme is a tuple (HE.KeyGen, HE.Enc, HE.Dec,HE.Eval) of probabilistic polynomial time (PPT) algorithms.

- (sk; pk; evk) ← HE:KeyGen
- c ← HE:Enc(pk; m)
- m ← HE:Dec(sk; c)

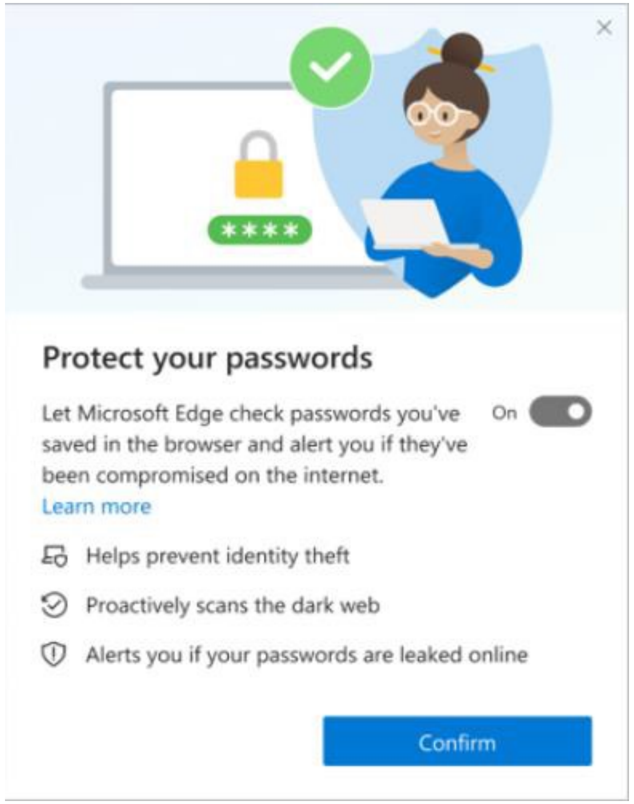
- For function f from a set S

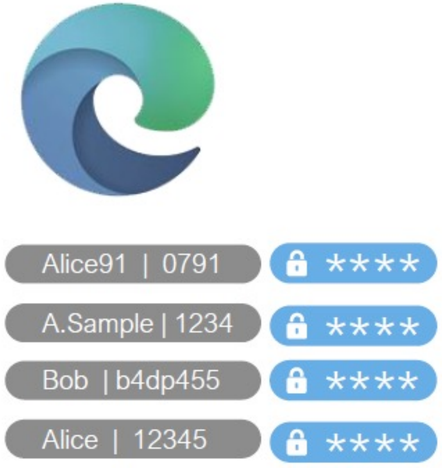
$$c_f \leftarrow \text{HE:Eval}(pk;\ f\ ;\ evk;\ c_1;\ \cdots;\ c_l\ ),$$

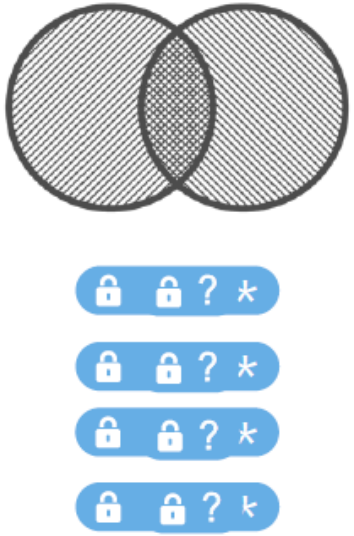
where c_i =HE:Enc(pk; m_i)

- We have $f(m_1, m_2, \cdots, m_l) = \text{HE:Dec}(sk;\ c_f)$

More applications in practice

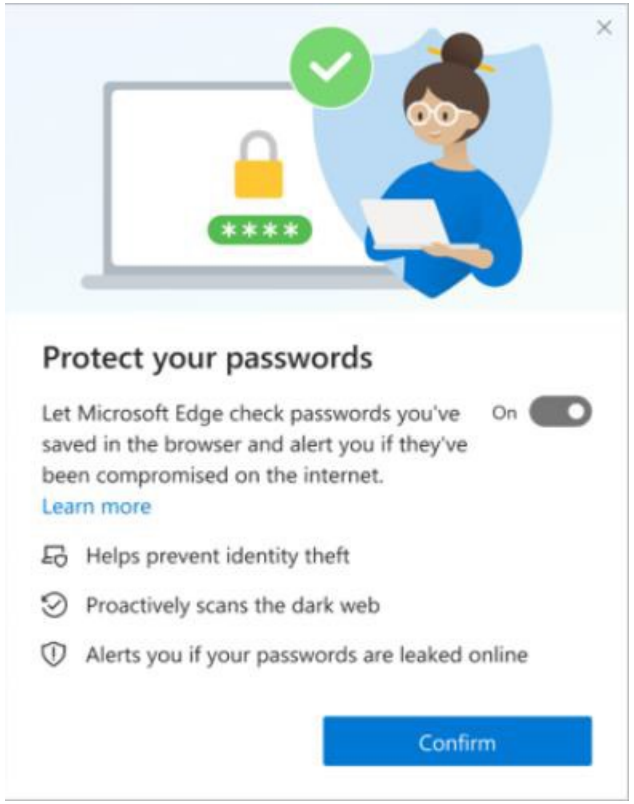


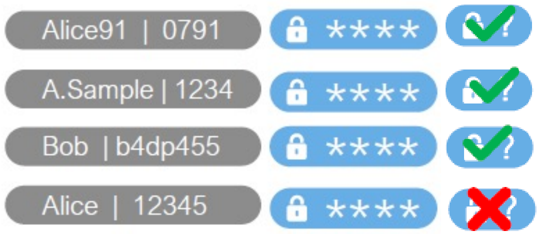


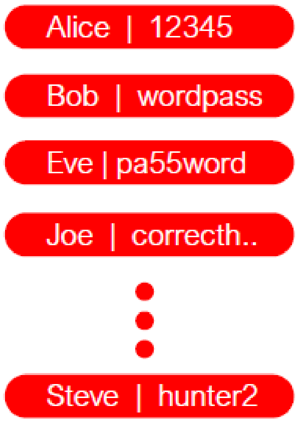


Delegate the processing of data without giving away access to it

More applications in practice







Delegate the processing of data without giving away access to it

If f is only multiplication

$N = pq$

RSA Enc $\quad M \xrightarrow{\mathbf{E}} C \leftarrow M^e \bmod N \xrightarrow{\mathbf{D}} M \leftarrow C^d \bmod N$

$$E(m_1) = m_1{}^e \qquad E(m_2) = m_2{}^e$$

HE:Eval $\quad E(m_1)\times E(m_2)$

$$= m_1{}^e \times m_2{}^e$$
$$= (m_1\times m_2)^e$$
$$= E(m_1\times m_2)$$

$$\boxed{E(m_1)\times E(m_2) = E(m_1\times m_2)}$$

If f is only addition

- Paillier Encryption

  $N = pq$

- Enc: c $\leftarrow Enc(m) = (1+N)^m r^N mod\ N^2$

- Dec: $c^{\phi(N)} = (1+N)^{m\phi(N)}\ r^{N\phi(N)} mod\ N^2$

$$= (1+N)^{m\phi(N)}\ 1\ mod\ N^2$$

$$= 1 + m\phi(N)N\ \ mod\ N^2$$

Euler's theorem: for all $a \in \boldsymbol{Z}_N^*$

$$a^{\phi(N)} = a^{(p-1)(q-1)} = 1 \pmod N$$

Euler's theorem: for all $a \in \boldsymbol{Z}_{N^2}^*$

$$a^{N\phi(N)} = a^{N(p-1)(q-1)} = 1 \pmod{N^2}$$

Paillier, Pascal (1999). "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes"

If f is only addition

$N = pq$

Paillier Enc $\quad M \xrightarrow{\mathbf{E}} C \leftarrow (1+N)^m r^N \bmod N^2 \xrightarrow{\mathbf{D}} C^{\phi(N)} \bmod N^2$

$$E(m_1) = (1+N)^{m_1} r_1^N \qquad E(m_2) = (1+N)^{m_2} r_1^N$$

HE:Eval
$$\begin{aligned} E(m_1) \times E(m_2) &= (1+N)^{m_1} r_1^N \times (1+N)^{m_2} r_1^N \\ &= (1+N)^{m_1+m_2} r_1^N r_2^N \\ &= E(m_1+m_2) \end{aligned}$$

Paillier, Pascal (1999). "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes"

Partially HE --- > Fully HE

- Both RSA, Paillier are partially homomorphic encryption

- How to achieve Fully homomorphic encryption?
- Where function f could be any polynomial size circuit or polynomial function.

Partially HE --- >Somewhat HE-----> Fully HE

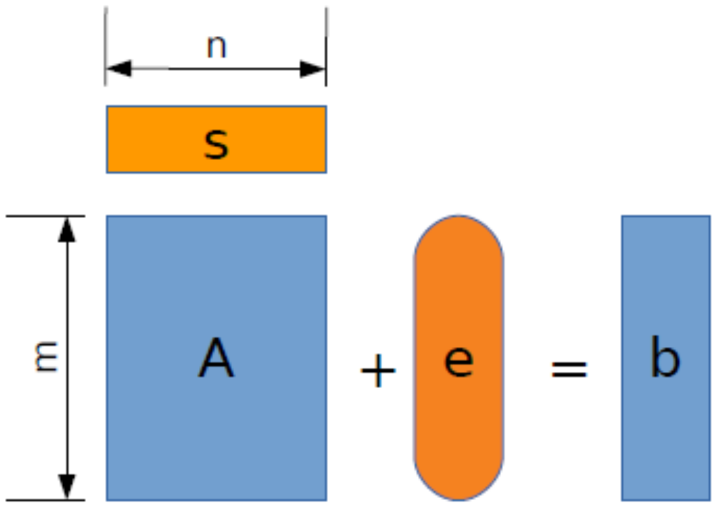
- Both RSA, Paillier are partially homomorphic encryption

- We first handle somewhat HE, which supports both add and multi in a very limited level.

Learning With Errors (LWE)

LWE function family:

- Key: $A \in Z_q[n \times m]$
- $LWE_A(s,e) = As + e \pmod q$
- Small $|e|_{max} < \beta = O(\sqrt{n})$
- q,m=poly(n)



Regev (2005): assuming quantum hard lattice problems

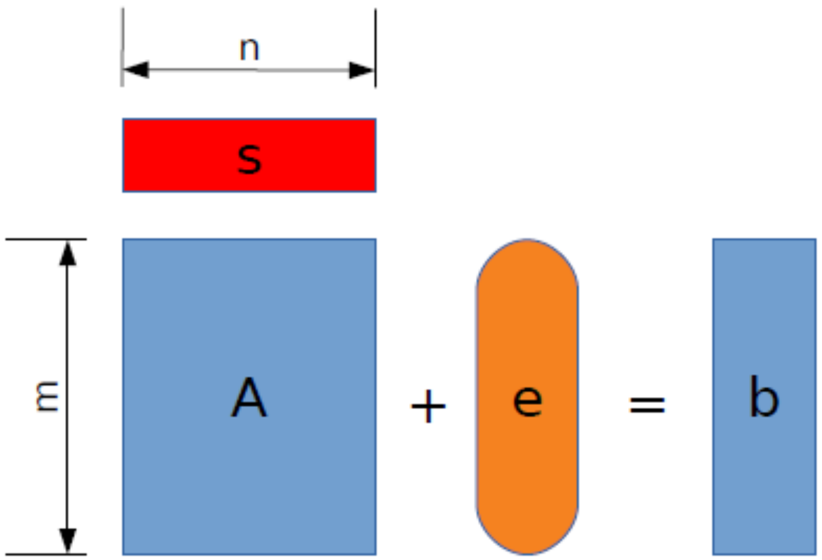
- LWE_A is one-way: Hard to recover (s,e) from [A,b]
- $b=LWE_A(s,e)$ is indistinguishable from uniform over $Z_q[m]$

Regev: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography

Symmetric key Encrypting with LWE

- Idea: Use b=As+e as a one-time pad

  - secret key: $s \in Z_q^n$,
  - message: $m \in Z^m$
  - encryption randomness: [A,e]
  - $E_s(m; [A,e]) = [A, b+m]$

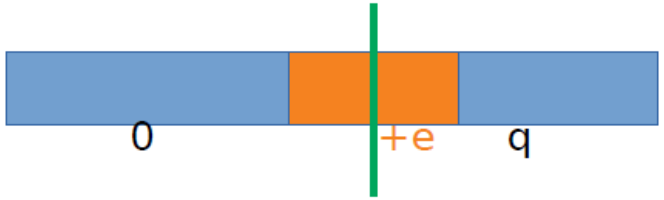


Decryption with noise

$E_s(m;[A,e]) = [A,b+m]$ where $b = As+e$

Decryption:

- $D_s([A,b+m]) = (b+m) - As = m+e \bmod q$



- Low order bits of m are corrupted by e

Only use the highest bits

Operations on LWE Ciphertexts

$$[A_1, A_1 s + e_1 + m_1] \; + [A_2, A_2 s + e_2 + m_2]$$

$$= [(A_1 + A_2), (A_1 + A_2)s + (e_1 + e_2) + \; (m_1 + m_2)]$$

Add: $E(m_1;\beta_1) + E(m_2;\beta_2) \subset E(m_1 + m_2;\beta_1 + \beta_2)$

Neg: $-E(m;\beta) = E(-m;\beta)$

Mul: $c{*}E(m;\beta) = E(c{*}m;\ c{*}\beta)$

Noise increasing

So the order the operations is limited

Somewhat HE (SWHE)

A Fully Homomorphic Encryption Scheme

Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

ABSTRACT

We propose a fully homomorphic encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. Our solution comes in three steps. First, we provide a general result – that, to construct an encryption scheme that permits evaluation of *arbitrary circuits*, it suffices to construct an encryption scheme that can evaluate (slightly augmented versions of) its *own decryption circuit*; we call a scheme that can evaluate its (augmented) decryption circuit *bootstrappable*.

Next, we describe a public key encryption scheme using *ideal lattices* that is *almost* bootstrappable. Lattice-based cryptosystems typically have decryption algorithms with low circuit complexity, often dominated by an inner product computation that is in NC1. Also, *ideal* lattices provide both additive and *multiplicative* homomorphisms (modulo a public-key ideal in a polynomial ring that is represented as a lattice), as needed to evaluate general circuits.

Unfortunately, our initial scheme is not quite bootstrappable – i.e., the depth that the scheme can correctly evaluate can be logarithmic in the lattice dimension, just like the depth of the decryption circuit, but the latter is greater than the former. In the final step, we show how to modify the scheme to reduce the depth of the decryption circuit, and thereby obtain a bootstrappable encryption scheme, without reducing the depth that the scheme can evaluate. Abstractly, we accomplish this by enabling the *encrypter* to start the decryption process, leaving less work for the decrypter, much like the server leaves less work for the decrypter in a server-aided cryptosystem.

Categories and Subject Descriptors: E.3 [Data Encryption]: Public key cryptosystems

General Terms: Algorithms, Design, Security, Theory

1. INTRODUCTION

We propose a solution to the old open problem of constructing a *fully homomorphic encryption scheme.* This notion, originally called a *privacy homomorphism*, was introduced by Rivest, Adleman and Dertouzos [54] shortly after the invention of RSA by Rivest, Adleman and Shamir [55]. Basic RSA is a multiplicatively homomorphic encryption scheme – i.e., given RSA public key $\mathrm{pk} = (N, e)$ and ciphertexts $\{\psi_i \leftarrow \pi_i^e \bmod N\}$, one can efficiently compute $\prod_i \psi_i = (\prod_i \pi_i)^e \bmod N$, a ciphertext that encrypts the product of the original plaintexts. Rivest et al. [54] asked a natural question: What can one do with an encryption scheme that is *fully* homomorphic: a scheme $\mathcal{E}$ with an efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that, for any valid public key pk, *any* circuit C (not just a circuit consisting of multiplication gates), and any ciphertexts $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, outputs

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t) ,$$

a valid encryption of $C(\pi_1, \ldots, \pi_t)$ under pk? Their answer: one can arbitrarily *compute on encrypted data* – i.e., one can process encrypted data (query it, write into it, do anything to it that can be efficiently expressed as a circuit) without the decryption key. As an application, they suggested private data banks: a user can store its data on an untrusted server in encrypted form, yet still allow the server to process, and respond to, the user's data queries (with responses more concise than the trivial solution: the server just sends all of the encrypted data back to the user to process). Since then, cryptographers have accumulated a list of "killer" applications for fully homomorphic encryption. However, prior to this proposal, *we did not have a viable construction.*

1.1 Homomorphic Encryption

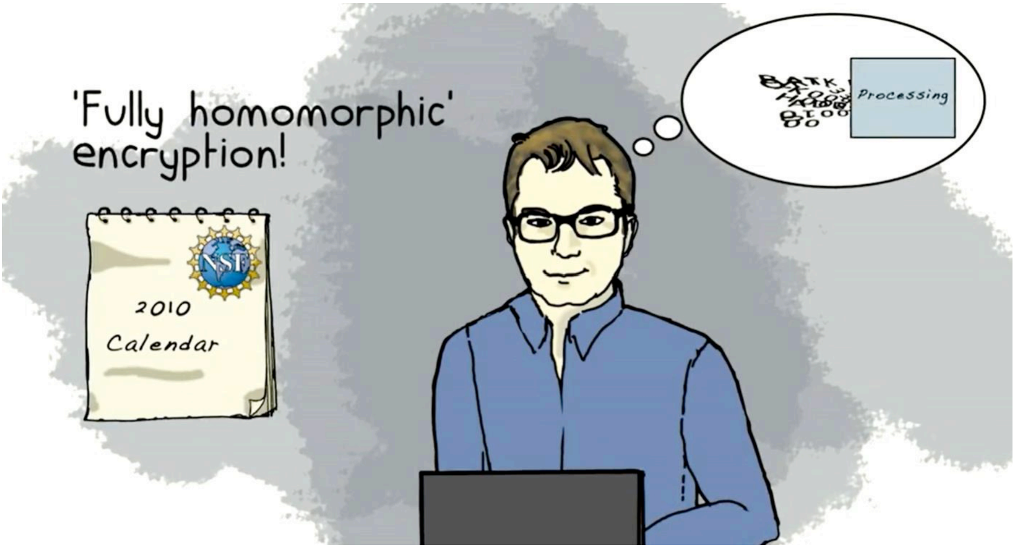
A homomorphic public key encryption scheme $\mathcal{E}$ has four algorithms $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Decrypt}_{\mathcal{E}}$, and an additional algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that takes as input the public key pk, a circuit C from a permitted set $\mathcal{C}_{\mathcal{E}}$ of circuits, and a tuple of ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$; it outputs a ciphertext ψ. The computational complexity of all of these algorithms must be polynomial in security parameter λ and (in the case of $\mathsf{Evaluate}_{\mathcal{E}}$) the size of C. $\mathcal{E}$ is *correct* for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair (sk, pk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, any plaintexts $\pi_1, \ldots, \pi_t$, and any ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ with $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, it is the case that:

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi) \Rightarrow C(\pi_1, \ldots, \pi_t) = \mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$$

By itself, mere correctness does not exclude trivial schemes.[1] So, we require ciphertext size and decryption time to be up-

[1] In particular, we could define $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi)$ to just output (C, Ψ) without "processing" the circuit or ciphertexts at all, and $\mathsf{Decrypt}_{\mathcal{E}}$ to decrypt the component ciphertexts and apply C to results.

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.



2009: Gentry,
"Fully Homomorphic Encryption Using Ideal Lattices"

What if a homomorphic encryption scheme can decrypt itself with an encrypted key

Bootstrapping

- SWHE + Bootstrapping → (leveled) FHE

- Assume $c = Enc_{s_2}(s_1, e_2)$

Keep in mind: e_2 is small and e_1 is large

- After several HE operations (Add, Mul, etc.)
- Assume $c_1 = Enc_{s_1}(m, e_1)$
- is the encryption of m under secret key s_1 with very large noise e_1

- Denote function f_{c_1}: $s_1 \to Dec_{s_1}(c_1)$

Bootstrapping

Keep in mind: e_2 is small and e_1 is large

- $c = Enc_{s_2}(s_1, e_2)$ $\quad\quad c_1 = Enc_{s_1}(m, e_1)$

Assume SWHE supports Eval *Dec*

- Apply f_{c_1} to $c = Enc_{s_2}(s_1, e_2)$, we get

$\boxed{f_{c_1}: s_1 \to Dec_{s_1}(c_1)}$

- $c_2 = Enc_{s_2}(f_{c_1}(s_1), e_2)$
$= Enc_{s_2}(Dec_{s_1}(c_1), e_2)$
$= Enc_{s_2}(m, e_2)$

Bootstrapping: "refreshes" a ciphertext by running the decryption function on it homomorphically, resulting in a reduced noise.

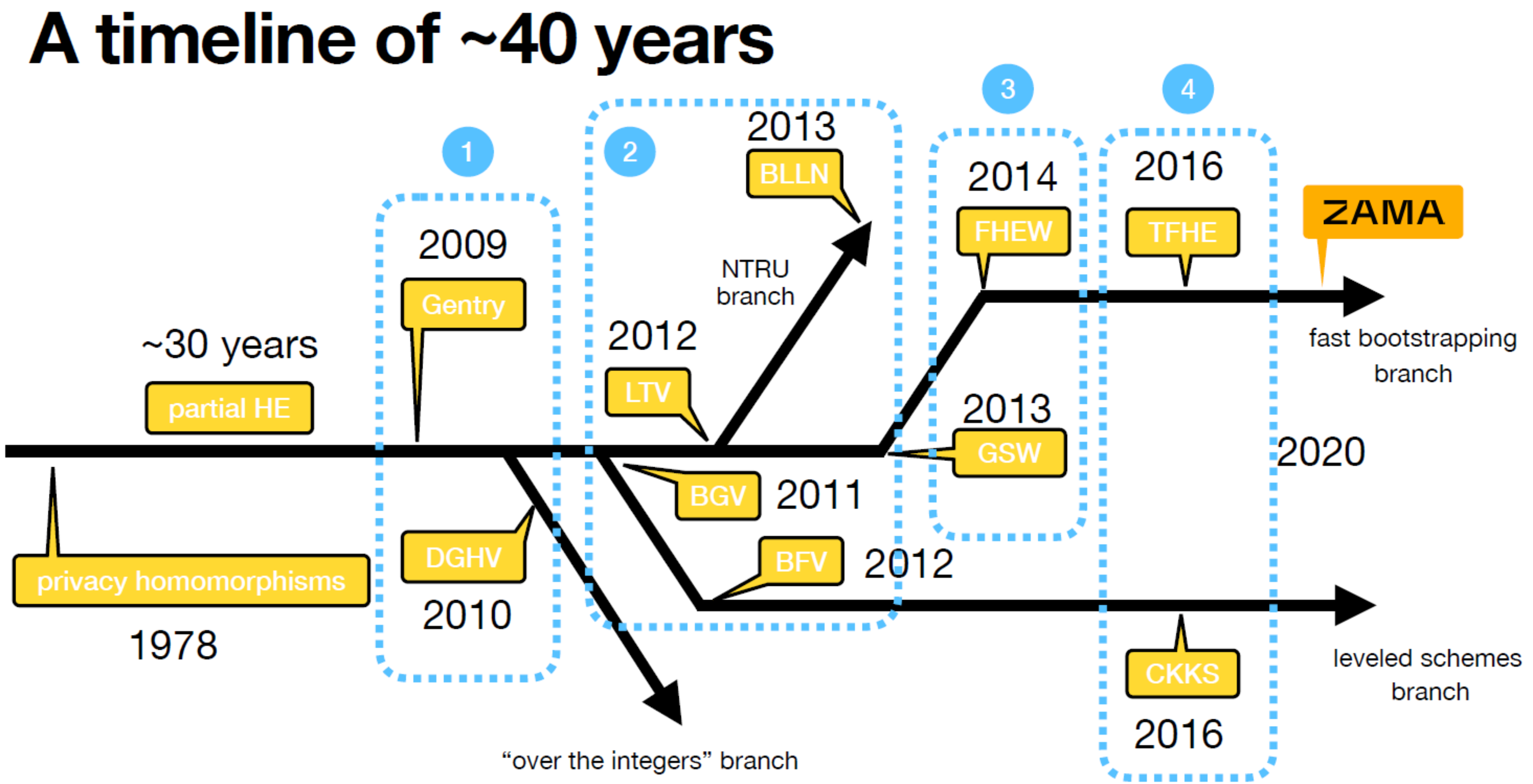
$$c_1 \text{ under key } s_1 \text{ with noise } e_1 \xrightarrow{refresh} c_2 \text{ under key } s_2 \text{ with noise } e_2$$
$$\mathrm{Dec}(c_1, s_1) = \mathrm{Dec}(c_2, s_2) \text{ and } |e_2| < |e_1|$$

Focus on SWHE

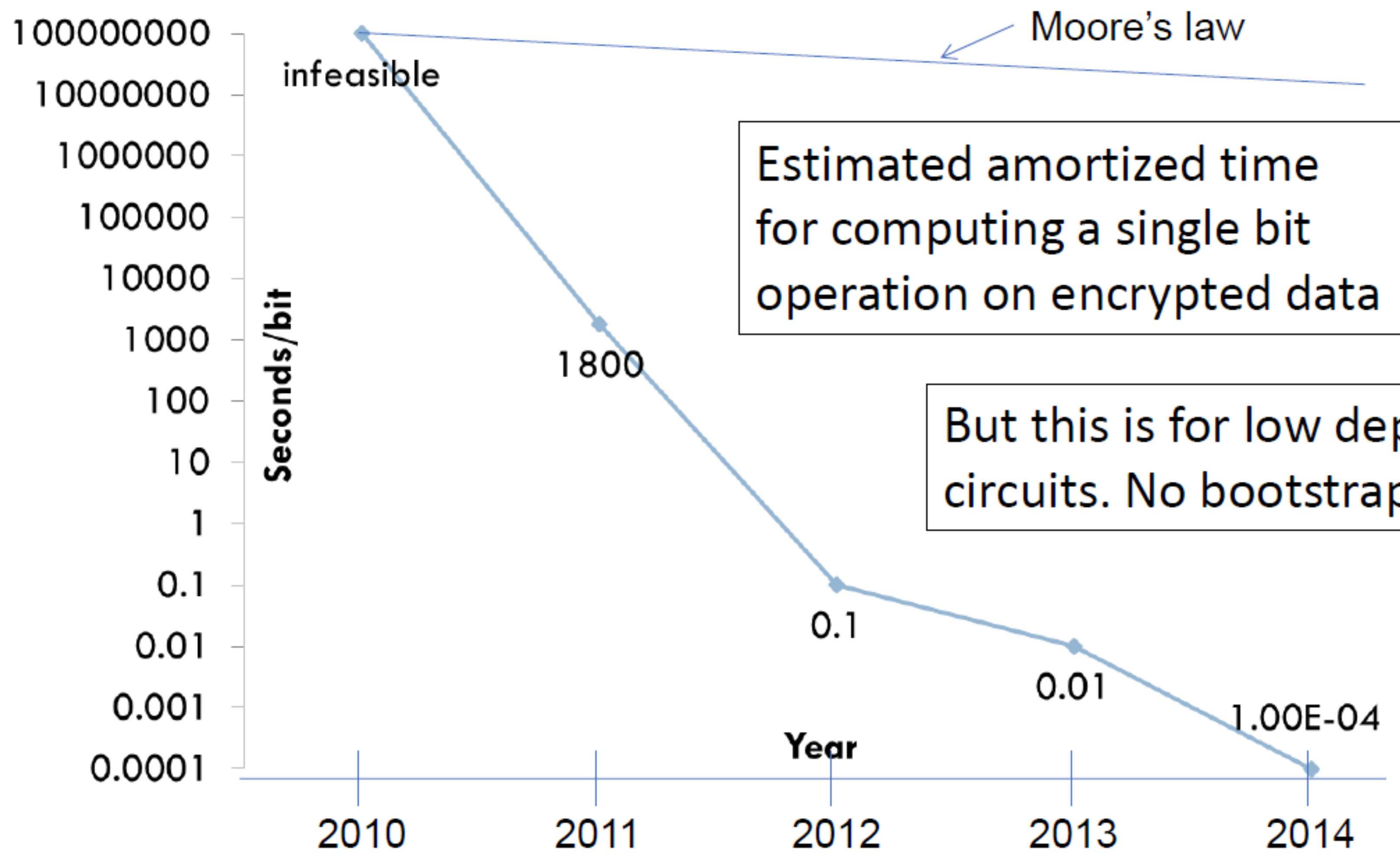
So, we only need to focus on the construction of SWHE which supports the Eval operation of Dec

The scheme proposed by Gentry is rather impractical

1-4 generations of FHE



First 2 Generations of FHE



Estimated amortized time for computing a single bit operation on encrypted data

But this is for low depth (e.g., 10) circuits. No bootstrapping here!

4^{th} Generation FHE (CKKS Scheme)

Homomorphic Encryption for Arithmetic of Approximate Numbers

Jung Hee Cheon[1], Andrey Kim[1], Miran Kim[2], and Yongsoo Song[1]

[1] Seoul National University, Republic of Korea
{jhcheon, kimandrik, lucius05}@snu.ac.kr
[2] University of California, San Diego
mrkim@ucsd.edu

Abstract. We suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new *rescaling* procedure for managing the magnitude of plaintext. This procedure truncates a ciphertext into a smaller modulus, which leads to rounding of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. As a result, our decryption structure outputs an approximate value of plaintext with a predetermined precision.
We also propose a new batching technique for a RLWE-based construction. A plaintext polynomial is an element of a cyclotomic ring of characteristic zero and it is mapped to a message vector of complex numbers via complex canonical embedding map, which is an isometric ring homomorphism. This transformation does not blow up the size of errors, therefore enables us to preserve the precision of plaintext after encoding.

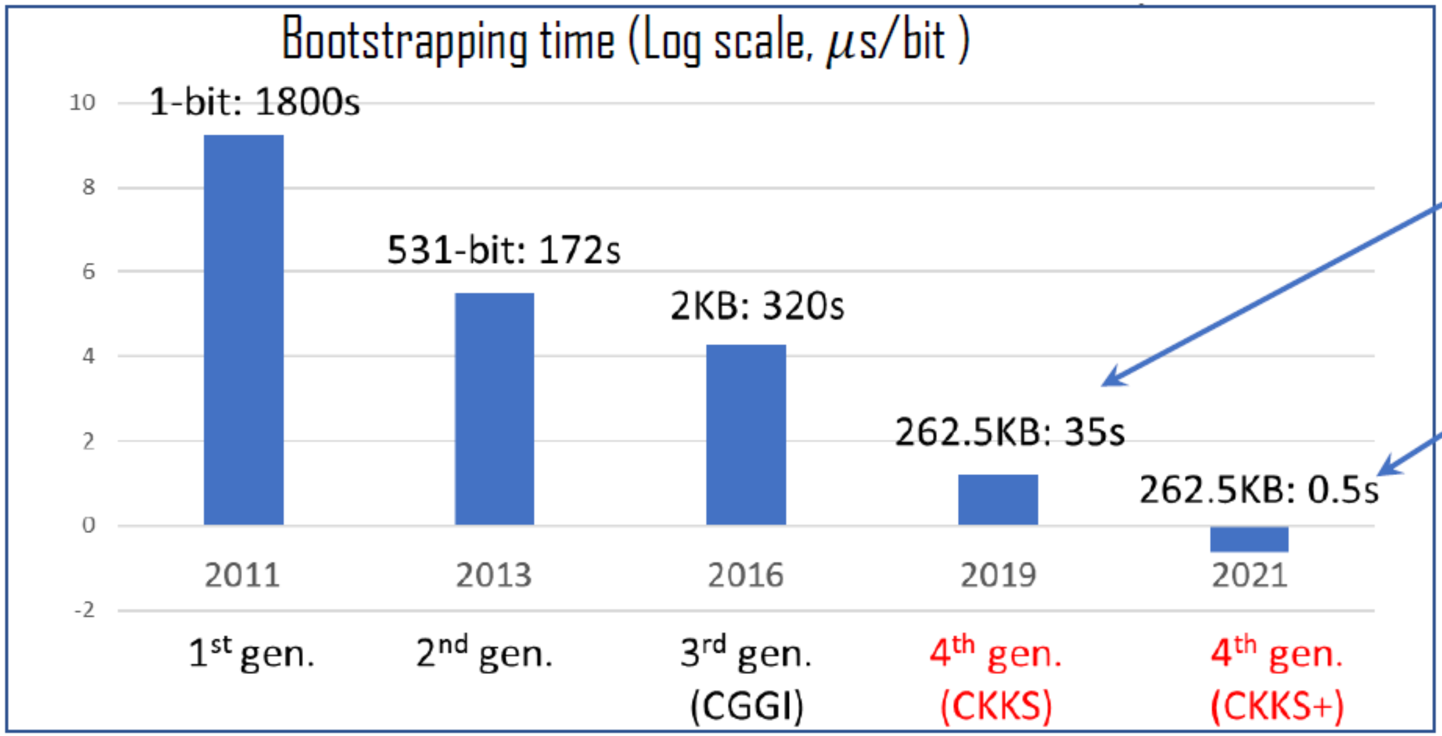
1^{st} and 2^{nd} Gens: mod-p numbers, arithmetic circuits

3^{rd} Gen: bits, boolean circuits

4^{th} Gen: real (or complex) numbers, approximate (floating pt) arithmetic

Works great in apps that use floating point, like neural networks

HE is getting faster 8 times every year



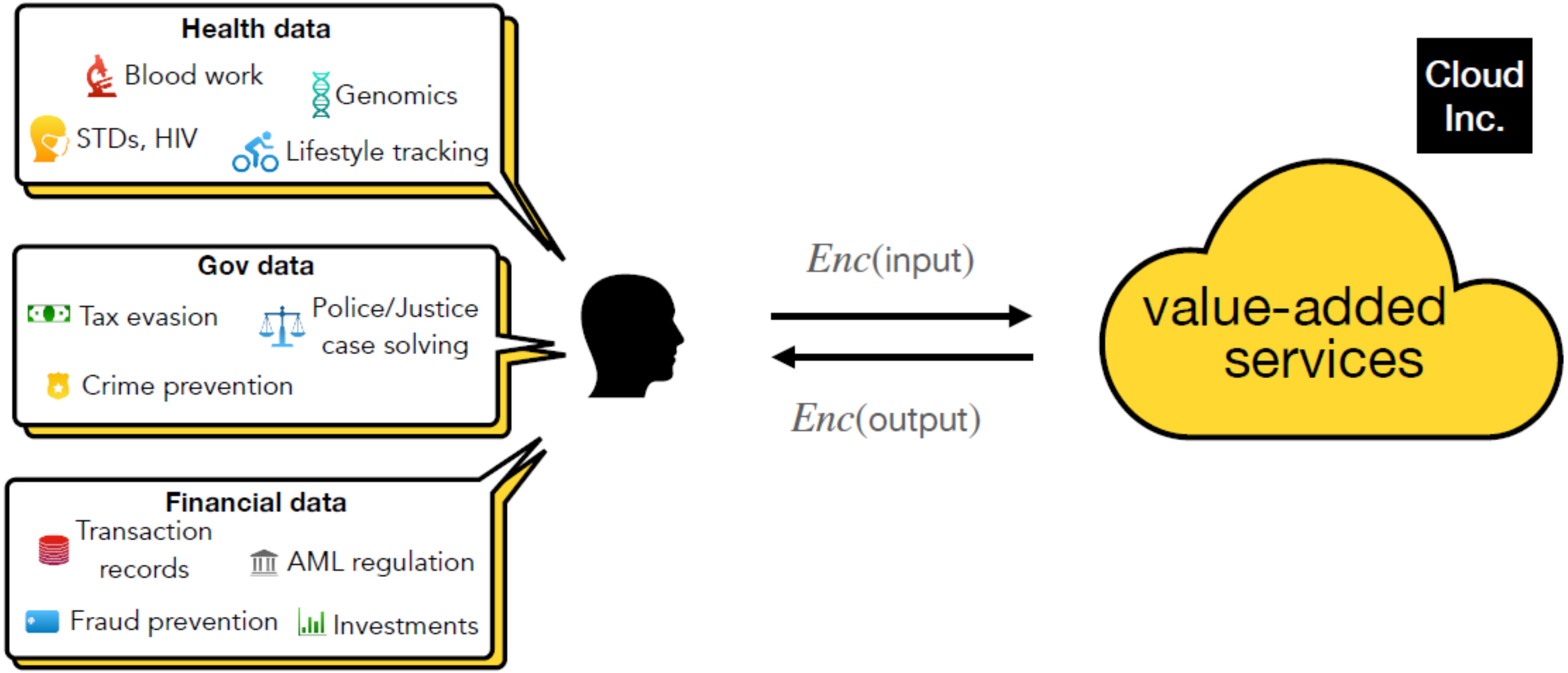
$19\mu s/\text{bit}$ bootstrapping time! (amortized)

$0.29\mu s/\text{bit}$ bootstrapping time! (amortized)

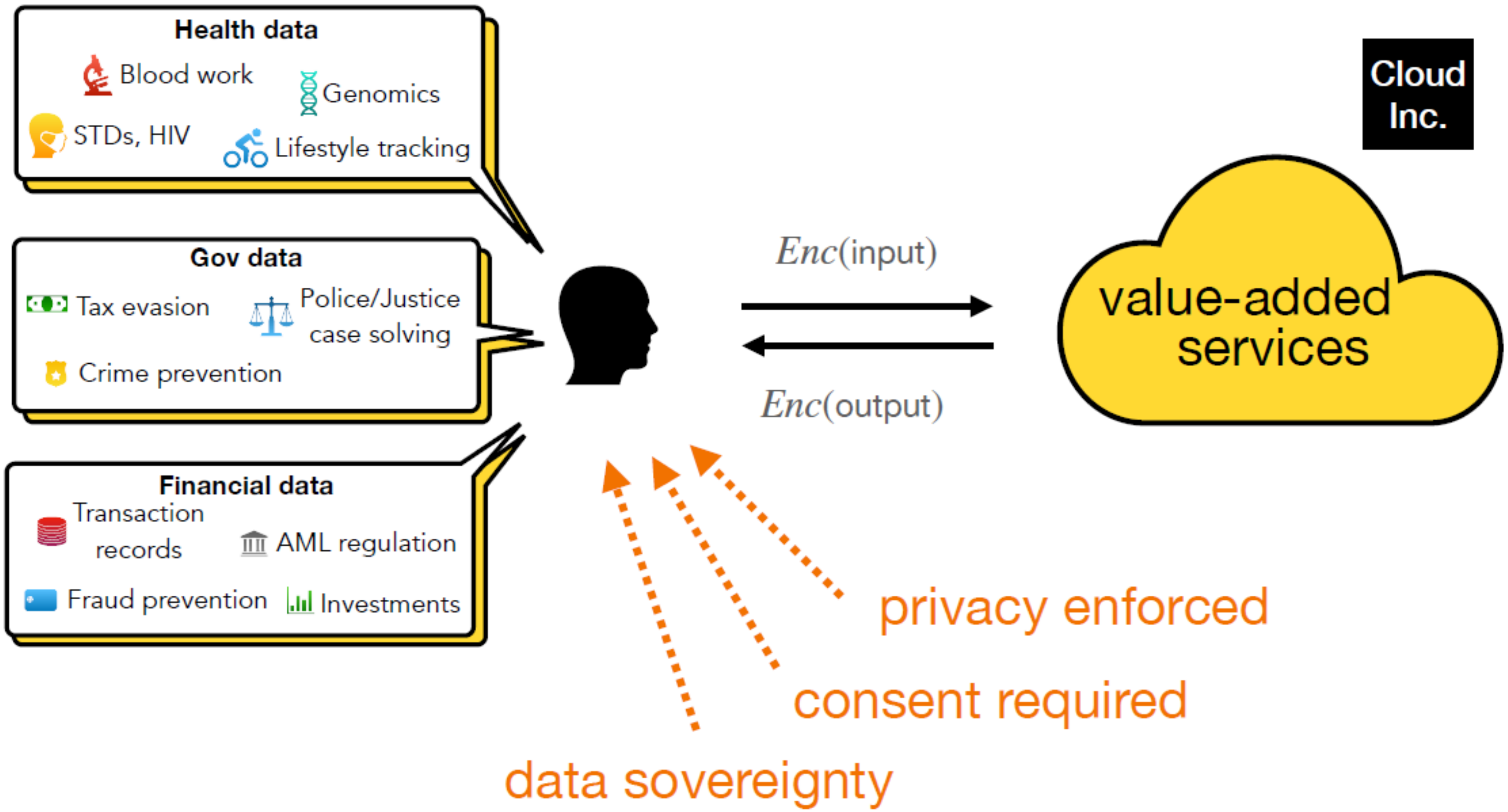
서울대학교 SEOUL NATIONAL UNIVERSITY CRYPTO LAB

CKKS: CPU-based, CKKS+: GPU-based

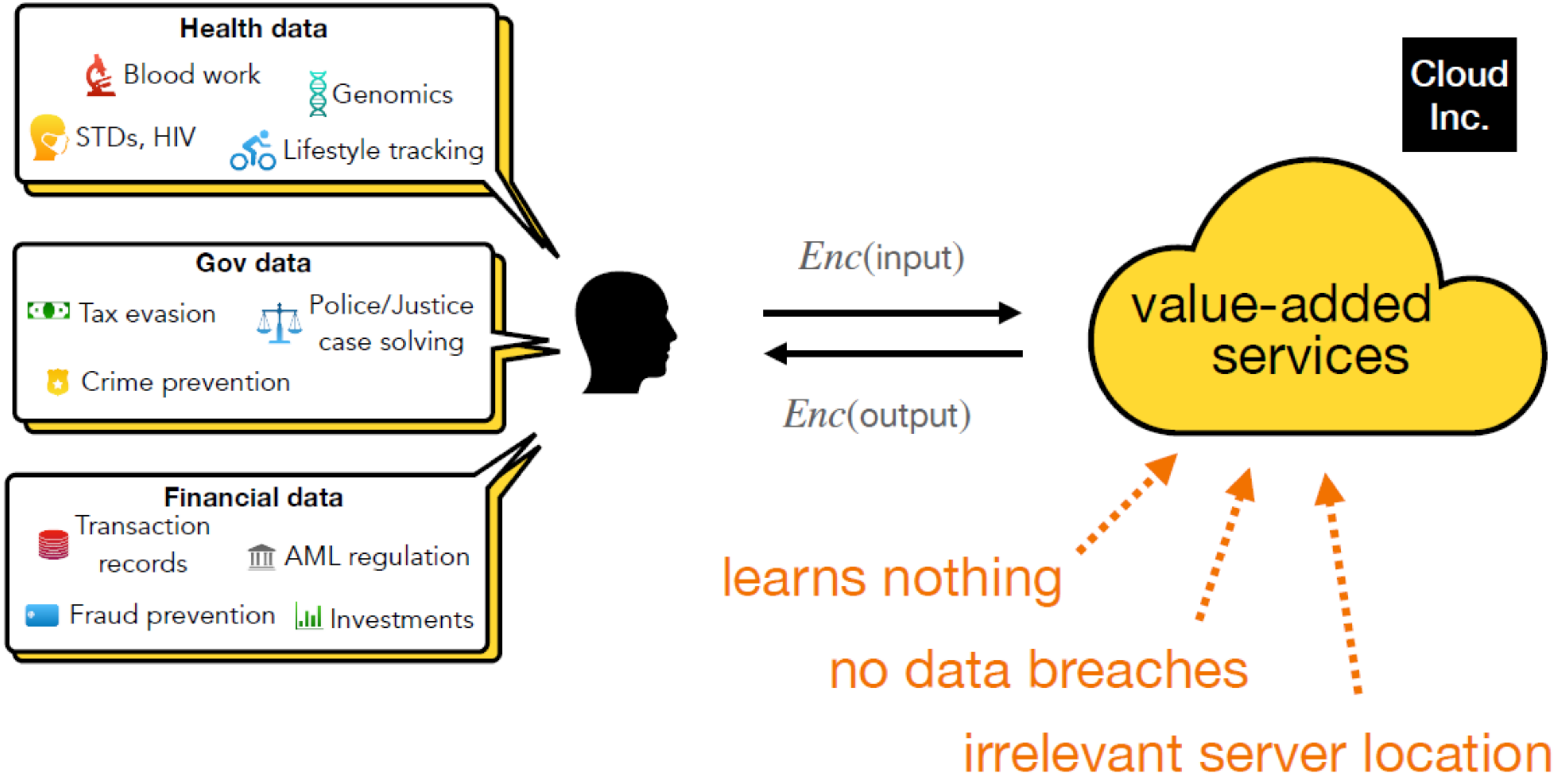
FHE- Game changer



FHE- Game changer



FHE- Game changer



Other applications

- **machine learning**

- **etc,.**

Summary

- We could build FHE from somewhat HE

- Further from lattice-based cryptography.

- FHE has a lot of applications

Thank you