Public Key Cryptography

From RSA and EC to Post-Quantum

Frederic Detienne
Distinguished Engineer

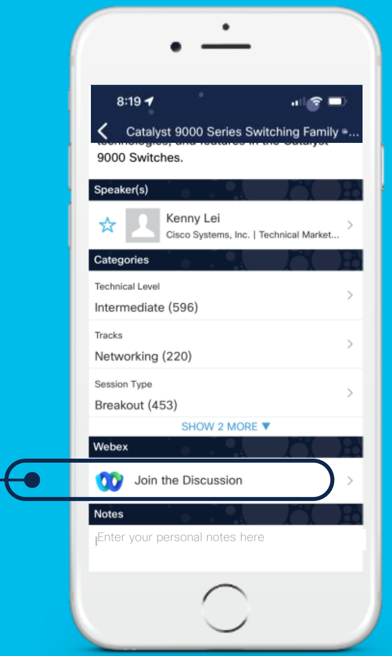
Cisco Webex App

Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click "Join the Discussion"
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

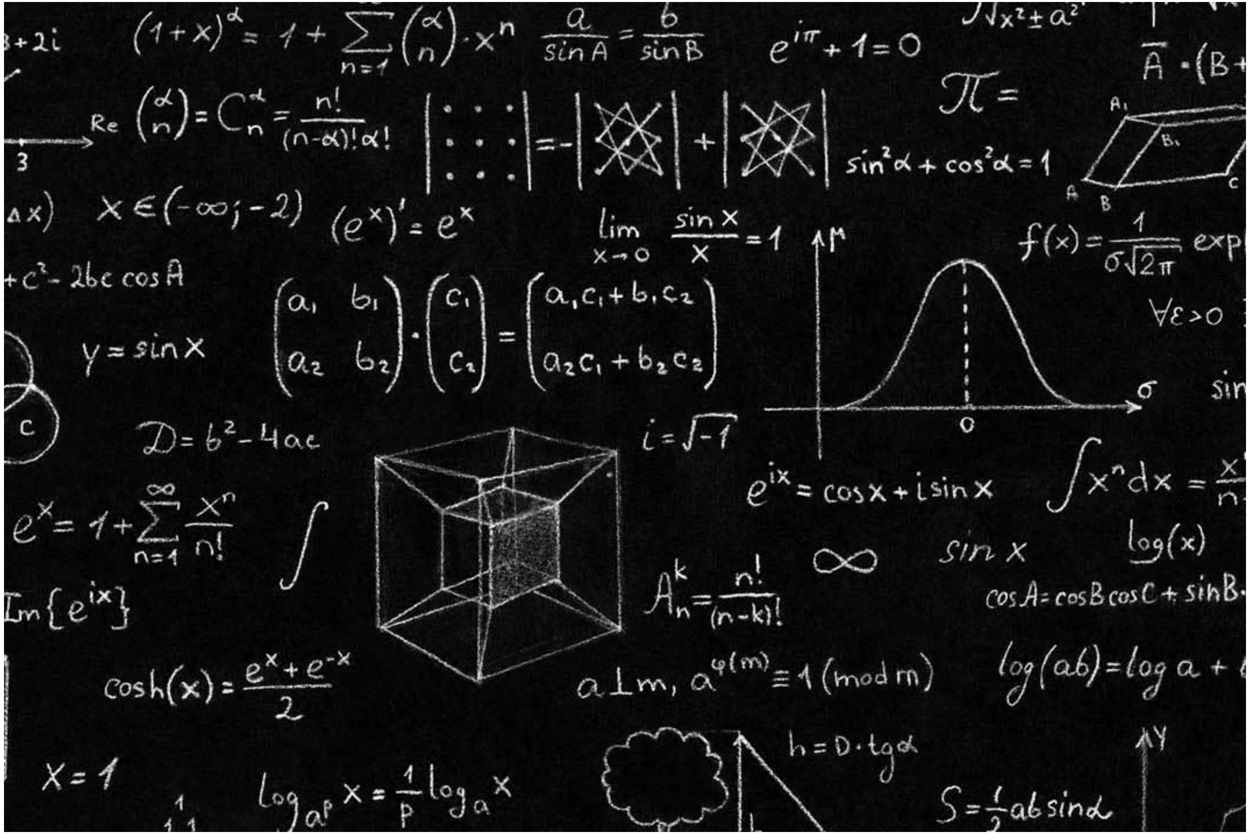


Webex spaces will be moderated until February 24, 2023.

Agenda

- A Brief Introduction
- MODP: Multiplicative Group of Integers Modulo P
- ECC: Elliptic Curve Cryptography
- Enters The Quantum Computer
- Lattice Based Cryptography
- Conclusion and Recommendations

Today is a bout making math fun!



Introduction

Cryptographic Mechanisms


Encryption


Signatures


Data Authentication (HMAC)


Random Number Generation


Key Establishment

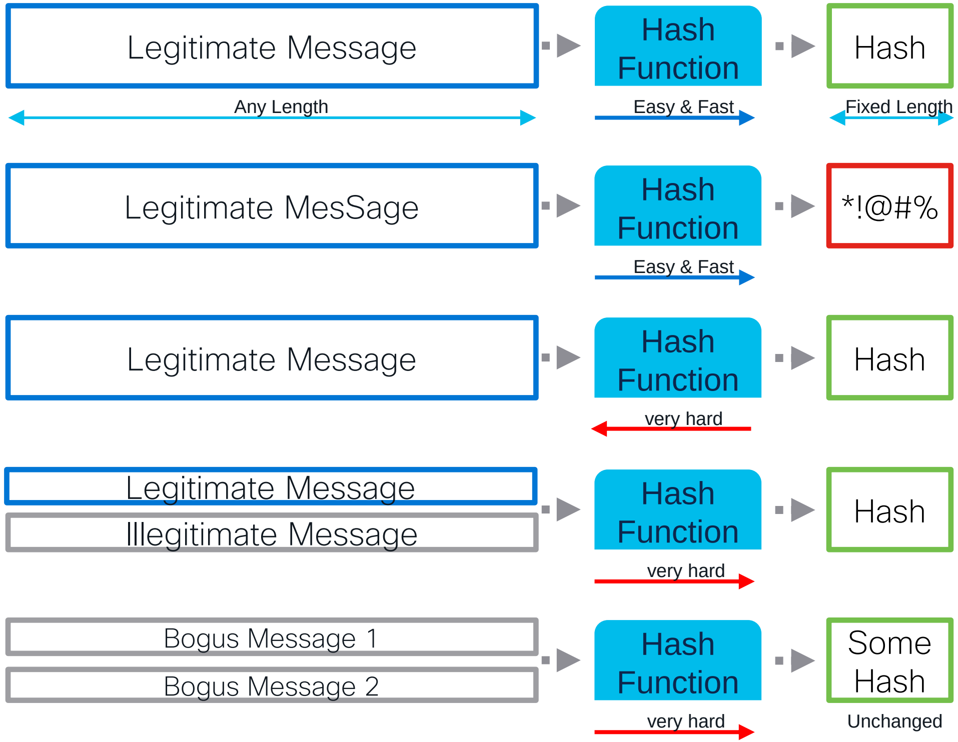

Hashing

Today - Suite B

Authenticated Encryption	AES-GCM
Authentication	HMAC-SHA-2
Key Establishment	ECDH
Digital Signatures	ECDSA
Hashing	SHA-2
Entropy	SP800-90
Protocols	TLSv1.2, IKEv2, IPsec, MACSec

Hashes and HMAC’s Focus on SHA-2

What is a Cryptographic Hash Function



Fixed length output

Avalanche effect

(small change in message, big change in hash)

Pre-image resistance

(message can not be found from hash)

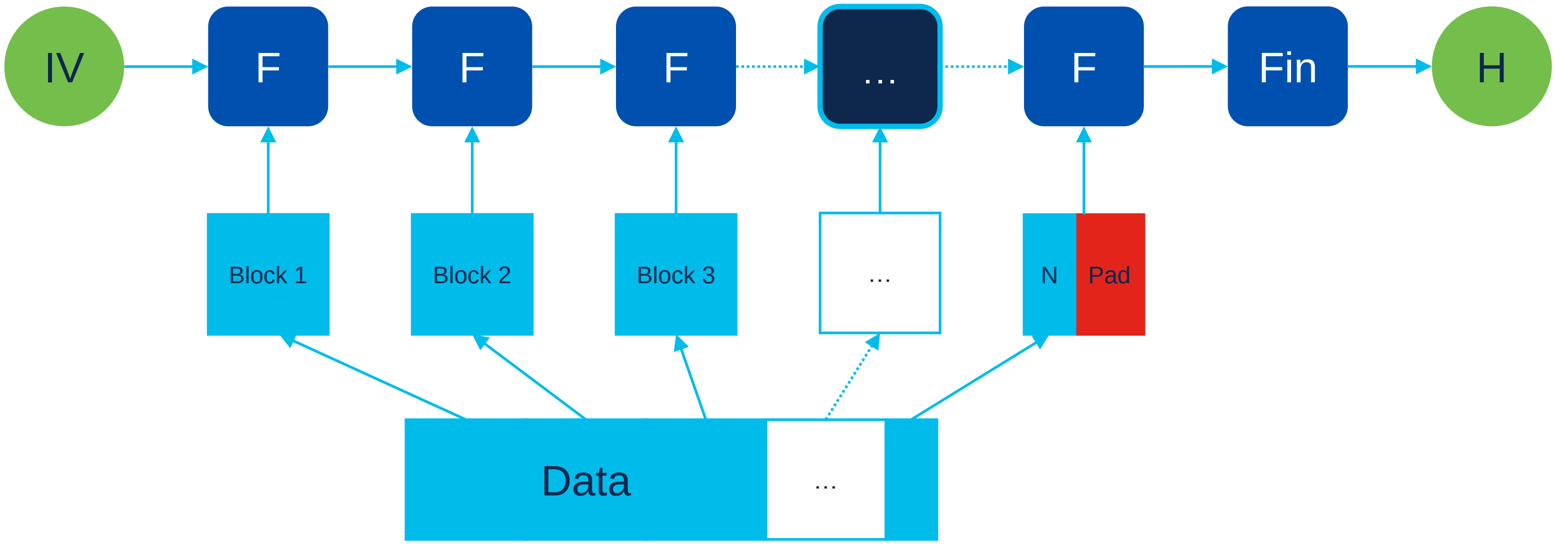
Second pre-image resistance

(legitimate message and hash are imposed; find new message)

Collision resistance

(attacker gets to select message 1 and 2 ; hash must match)

The Merkle–Damgård Construction



Symmetric Encryption Algorithms: One Time Pad & AES

One Time Pad

- A Pad is a truly random sequence of numbers
- Pad is used as encryption and decryption key through modular addition
- The Pad must be as long as the message
- The Pad must be used ONLY ONCE
- If used properly, this is the strongest possible encryption scheme

M	1	0	0	1	1	0	1	1	1	…
Pad	0	1	1	0	0	0	1	0	1	…
Cypher	1	1	1	1	1	0	0	1	0	…

A One Time Pad (here using XOR)

One Time Pad - example

	H	E	L	L	O	message
	7	4	11	11	14	
+	23	12	2	10	11	key
=	30	16	13	21	25	m + k
mod 26	4	16	13	21	25	(m+k) mod 26
	E	Q	N	V	Z	ciphertext

	E	Q	N	V	Z	ciphertext
	4	16	13	21	25	
-	23	12	2	10	11	key
=	-19	4	11	11	14	c - k
mod 26	7	4	11	11	14	(c-k) mod 26
	H	E	L	L	O	message

Issue 1 – Key Length

	H	E	L	L	O	message
	7	4	11	11	14	
+	23	12	2	10	11	key
=	30	16	13	21	25	m + k
mod 26	4	16	13	21	25	(m+k) mod 26
	E	Q	N	V	Z	ciphertext

Key must have the same size as message… Key exchange is a problem!

Use high quality Deterministic Random Bit Generator (DRBG)

Select Carefully… ☺

Issue 2 – Key Re-use & Known Plain Text Attack

	H	E	L	L	O	message
	7	4	11	11	14	
+	23	12	2	10	11	key
=	30	16	13	21	25	m + k
mod 26	4	16	13	21	25	(m+k) mod 26
	E	Q	N	V	Z	ciphertext

Assumption #1: Attacker knows some plain text (e.g. injection, guess,…)

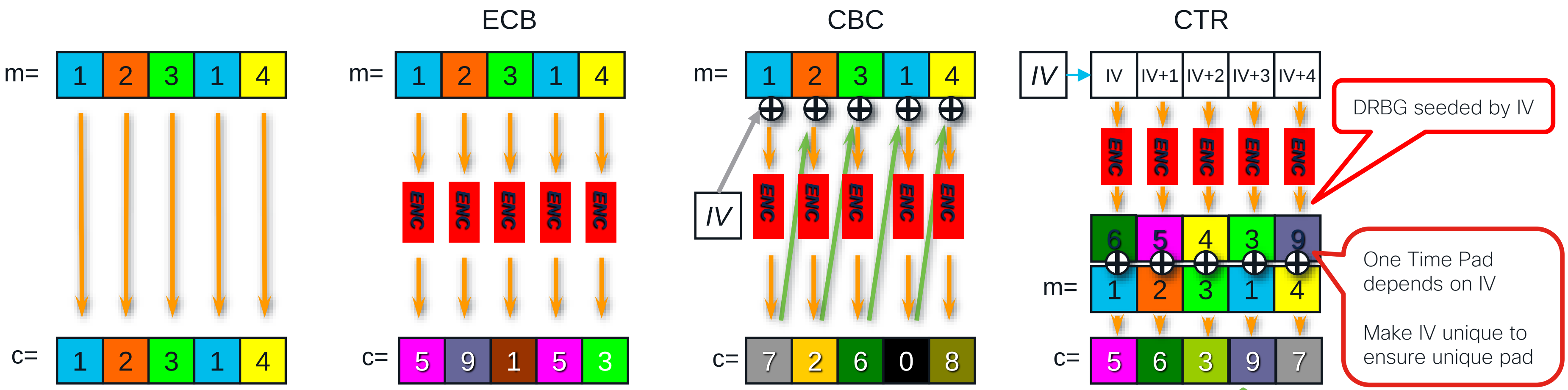
Assumption #2: Attacker can wiretap ciphertext

	H	E	L	L	O	known message
	4	16	13	21	25	ciphertext
-	7	4	11	11	14	known message
=	-3	12	2	10	11	c - m
mod 26	23	12	2	10	11	(c - m) mod 26
=	KEY					

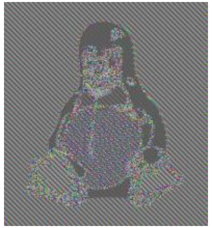
Conclusion: Attacker can compute the key easily

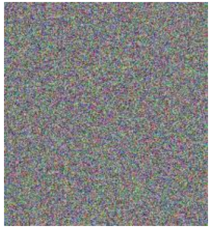
➔ DO NOT REUSE KEY !!

Block Cipher Mode of Operation (ECB, CBC, counter)



Parallel encryption pipelines.
Efficient implementation possible





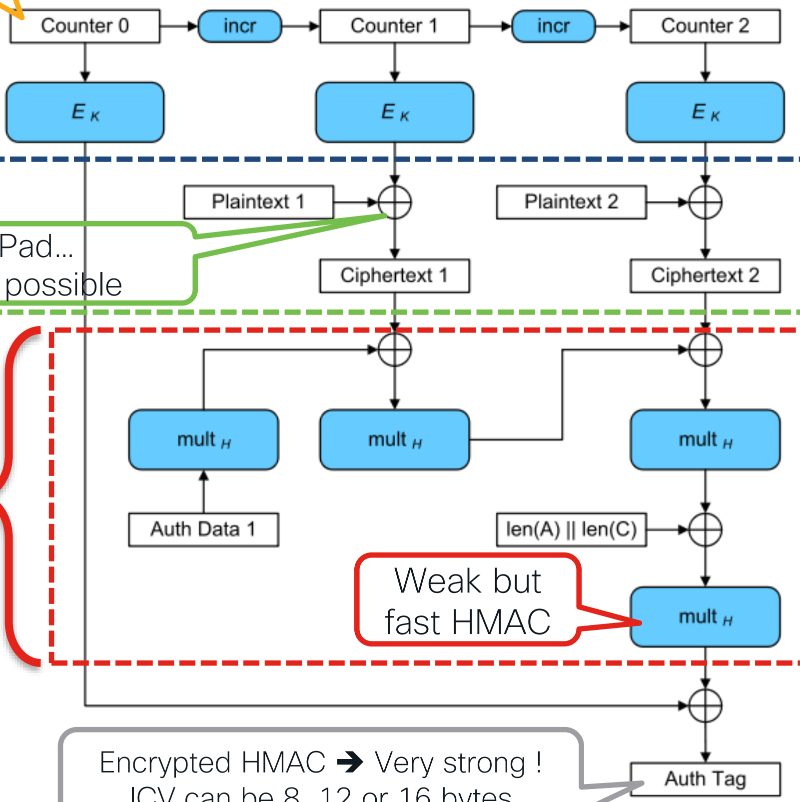
Penguin source: Wikipedia

AES GCM

Fed from Initialization Vector

One Time Pad Algorithm

AES Based PRNG generate pad… Secure CTR DRBG



One Time Pad… Parallelization possible

Galois HMAC

Weak but fast HMAC

Encrypted HMAC ➔ Very strong ! ICV can be 8, 12 or 16 bytes

$GF(2^{128})$
Polynomial $x^{128}+x^7+x^2+x+1$
$GHASH(H, A, C) = X_{m+n+1}$
u,v bits in A_m, P_n

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{128-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* \| 0^{128-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\text{len}(A) \| \text{len}(C))) \cdot H & \text{for } i = m+n+1 \end{cases}$$

AES GCM in summary
- AES is more secure than 3DES
- AES-CTR CAN be much faster (implementation…)
- GMAC consumes less than SHA-2 (or even SHA-1)

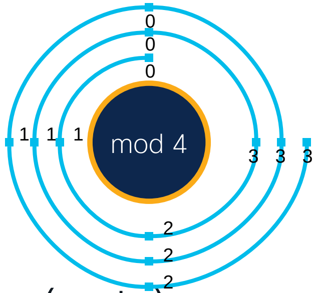
MODP Multiplicative Group of Integers Modulo P

RSA

- Rivest, Shamir, Adleman (1977)
  - Patented but expired => no more royalty
- Public key cryptosystem
- Variable key length (usually 512-2048 bits)
- Based on the (current) difficulty of factoring very large numbers

Modular Arithmetic

- Modulo is like a clock



- $b^x \bmod n = r$ also written as $b^x \equiv r \pmod{n}$
  - b is the base
  - x is the exponent
  - n is the modulus
  - r is the remainder
- Knowing b, x & n, it is very easy to compute r
- Knowing x, r & n, it is very difficult to compute $b = \sqrt[x]{r} \bmod n$ aka the RSA problem
- Knowing b, r & n, it is very difficult to compute $x = \log_b(r) \bmod n$ aka the discrete log problem

unless there are trapdoors

Encryption with Modular Arithmetic

Alice

Must send a private message **m**

Takes **n** & **e** from Bob
(we assume m < n)
Computes **c = m^e mod n**

c

Attacker can not guess **m** just knowing **c, n and e**

To decrypt, the attacker would need to compute $m = \sqrt[d]{m'} \bmod n$
➔ RSA Problem

Bob

Selects three numbers **n, d & e**
n & **e** are **public**, **d** is **secret**
e, d are chosen such as $ed \equiv 1 \bmod n$

Computes **$m' = c^d$ mod n**

$$
\begin{aligned}
m' &= c^d \bmod n \\
&= (m^e)^d \bmod n \\
&= m^{ed} \bmod n \\
&= m^1 \bmod n \\
&= m
\end{aligned}
$$

➔ Bob has reversed the operation !!
➔ Bob knows d but nobody else…
➔ We have an encryption scheme

Signature with Modular Arithmetic

Alice

Takes **n** & **e** from Bob

Computes **m' = c^e mod n**

$m' = c^e \bmod n$
$= (m^d)^e \bmod n$
$= m^{de} \bmod n$
$= m^1 \bmod n$
$= m \bmod n$
$= m$

➔ Bob must have sent the **c,m**

Bob

Selects three numbers **n, d & e**
n & **e** are **public**, **d** is **secret**
e, d are chosen such as $ed \equiv 1 \bmod n$

Must send a signed message **m**

Computes **c** = $m^d \bmod n$
(we assume $m < n$)

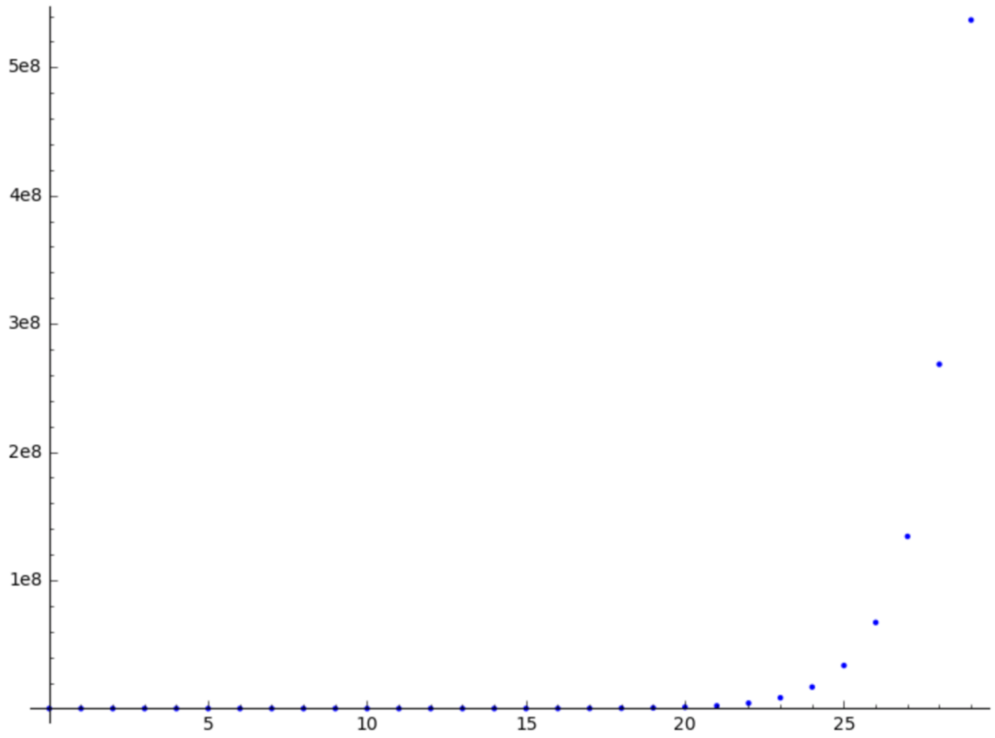
Attacker can not guess **d** just knowing **m, n and e**

c, m

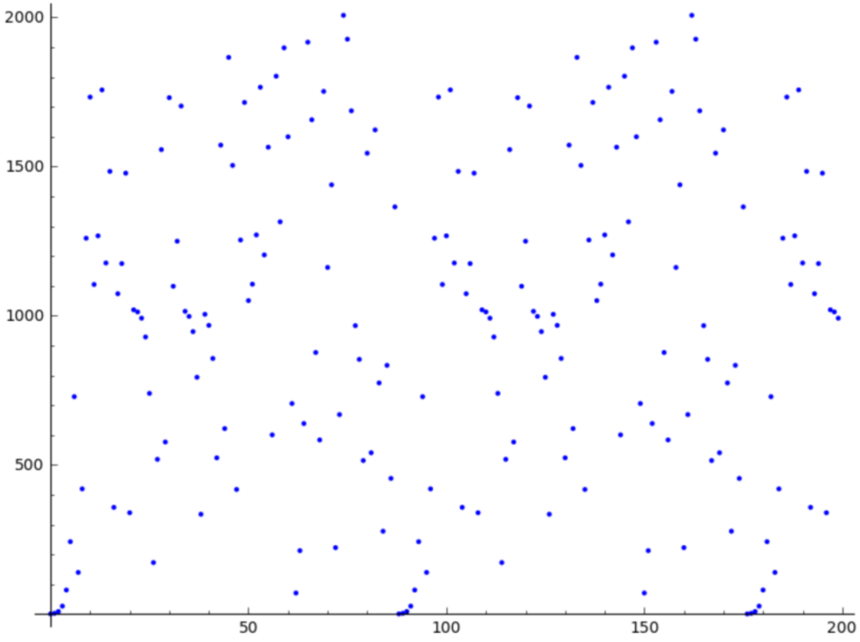
To forge the signature, the attacker would need to compute
$d = \log_e(m') \bmod n$
➔ Discrete Logarithm Problem

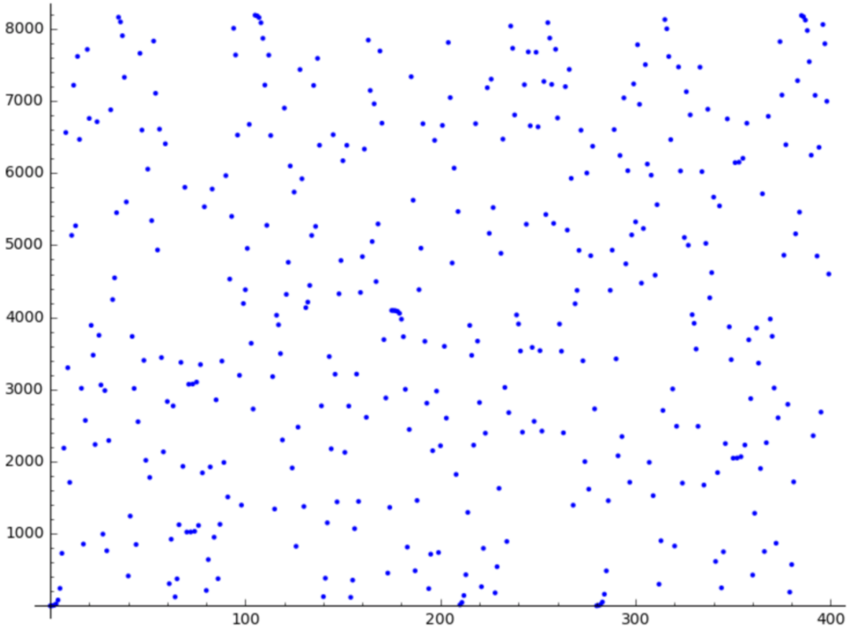
Now how can we find such e, d and n ?

Regular Exponentiation –Dichotomy to reverse



MODP breaks dichotomy





RSA keys – finding e,d,n | $m^{ed} \equiv m \pmod{n}$

- Choose two distinct prime numbers *p, q* and hide them forever!
- $n = p.q \rightarrow$ n is hard to factor if p & q are very large
- $\varphi(n) = n-(p+q-1)$
  - p & q are prime ➔ $\varphi(p)=p-1$ $\varphi(q)=q-1$
  - $\varphi(n) = \varphi(pq) = \varphi(p)\,\varphi(q) = (p-1)(q-1) = n-(p+q-1)$
- Final steps

  Euler theorem…
  - $1^k = 1$ ➔ $(m^{\varphi(n)})^k \equiv 1^k \pmod{n}$ ➔ $m^{k\varphi(n)} \equiv 1 \pmod{n}$
  - $1m = m$ ➔ $m\, m^{k\varphi(n)} \equiv m \pmod{n}$ ➔ $m^{k\varphi(n)+1} \equiv m \pmod{n}$
  - we look for e,d,n such that $m^{ed} \equiv m^{k\varphi(n)+1} \equiv m \pmod{n}$ ➔ $ed = k\,\varphi(n) + 1$
  - ➔ $d = \dfrac{k\,\varphi(n)+1}{e} = \dfrac{k\,(n-(p+q-1))+1}{e}$
- Select **e**, small integer and **k** such that $GCD(d, \varphi(n)) = 1$ (i.e. d & $\varphi(n)$ are co-prime)
  - **e** is usually 3 or 65537
  - adjust **k** to make **d** an integer

m – arbitrary message
n – the modulus
e – the public key
d – the private key

DH –Diffie-Hellman

Alice

Select a generator **g** and a modulus **p**
Pick a random number **a**
Keep **a** secret!!
Compute A_{pub} = g^a mod p

The group definition

A_{pub}, (g, p)

Attacker can not guess **a**

Attacker can not guess **b**

B_{pub}

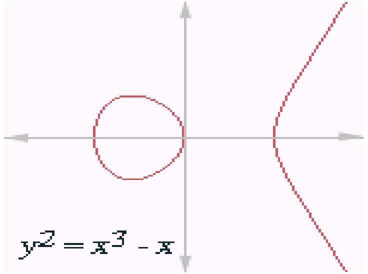
Bob

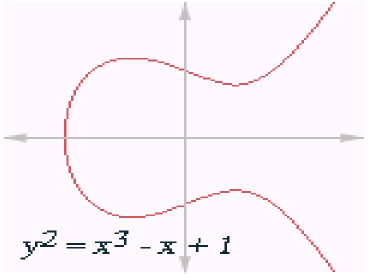
Using the same **g** and **p** as Alice
Pick a random number **b**
Keep **b** secret!!
Compute B_{pub} = g^b mod p

$Secret_{Alice}$ = $(B_{pub})^a$ mod p = Secret = $g^{a.b}$ mod p = $Secret_{Bob}$ = $(A_{pub})^b$ mod p

ECC
Elliptic Curve
Cryptography

What is an elliptic curve ?

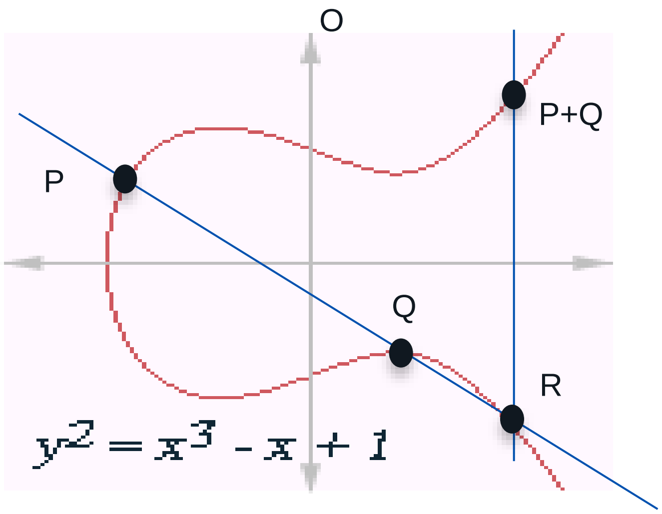


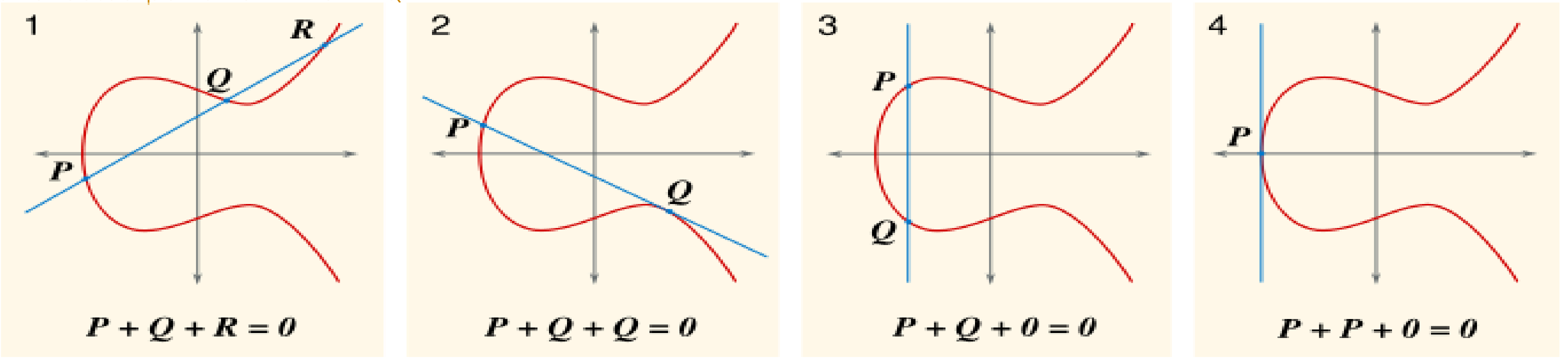


- A curve of general equation $y^2=x^3+ax+b$
  - It MUST be a smooth curve
  - Its discriminant MUST BE NON ZERO: $D = 4A^3 + 27B^2$
- The Elliptic Curve is the set of points
  - that satisfy the equation of the curve (ie. that "belong" to the curve)
  - Plus a special point at infinity that we call O (the letter O)

Elliptic Curve Addition

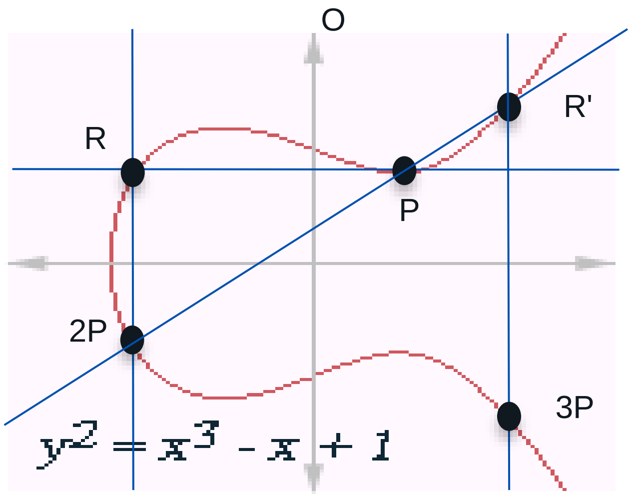
- Let P and Q be two points on the curve
- A line (P,Q) cuts the curve at a third point R
  - If the line is parallel to the Y axis, this point is O
  - If the line is tangent to the curve, the tangent point is counted twice
- The group operator + is defined such as
  - P+Q+R = O; O is the identity
- The reflected point from R is P+Q



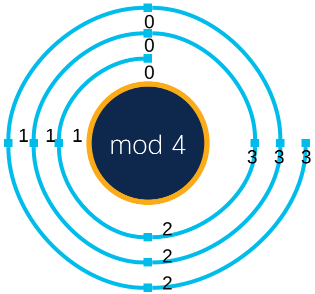


The scalar multiplication n*P

- Let’s start with P+P = 2*P
- For drawing (P,P)
  - draw a tangent to the curve → R
  - (O,R) cuts in P+P=2P
- This is a scalar multiplication
  - One can derive 3P = 2P+P, 4P = 3P+P,…nP = (n-1)P+P



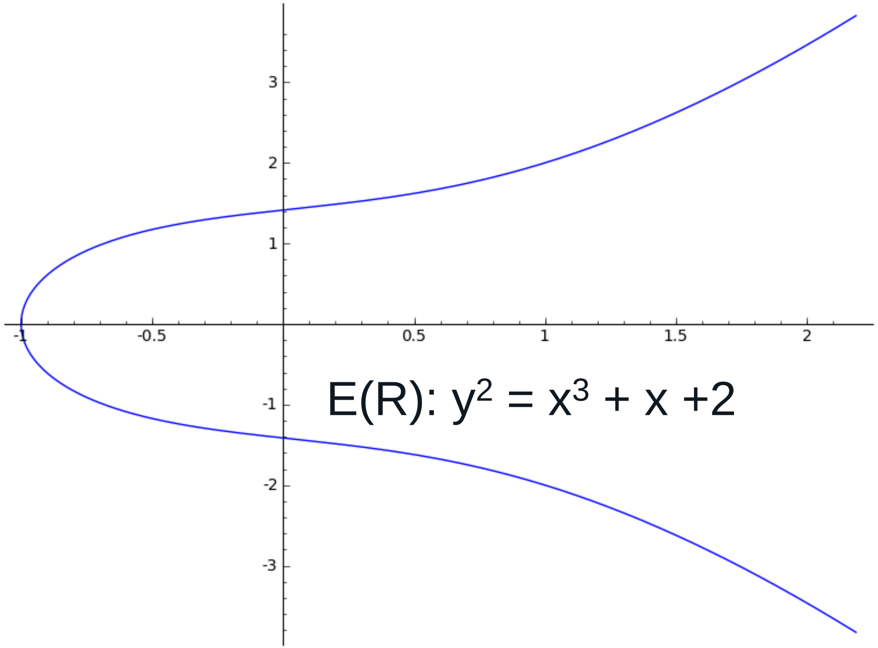
Fast Forward – the finite fields F_m & F_{2^k}

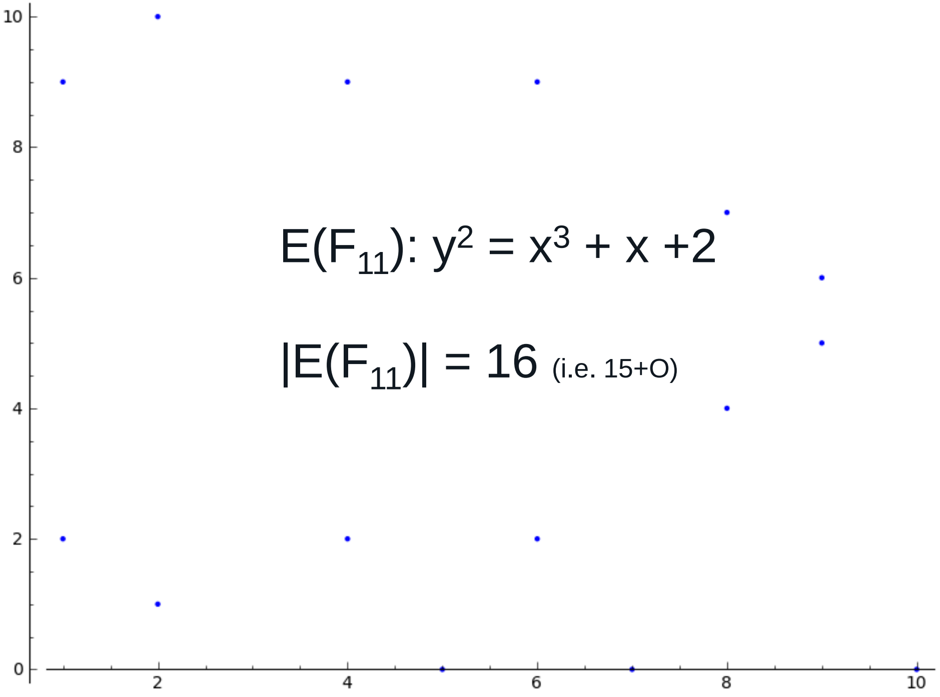


- Remember… modulo arithmetic
- Galois Field = Finite Field
- Let E be an elliptic curve defined over a finite field F_m (modulo m):
  - $E(Fm)$:{∞} U {(x,y) in $F_m x F_m$ | $y^2=x^3+ax+b$, a,b in F_m}
  - $E(F_m)$ is the set of points whose coordinates belong to $F_m x F_m$ and satisfy the equation + point at infinity
  - The set along group operations (+, x) seen before form an Abelian Group under multiplication ➔ a field.
  - For cryptography, m should be a prime number

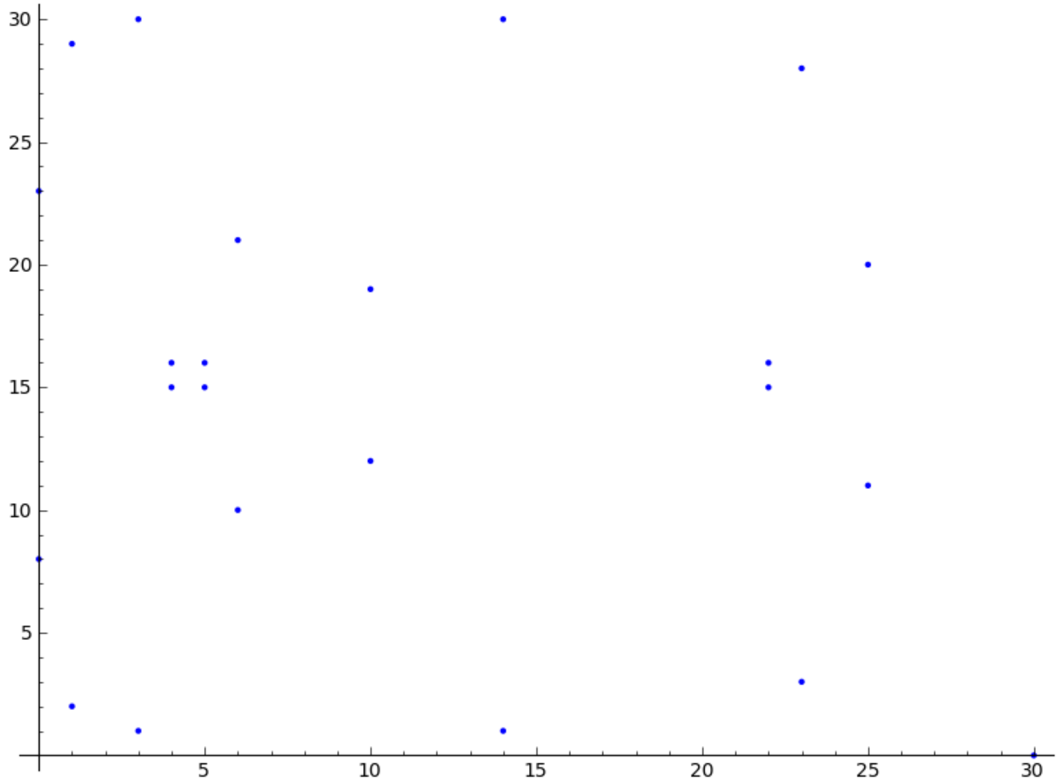
- It **seems (seemed ?)** more computationally efficient if **m = 2^k–1** yielding the notation F_{2^k}
  - Multiplication supposed to be more efficient ➔ very important for ECDH and ECDS
  - In this case, the Koblitz curve is used: $y^2 + xy = x^3 + ax^2 + 1$ where a=0 or a=1
  - For cryptography, k should be a prime number
  - m should remain a prime – it would be called a Mersenne Prime
  - **There is debate about the actual security and efficiency of these curves**!
- The **order of a group *G*** is the **cardinality** of that group written ***ord(G)*** or ***|G|***.
- The order of a **point *P* in a group *G*** is the value *n* such that $n*P = O$ written ***ord(p)*** or ***|p|***

Example Curve





Example on F_{31} – Complexity Increases

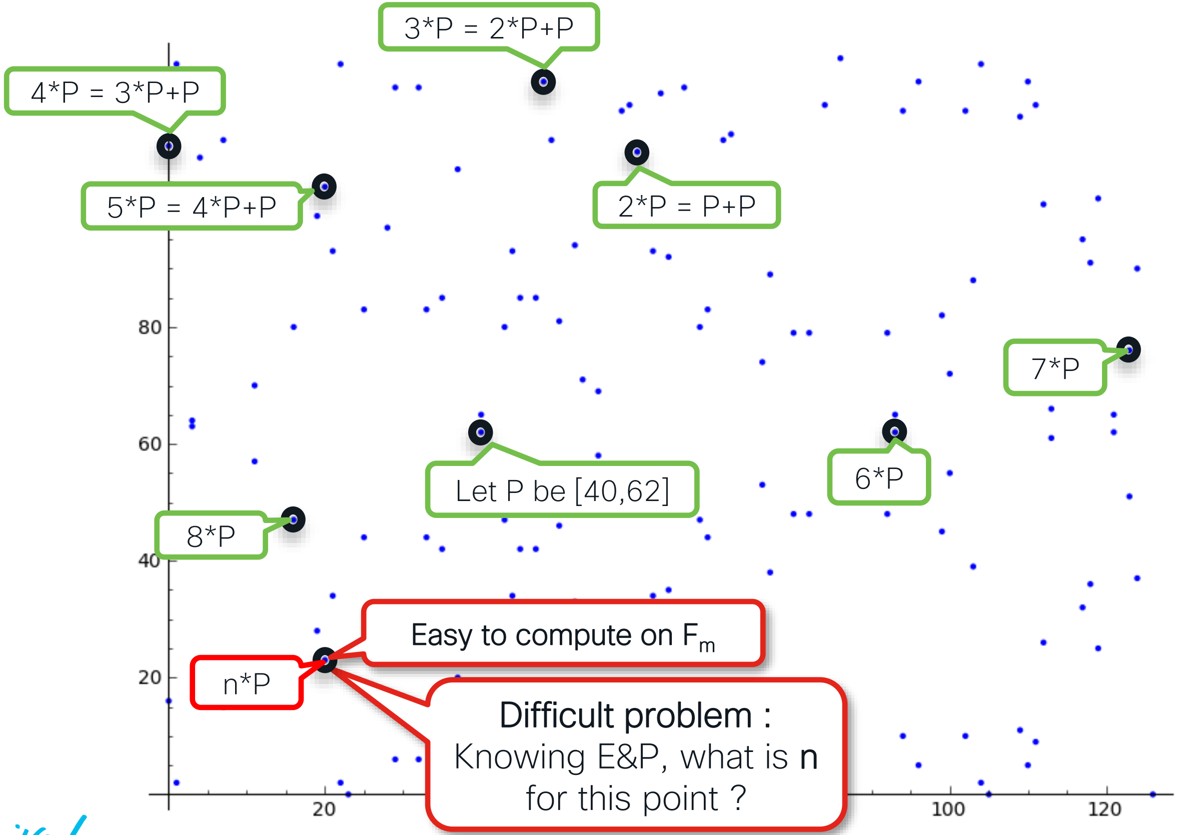


$m = 2^5-1 = 31$

$E(F_{31}): y^2 = x^3 + x + 2$

$|E(F_{31})| = 24$

The same on F_{127} – Complexity Further Increases



$m = 2^7 - 1 = 127$

$E(F_{127}): y^2 = x^3 + x + 2$

$|E(F_{127})| = 136$

ECDH – Elliptic Curve Diffie-Hellman

Alice

Bob

Select a curve **f** and a point **P** on the curve
Pick a random number **a**
Keep **a** secret!!
Compute $\mathbf{A_{pub}} = \mathbf{a*P}$

The curve definition f and point P

$\mathbf{A_{pub}, (P, f(x), m)}$

Attacker can not guess **a**

Attacker can not guess b

$\mathbf{B_{pub}}$

Using the same curve **f** and point **P**
Pick a random number **b**
Keep **b** secret!!
Compute $\mathbf{B_{pub}} = \mathbf{b*P}$

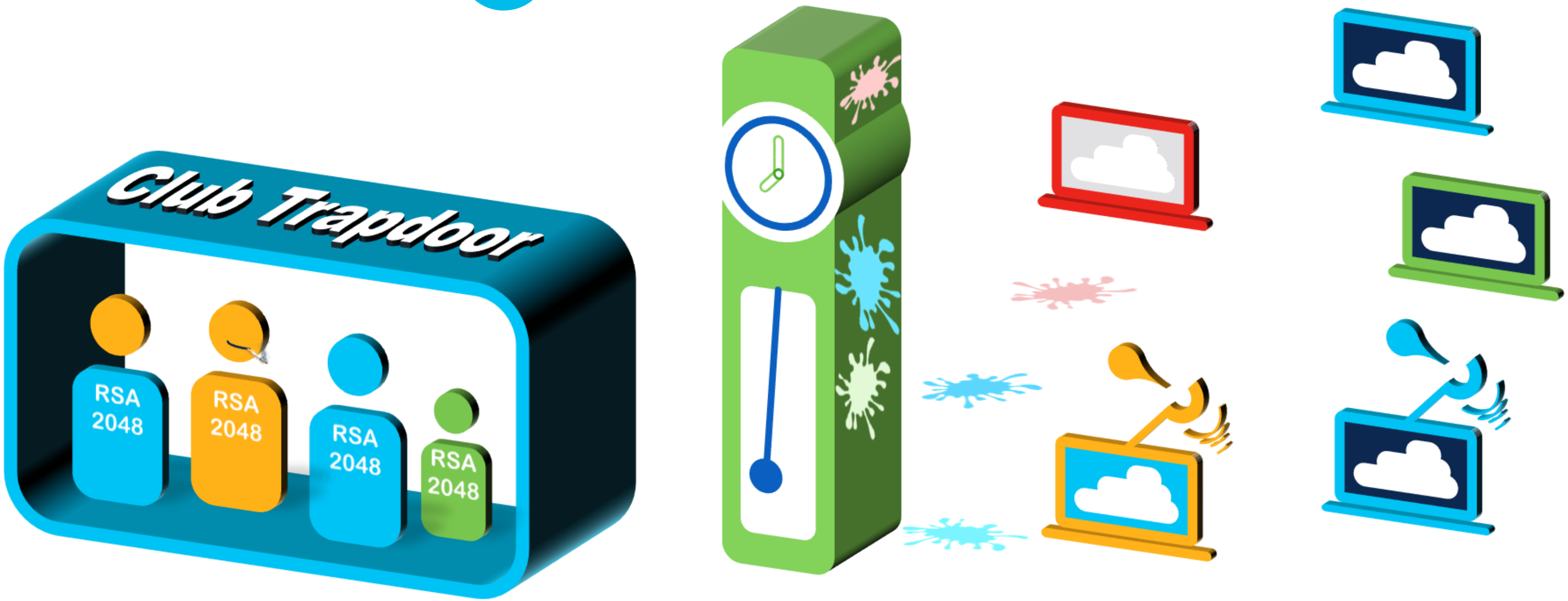
$Secret_{Init} = a * B_{pub}$ = $Secret = a * b * P$ = $Secret_{Resp} = b * A_{pub}$

Enters the Quantum Computer

Today's Cryptography Temporal Defense

TIME PROTECTS PUBLIC KEYS (Until Y2Q)

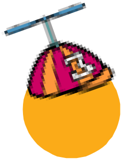
Public Key = Prime 1 x Prime 2



Shor’s Factoring Algorithm

Problem: For a given “N” find a “p” between “1” and “N” that divides “N”

$N = p_1 * p_2$



Peter Shor

1994

Part A
Classic
Reduction of factoring problem to period finding problem

\+

Part B
Quantum
Solve period finding problem using Quantum Fourier Transform

=

(Probabilistic Algorithm)

(Adjust it to taste if it’s wrong)

Shor’s algo converts exponential complexity to polynomial complexity

$x^N \rightarrow N^x$ **where N is the number of bits**

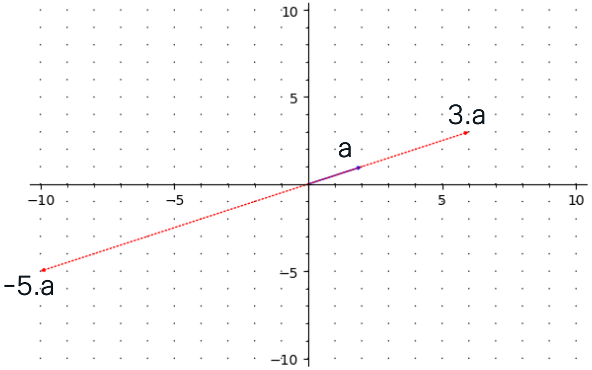
Lattice Based Cryptography

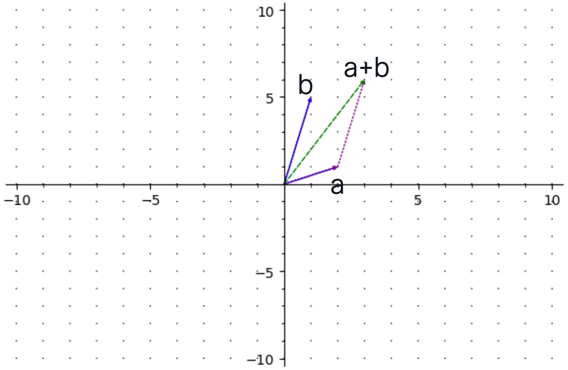
LBC, LWE, NTRU

Vectors

Commonly denoted $\boldsymbol{v}$ or $\vec{v}$
We will use v or $\boldsymbol{v}$

$$\boldsymbol{a}=\begin{pmatrix} a_1 \\ a_2 \\ \\ \dots \\ a_n \end{pmatrix} \qquad \boldsymbol{b}=\begin{pmatrix} b_1 \\ b_2 \\ \\ \dots \\ b_n \end{pmatrix}$$



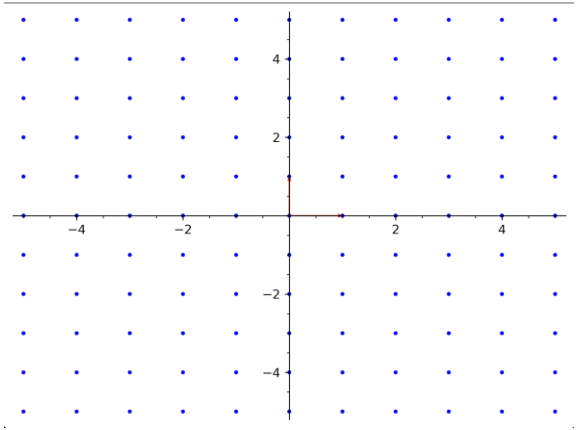


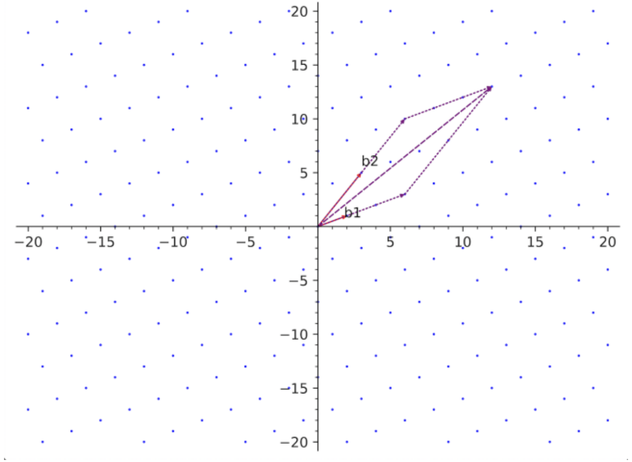
Operation	Formula	Result type
addition	$\boldsymbol{a}+\boldsymbol{b} = (a_1+b_1,\dots,a_2+b_2)$	Vector
Scalar multiplication	$x.\boldsymbol{a} = x.a_1+\dots+x.a_n$	Vector
Inner product	$\boldsymbol{a}.\boldsymbol{b} = a_1b_1+\dots+a_nb_n$	Scalar (number)

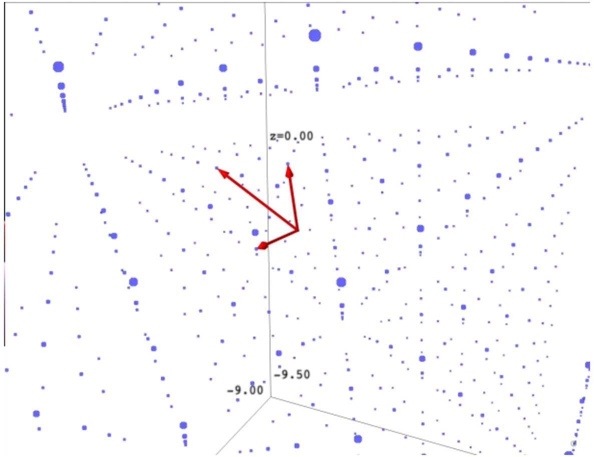
This is not a lattice

What is a Lattice ?

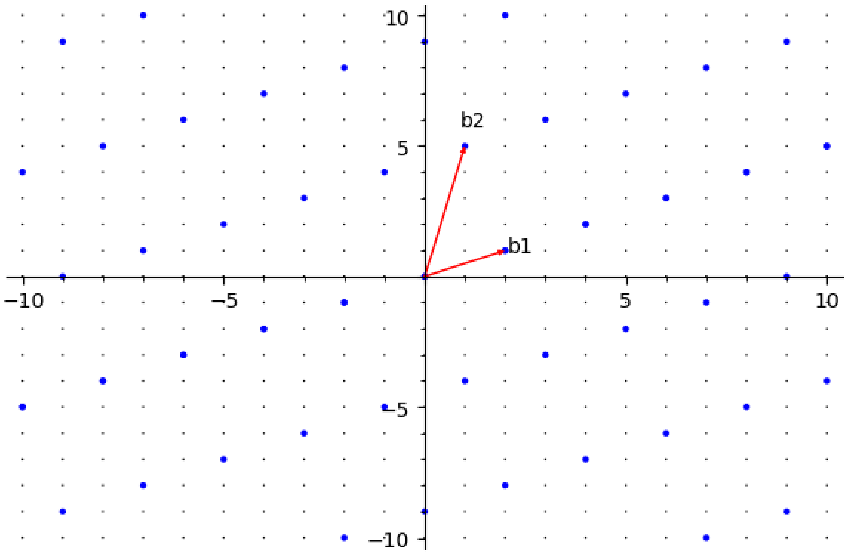
- A periodic “grid” in $\mathbb{Z}^m$
- All integer linear combinatons of n basis vectors $b_1, b_2, \ldots, b_n$
- Basis B = $\{b_1, \ldots, b_m\}$
- Lattice $\mathcal{L} = \sum_{i=1}^{m} a_i \cdot \boldsymbol{b}_i \ , a_i \in \mathbb{Z}$

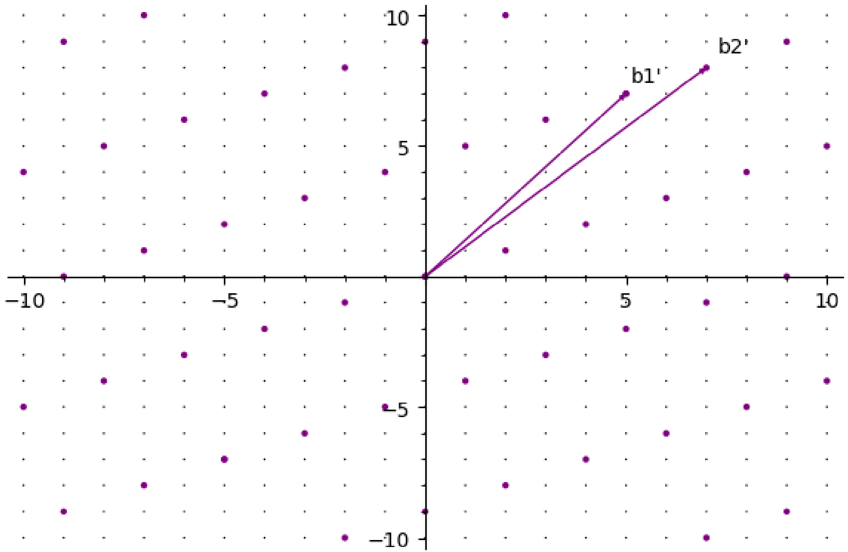






Good Basis, Bad Basis

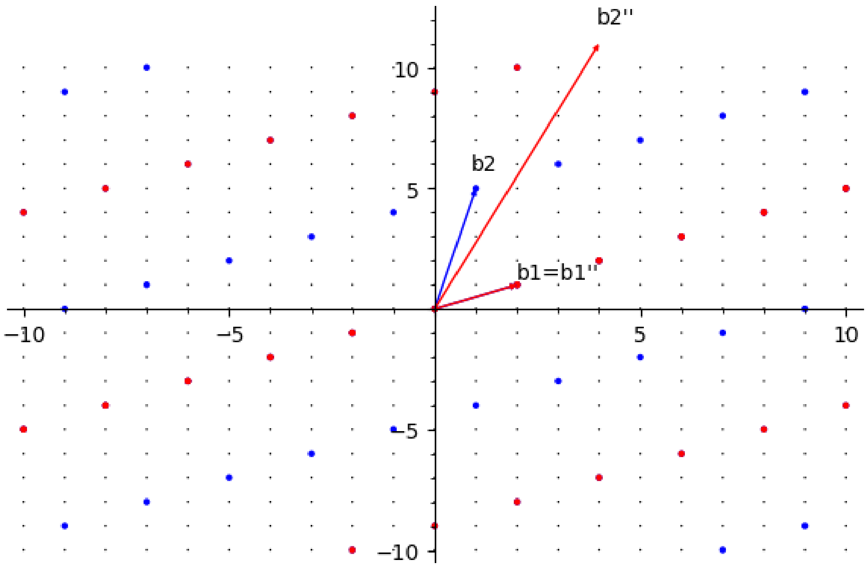




B= $\{b_1, b_2\}$: good basis
Short, almost perpendicular vectors

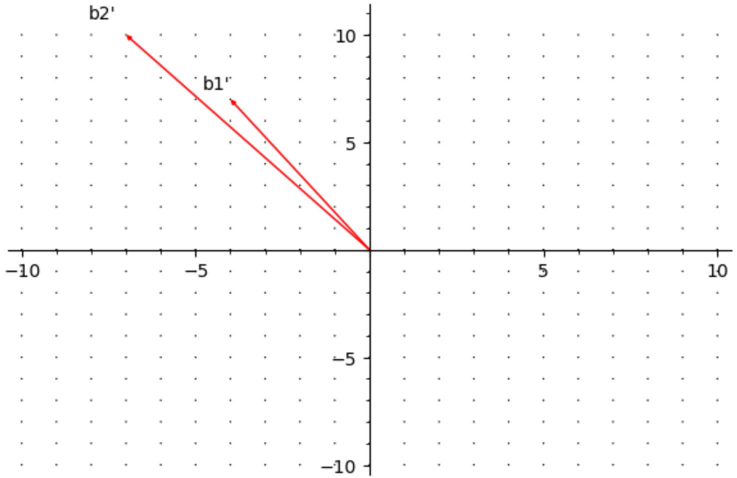
B= $\{b_1', b_2'\}$: bad basis
Long, not very perpendicular vectors

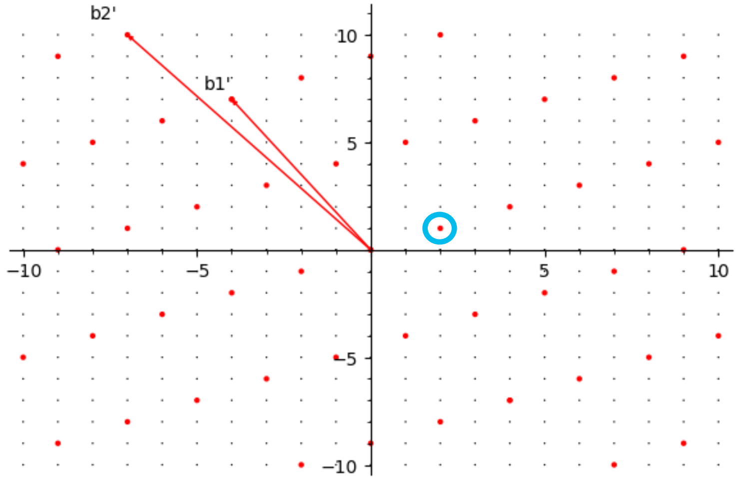
Not a basis



$B = \{b_1'', b_2''\}$: not a basis
The lattices do not overlap fully

Short Vectors is a Hard Problem



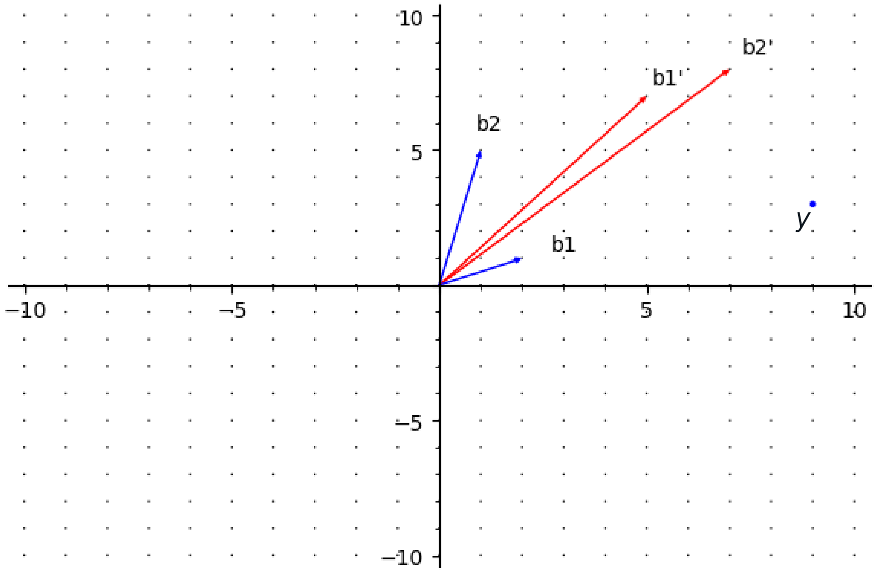


Given this base, what is the shortest possible non-trivial vector ?

Surprise!

Closest Vector Problem is a Hard Problem

What is the closest lattice vector to y ?



Babai's round-off algorithm:

$$\boldsymbol{v} = B.\lfloor B^{-1}.\boldsymbol{y} \rceil$$

Theorem:

$$\|\boldsymbol{v} - \boldsymbol{y}\| \le \frac{1}{2} \sum \|\boldsymbol{b_i}\|$$

In clear:
Better base ➔ Closer vector

Short Integer Solution & Learning With Errors

SIS

$\boldsymbol{Az} = 0$ with 'short' $\boldsymbol{z} \neq 0$

Average case SVP

(Bounded Distance Decision)

$$\mathcal{L}^{\perp}(A) = \{z \in \mathbb{Z}^m : A\boldsymbol{z} = 0\}$$

LWE

$(A, \boldsymbol{b}^t = \boldsymbol{s}^t A + \boldsymbol{e}^t)$ vs. $(A, \boldsymbol{b}^t)$

Average case BDD

(Bounded Distance Decision)

$$\mathcal{L}\,(A) = \{\boldsymbol{z}^t \equiv \boldsymbol{s}^t A \bmod q\}$$

These are other ways to define a lattice

Example Only!

Goldreich, Goldwasser, Halevi Cryptosystem

Alice

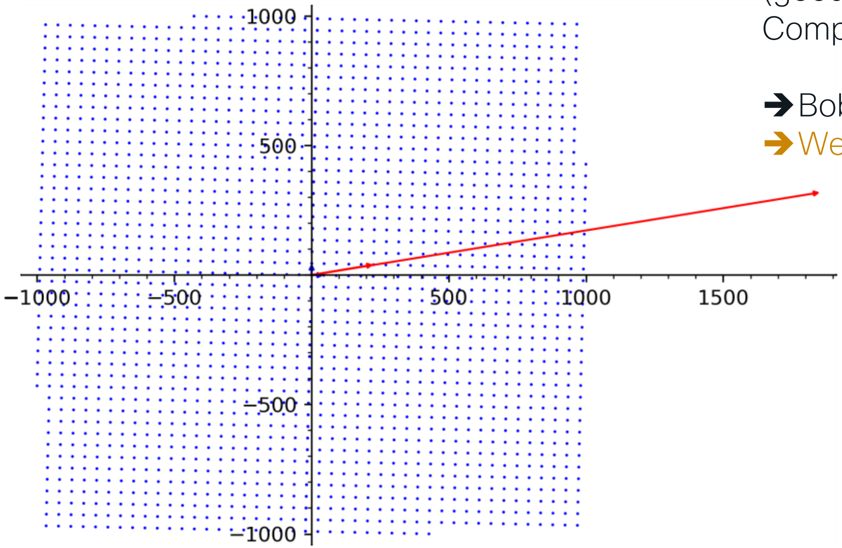
Must send a private message m

Takes V' from Bob
Computes $l = m * V'$
Generates an error vector r (n)
Computes $c = l + r$

Quantum secure
… but broken ☹

Eve can not guess *m* because of the Closest Vector Problem (bad basis)

c



Bob

Generates V (nxn), a good basis
Publishes V' (nxn), a bad basis

Applies Babai on (V, *c*) → finds *l*
(good basis → accuracy)
Computes $m = l * V'^{-1}$

➔ Bob has reversed the operation !!
➔ We have an encryption scheme

NIST Post Quantum Algorithm Selection

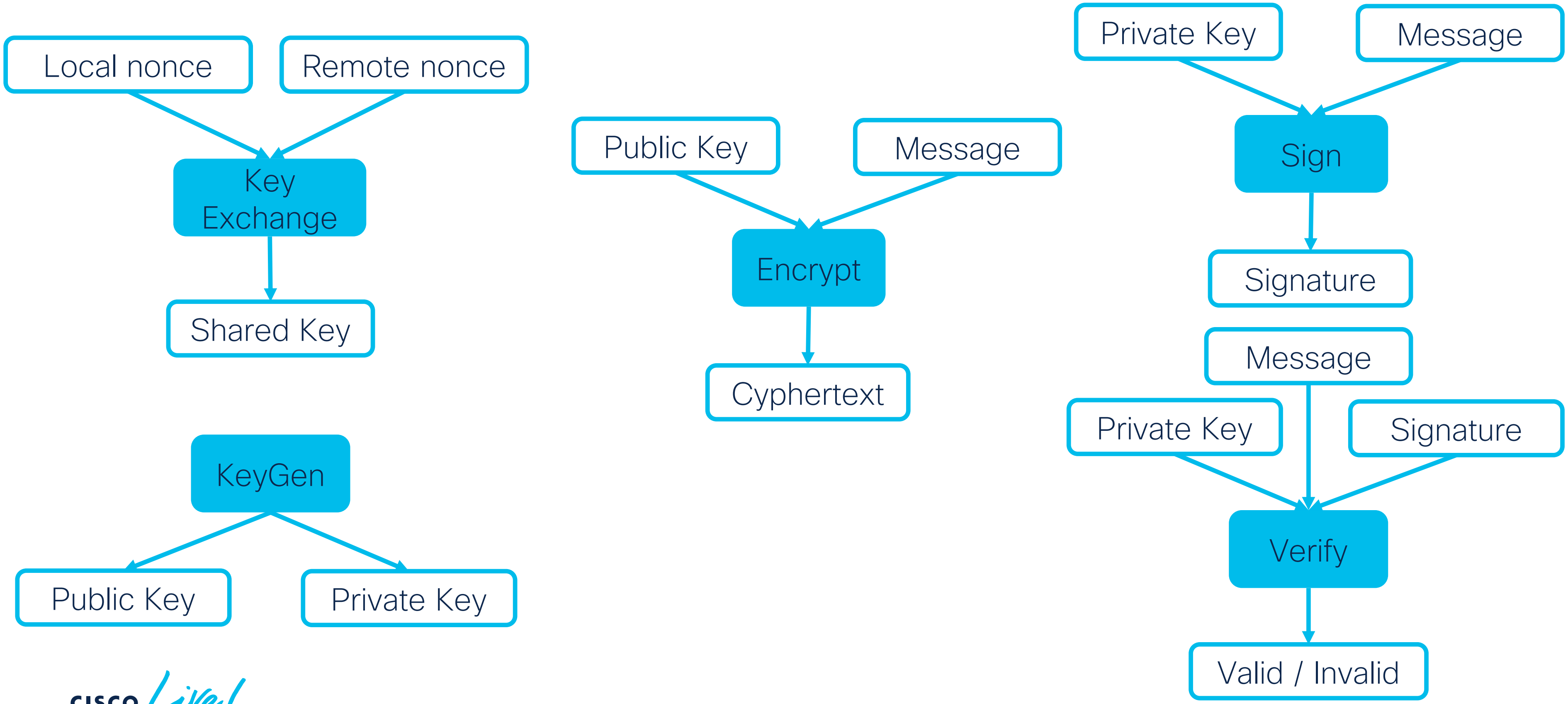
Selected Algorithms 2022

Type	Name	Math
Pub Key Encr and Key Exchange	CRYSTALS-KYBER	Lattice LWE (CVP)
Digital Signature	CRYSTAL-DILITHIUM	Lattice LWE (CVP)
Digital Signature	FALCON	Lattice NTRU(SVP) + FFT
Digital Signature	SPHINCS^{+}	Stateless hash-based

Ref: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

In Summary

Main Public-Key Cryptographic Primitives



Business Outcome

- Crypto is not broken; it evolves and so do attackers.
- These are good news! The more research, the more insight.
- Lattice-based cryptography is Post-Quantum ready

Evolve your systems as new recommended algorithms are released !

A Short Bibliography

- NIST SP 800-90A : Recommendations for Random Number Generation Using Deterministic Random Bit Generators
- NIST SP 800-38D : Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
- NIST SP 800-56A (R2): Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (i.e. DH, ECDH + key derivation methods)
- NIST 800-131Ar1: Transitions: Recommendations fro Transitioning the Use of Cryptographic Algorithms and Key Lengths
- NIST FIPS 140-2: Security Requirements for Cryptographic Modules
- NIST FIPS 186-4: Digital Signature Standard (DSS) (DSA, RSA (PKCS#1), ECDSA,…)
- NIST FIPS 180-4: Secure Hash Standard (SHA-1, SHA-256,…, SHA-512)
- NIST Routines: https://www.nsa.gov/ia/_files/nist-routines.pdf (Curve P-192, P-224, P-256 etc.)
- Safe Curves: http://safecurves.cr.yp.to
- Transcript Collision Attacks: Breaking authentication in TLS, IKE and SSH: http://www.mitls.org/downloads/transcript-collisions.pdf
- Simons institute:
  - https://simons.berkeley.edu/workshops/schedule/10563
  - https://simons.berkeley.edu/workshops/lattices-2020-boot-camp

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.

- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

The bridge to possible

Thank you

cisco Live!

ALL IN