Cisco live!

January 29 - February 2, 2018 · Barcelona

BRKSEC-3005

Cryptographic Protocols and Algorithms

Frederic Detienne – Distinguished Engineer

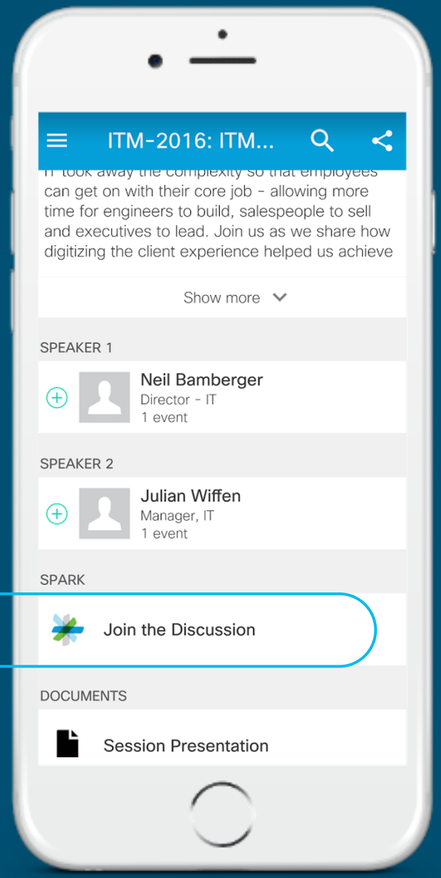
Cisco Spark

Questions?

Use Cisco Spark to communicate with the speaker after the session

How

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



cs.co/ciscolivebot#BRKSEC-3005

My Professional Life

- Belgian
  - live in Aywaille
- Joined Cisco on January 1, 1997
  - fd@cisco.com
- Distinguished Engineer (TAC)
  - Web Content, AAA, Firewalls, VPNs, IPTV
  - Bit of everything (stuff nobody else wanted)
  - Made DMVPN, then FlexVPN
  - Focus on Serviceability





Agenda

- A Brief Introduction
- Hash and HMAC's
- Symmetric Encryption
- MODP: Multiplicative Group of Integers Modulo P
- ECC: Elliptic Curve Cryptography
- Performances and Security
- Practical Applications; IKEv2, SSL and PKI
- Attacks, Weaknesses & Self-Inflicted Pain
- Conclusion and Recommendations

Introduction

The Tree of Math

What happened to that tree in front of math class ?

It grew squared roots

Cryptographic Mechanisms

Encryption

Signatures

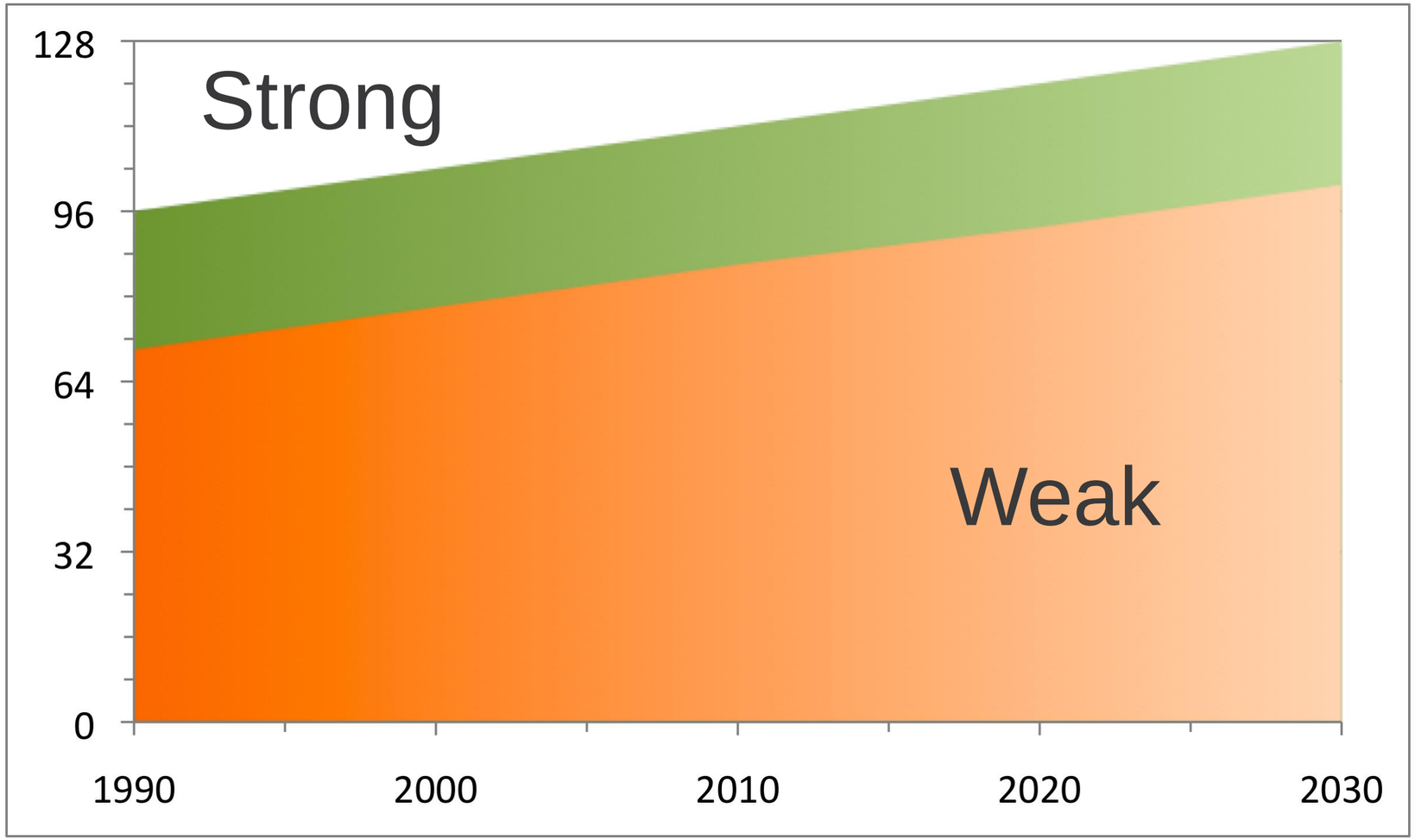
Data Authentication (HMAC)

Random Number Generation

Key Establishment

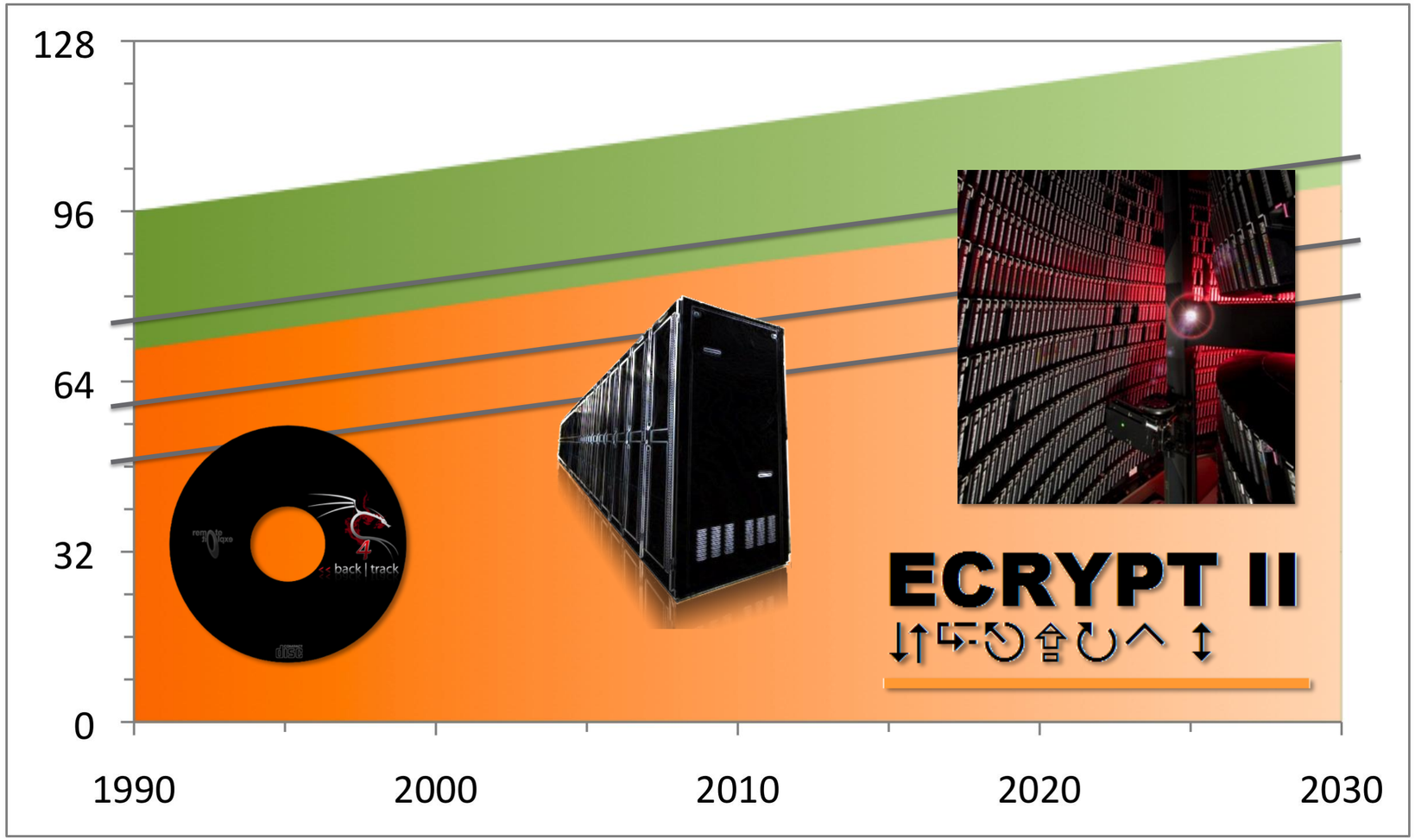
Hashing

Key Strength

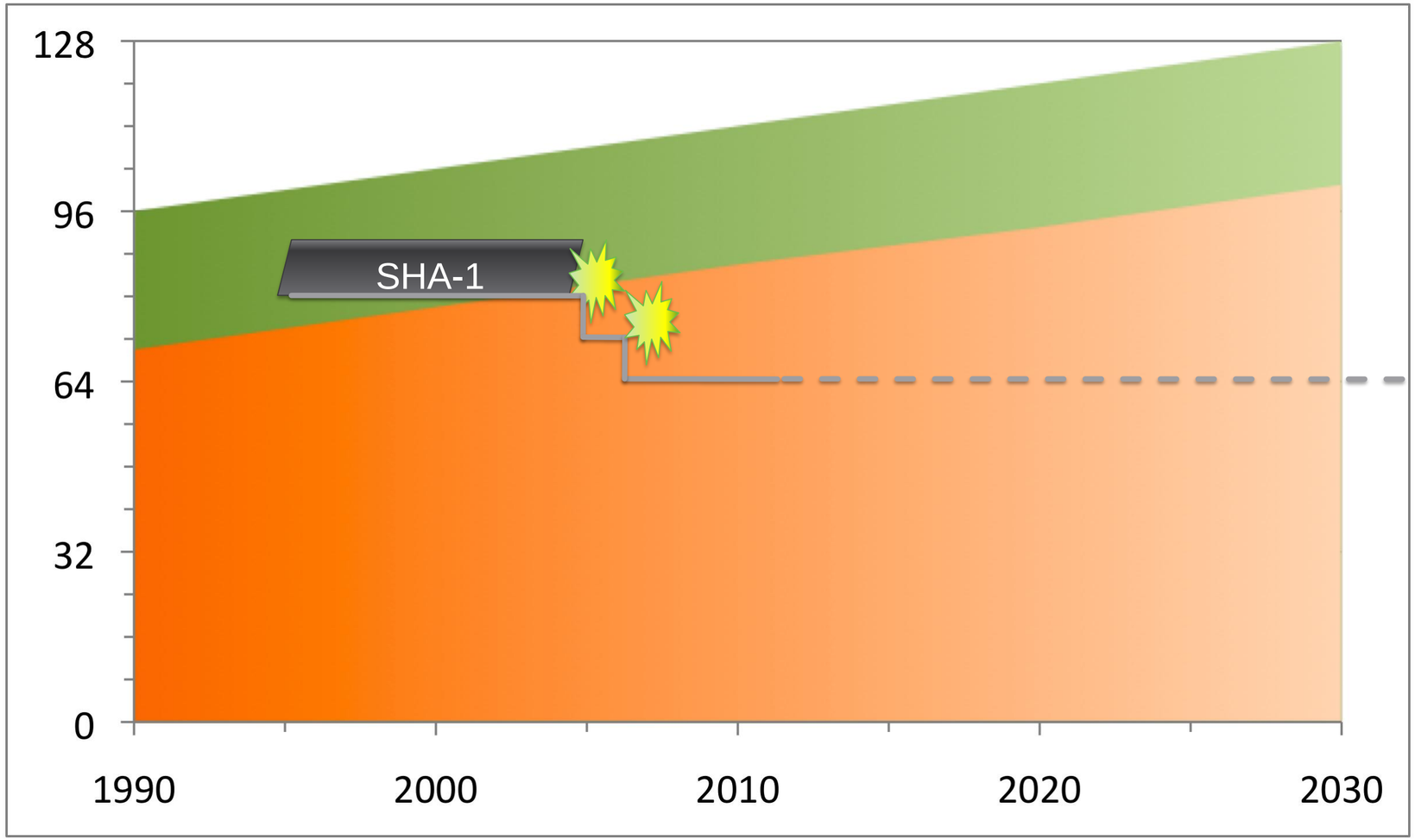


Sources: Lenstra and Verheul, NIST

Attacker Strength

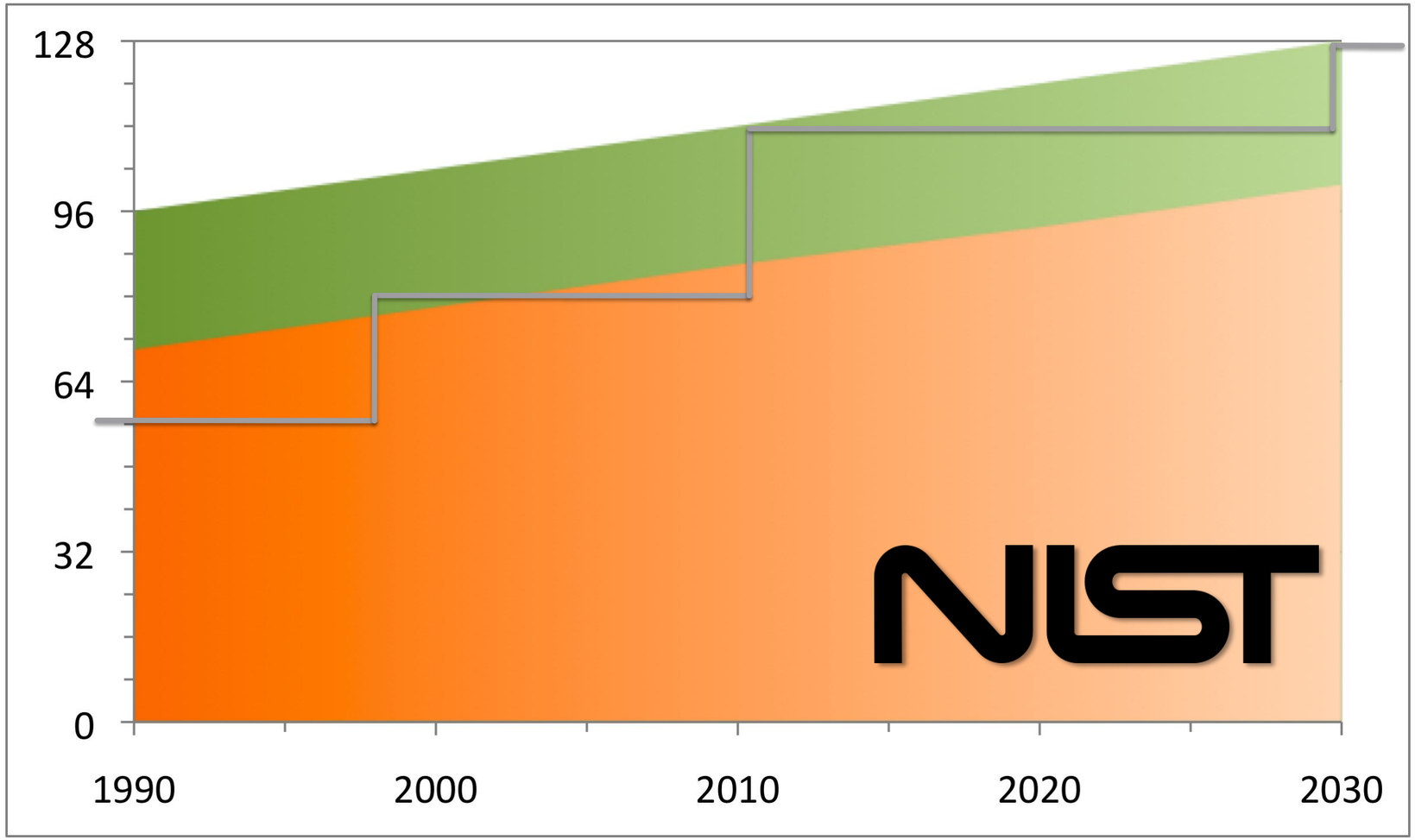


Algorithms Never Get Stronger

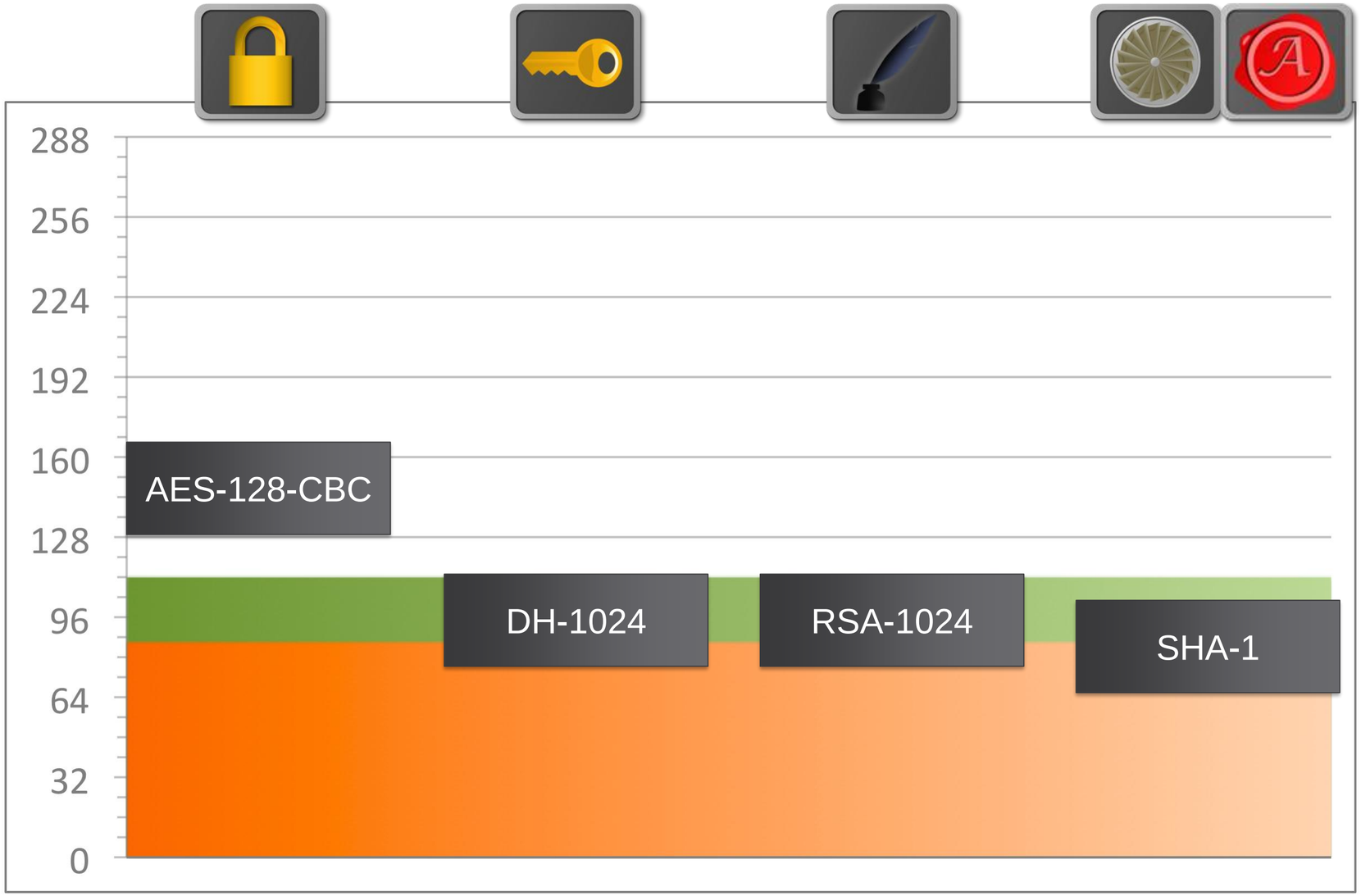


Sources: FIPS-180-1, Wang, Yin, Yu ‘05, Cochran ‘07

Strength increases by steps

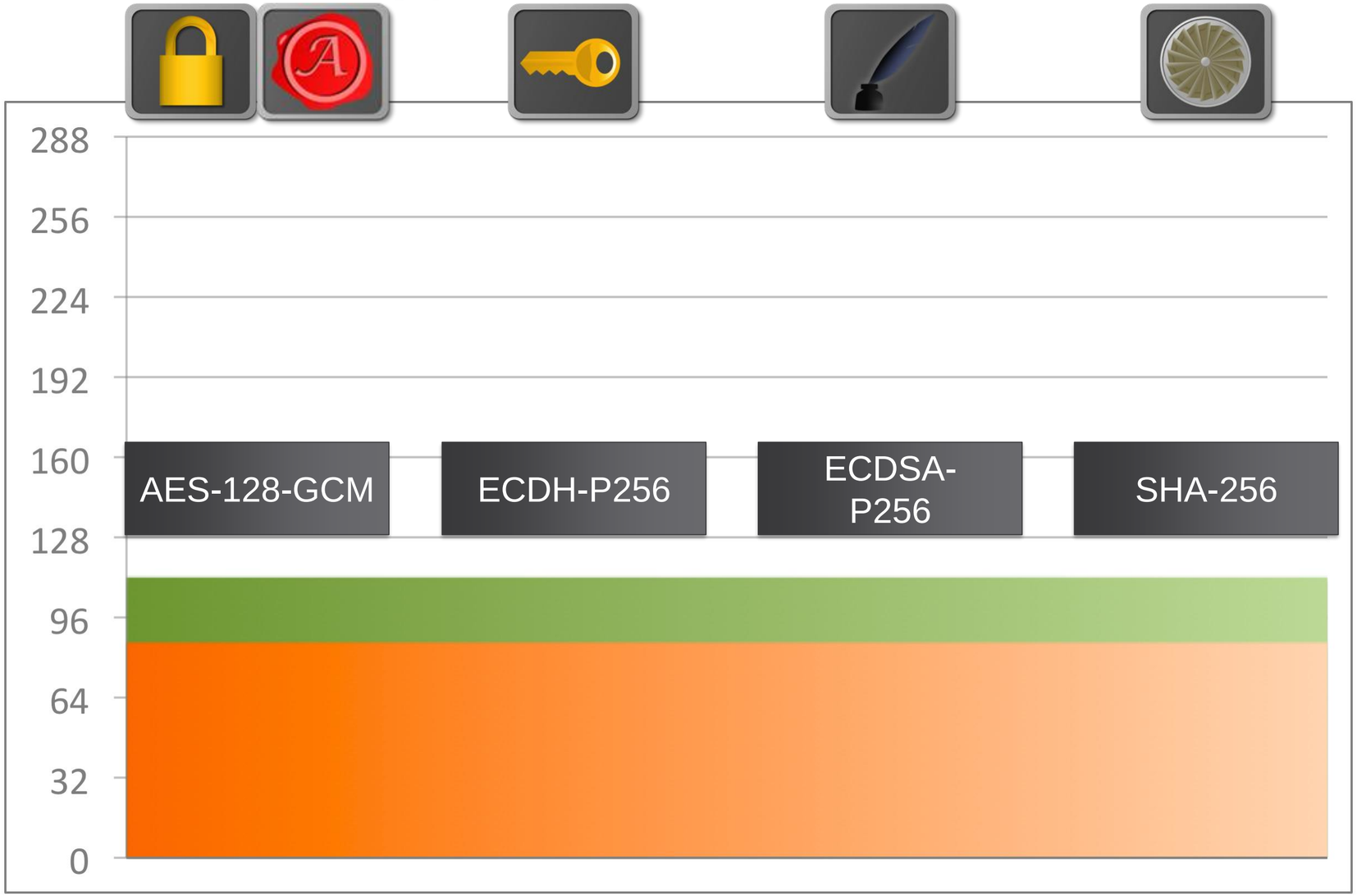


Prevalent



Next Generation Encryption

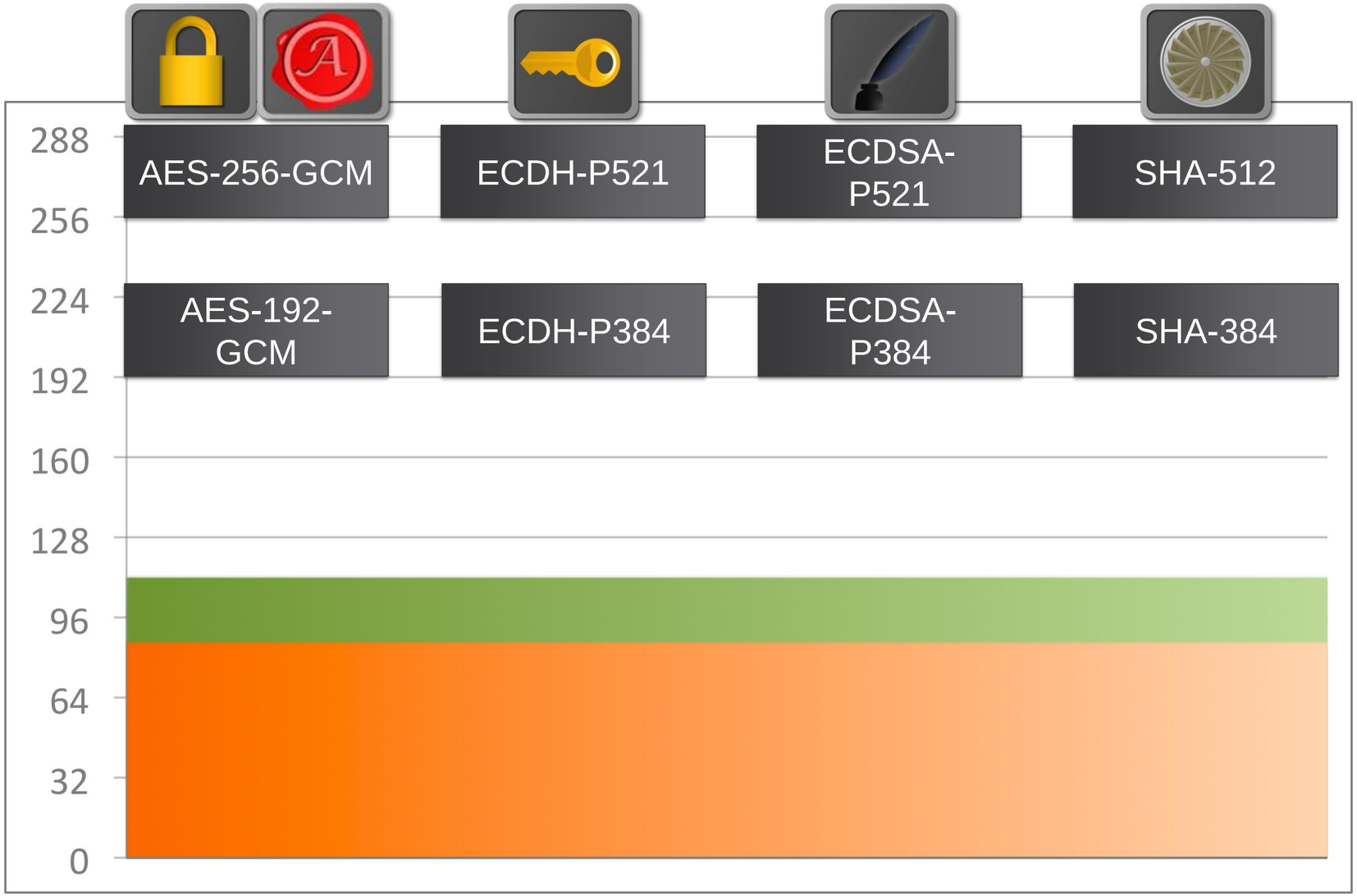
128-bit ➔ ±30 years



NGE higher security levels

Long term ☺

“Foreseeable future”

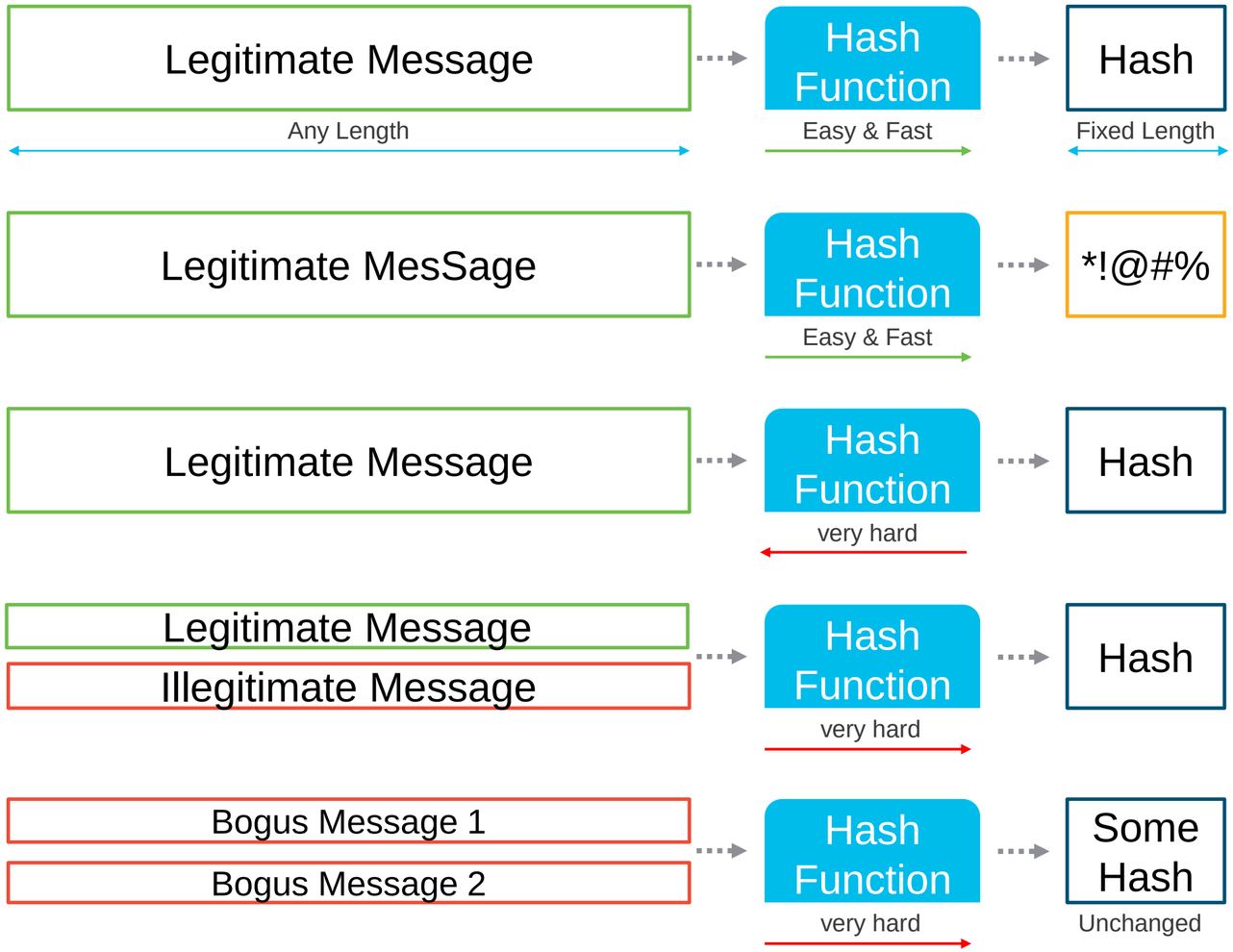


Next Generation Encryption

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)

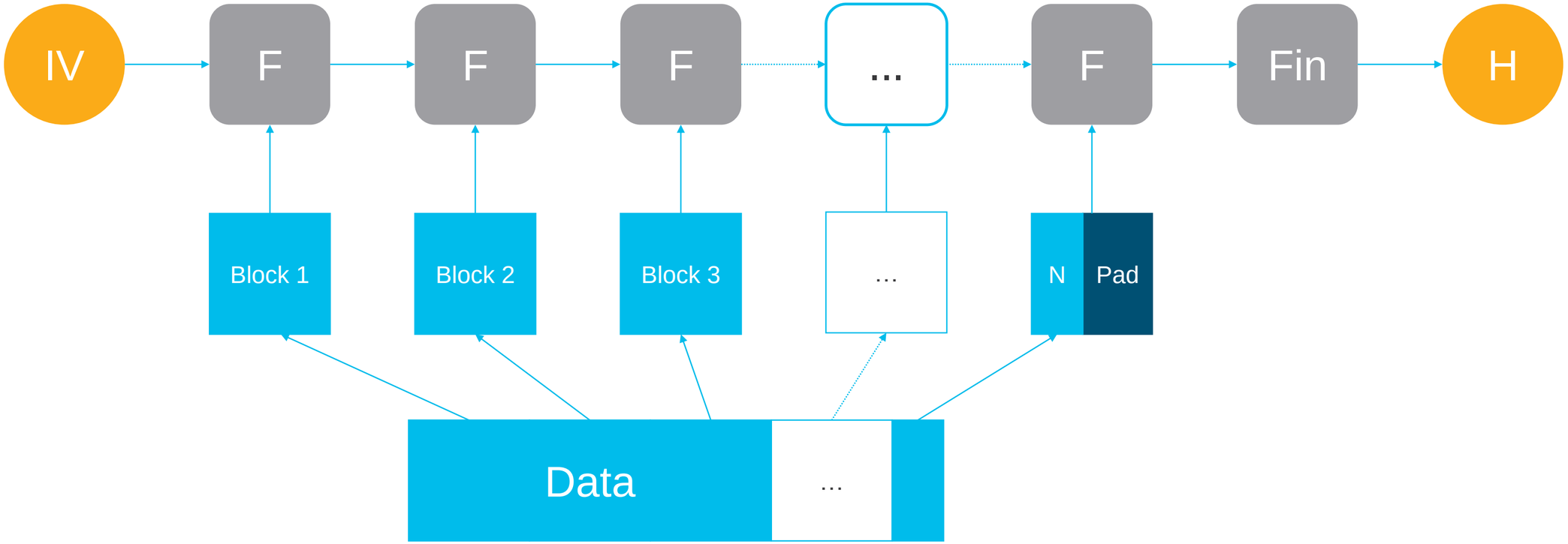
Second pre-image resistance

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

Collision resistance

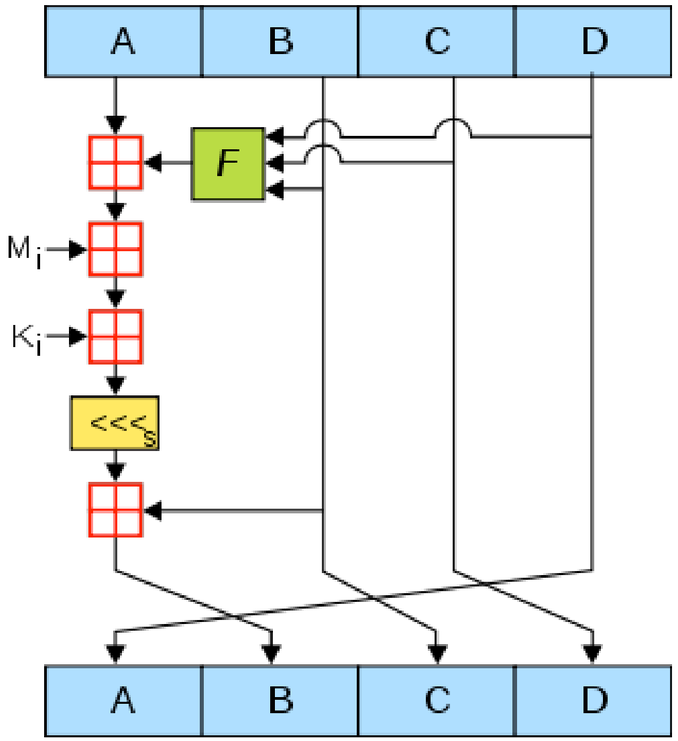
(attacker gets to select message 1 and 2 as long as they hash equally)

The Merkle–Damgård Construction



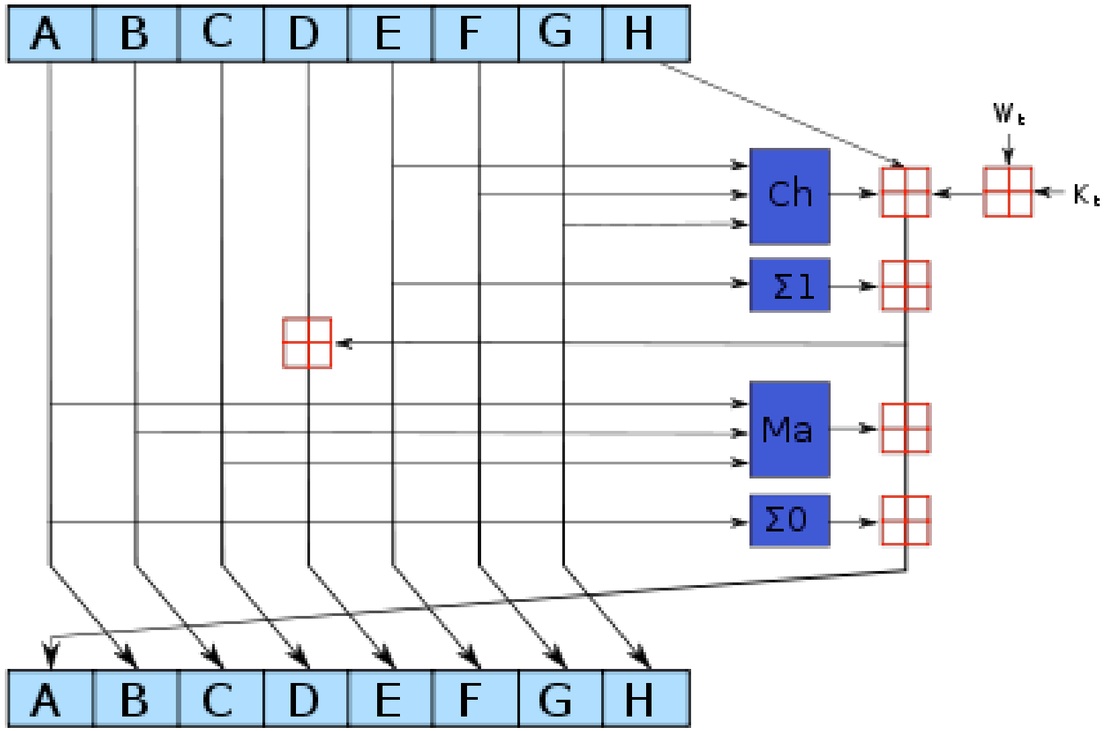
MD-5 vs SHA-2 – Hash Functions

MD-5
(60 rounds)



$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

SHA-2
(64-80 rounds)



$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

Reference: Wikipedia

Rough Hash Algorithms Comparison

Algorithm/variant		Output size	Max msg size	Collisions found
MD-5		128	2^{64}-1	yes
SHA-1		160	2^{64}-1	yes (hard)
SHA-2	SHA-256	256	2^{64}-1	no
	SHA-384	512	2^{128}-1	no
	SHA-512	512	2^{128}-1	no

Note the increasing output size !

IPsec truncates output to 128, 192 or 256 bits (16, 24 or 32 bytes).

Not IKE

Algorithm ID	Block Size	Output Length	Trunc. Length	Key Length	Algorithm Type
HMAC-SHA-256-128	512	256	128	256	auth/integ
HMAC-SHA-384-192	1024	384	192	384	auth/integ
HMAC-SHA-512-256	1024	512	256	512	auth/integ
PRF-HMAC-SHA-256	512	256	(none)	var	PRF
PRF-HMAC-SHA-384	1024	384	(none)	var	PRF
PRF-HMAC-SHA-512	1024	512	(none)	var	PRF

Reference: Wikipedia, RFC4868

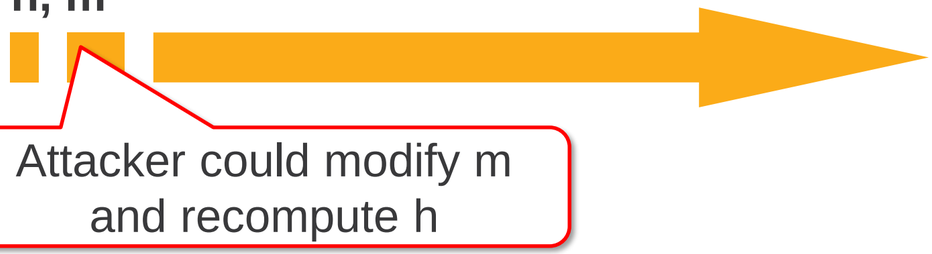
Using a Hash or an HMAC

Alice

Must send a message **m**
Goal: **data does not get corrupted in transit**

Computes h=HASH(m)

h, m



Bob

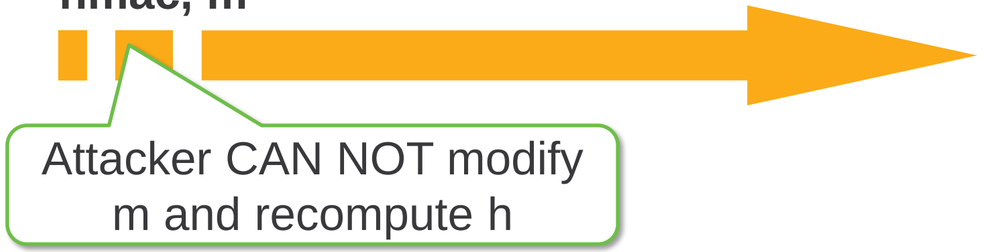
Computes h'=HASH(**m**)
Checks h' = h
If yes → message is valid
if no → message was damaged

Collision Resistance

Share pre-shared key **k** with Bob
Must send a message **m**
Goal: **Bob assured data comes from Alice**

Computes hmac=HASH(**m|k**)

hmac, m



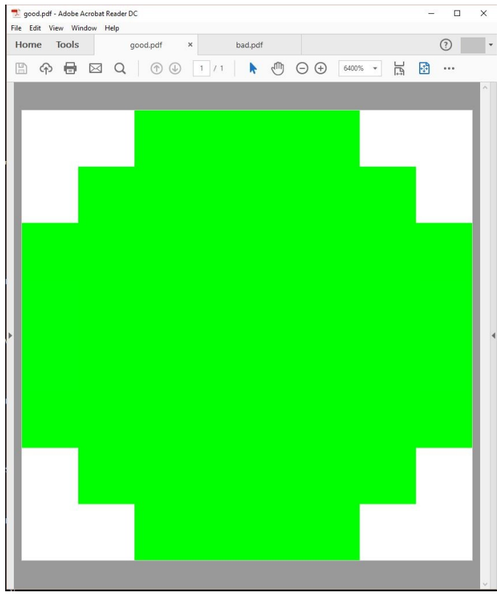
Share pre-shared key **k** with Alice

Computes h'=HASH(**m|k**)
Checks h' = h
If yes → message is valid
if no → message was damaged

Unforgeability

SHA-1 Collisions with SHAttered

good.pdf



This attack required over 9,223,372,036,854,775,808 SHA1 computations.

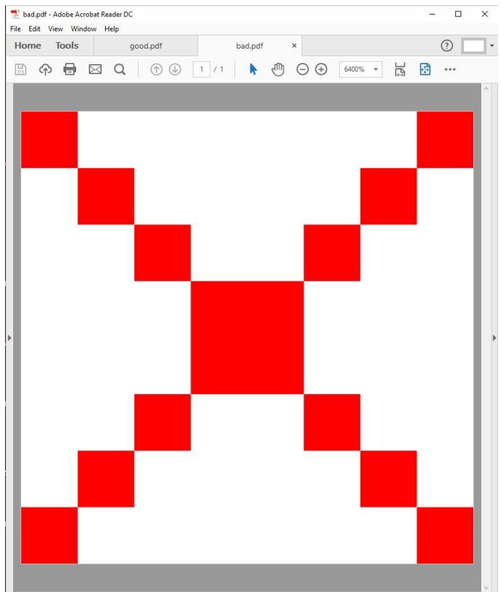
This took the equivalent processing power as 6,500 years of single-CPU computations and 110 years of single-GPU computations.

-SHAttered.io

The SHAttered attack is 100,000 faster than the brute force attack that relies on the birthday paradox. The brute force attack would require 12,000,000 GPU years to complete, and it is therefore impractical.

-SHAttered.io

bad.pdf



SHA-1 checksum:

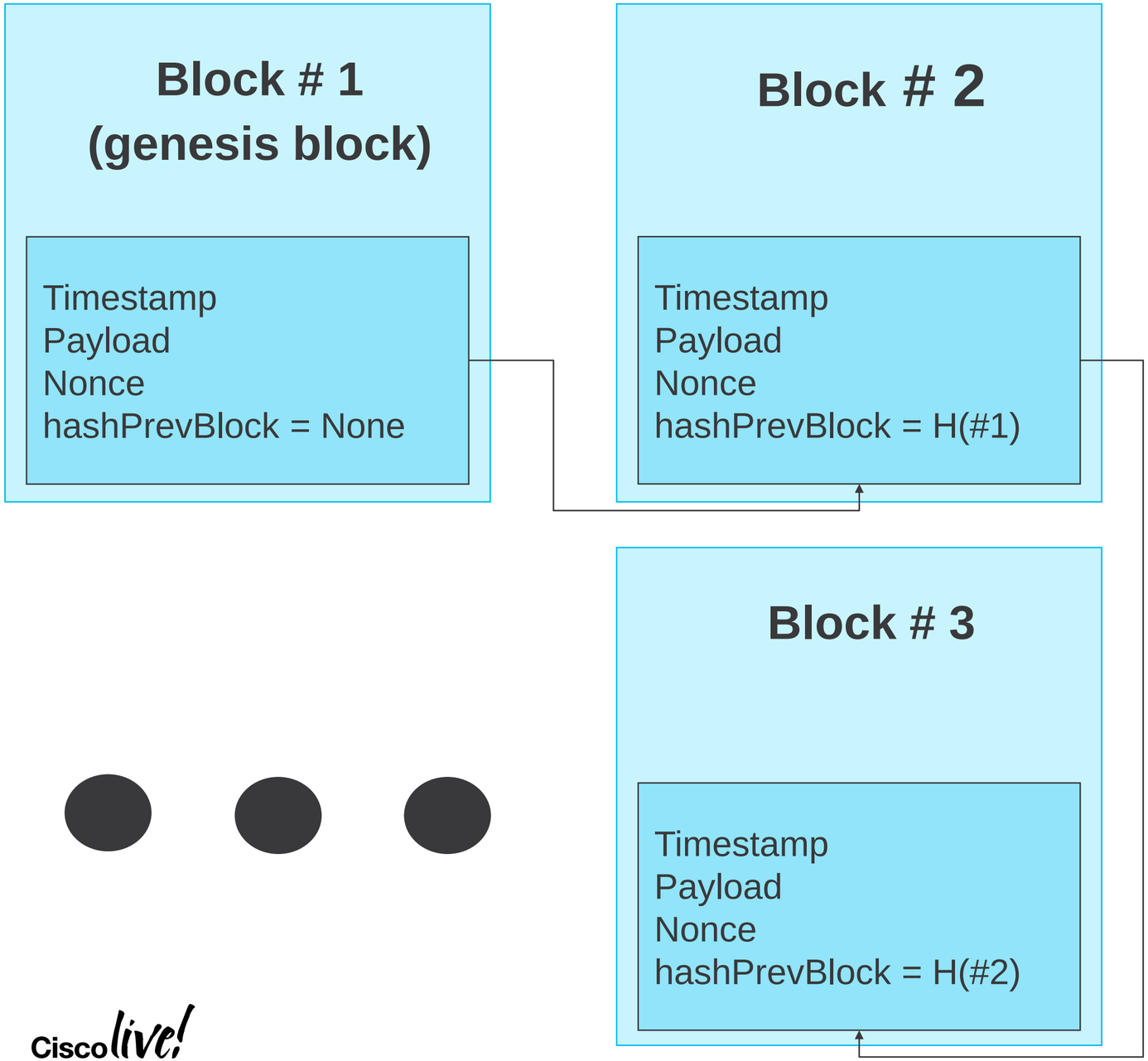
d00bbe65d80f6d53d5c15da7c6b4f0a655c5a86a

SHA-1 checksum:

d00bbe65d80f6d53d5c15da7c6b4f0a655c5a86a

Applied hashing: Blockchain & bitcoin & …

Blockchain




```
class Block(object):
    def __init__(self, hashPrevBlock, payload):
        self.hashPrevBlock = hashPrevBlock
        self.payload = payload
        self.timestamp = datetime.now()
        self.Nonce = self.mine()

    def hash(self, Nonce=None):

    def mine(self):
```

```
class BlockChain(object):
    def __init__(self, size):

    def verify(self):
```

Mining

```
def mine(self):
    # let's calculate hash until we have "0000" at the beginning
    for nonce in range(1, 10000000):
        attempt = self.hash(nonce)

        if attempt.startswith("0000"): # Mathematical challenge (find hash that starts with x-number of 0's
            logger.debug("We found matching hash and we are setting it as Nonce: " + attempt)
            self.Nonce = attempt
            return self.Nonce

    logger.error("We couldn't find nonce satisfying 0's condition")
    exit(1)
```

Purpose of mining is to reach a secure, tamper-resistant consensus

Reward: Transaction fee + Reward (initially 50BTC, nowadays 12.5)

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**

	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	4	12	2	10	11	(c - m) mod 26
=	**KEY**					

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

Assumption #2: Attacker can wiretap ciphertext

Conclusion: Attacker can compute the key easily

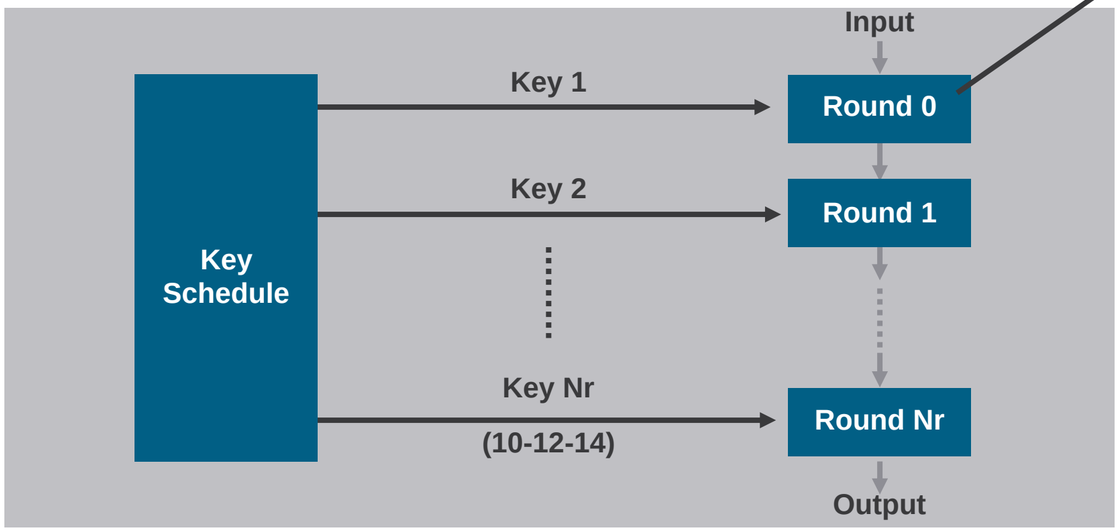
➔ DO NOT REUSE KEY !!

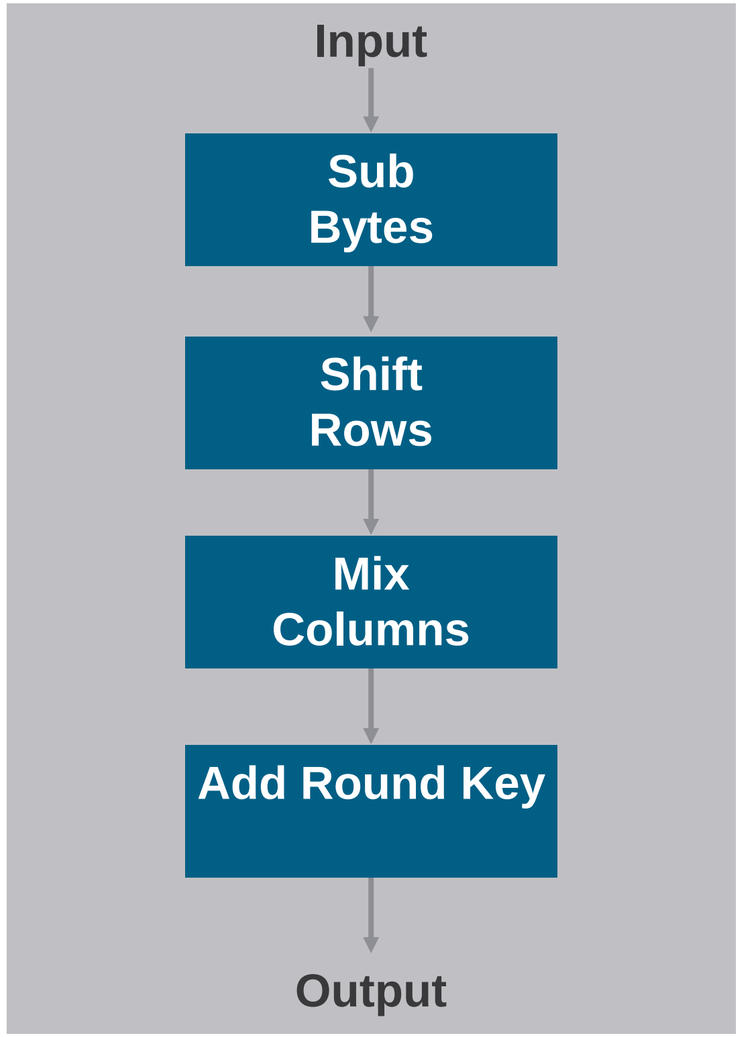
AES – The Advanced Encryption Standard

- The block size is large (128 bits standardized)
- The key size is large (128, 192 or 256 bits)
- AES operates on full bytes (faster on general purpose CPU's)
- National Institute of Standards and Technology:
  - "A machine that cracks 56 bits DES in 1 seconds takes 149 trillions years to crack 128-bits AES"
- Summary:
  - AES is faster and more secure than DES or 3-DES
  - AES is easier to implement than DES on tight hardware
  - **IS THIS TRUE ?**

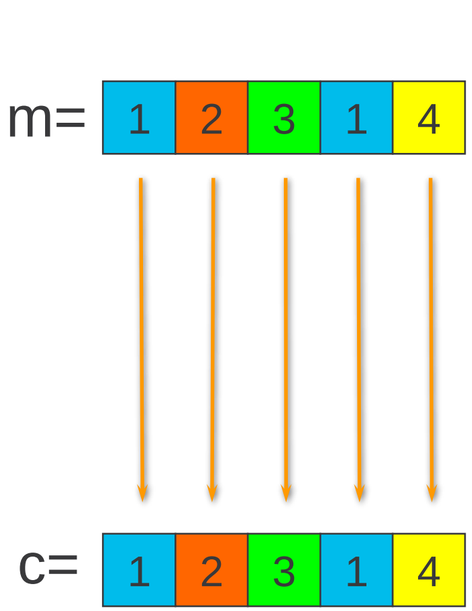
AES: Individual Rounds

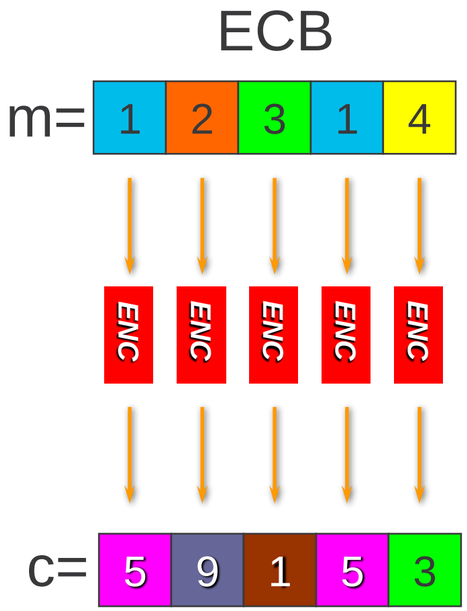
Note: Last Round Is Slightly Different from the Rest of the Rounds (no Mix Columns)

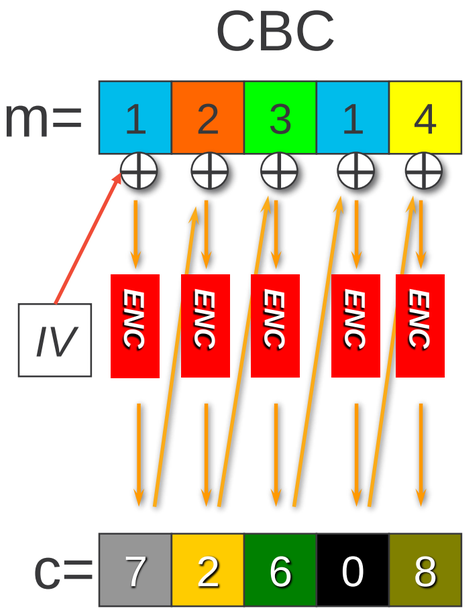


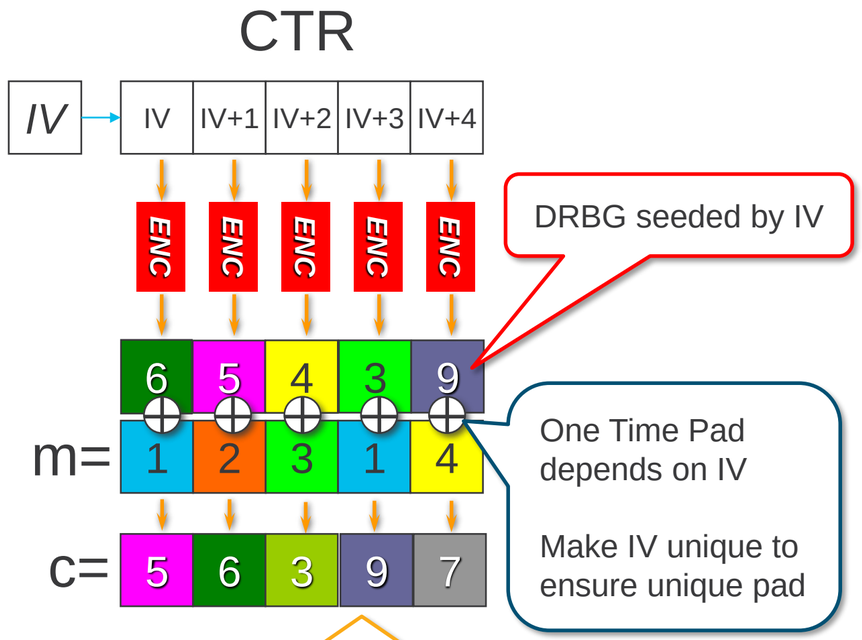


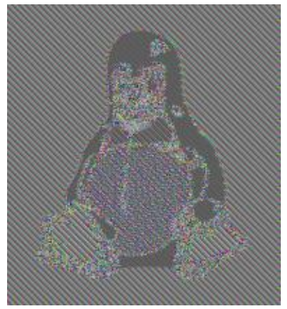
Block Cipher Mode of Operation (ECB, CBC, counter)

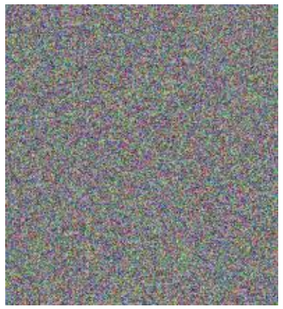












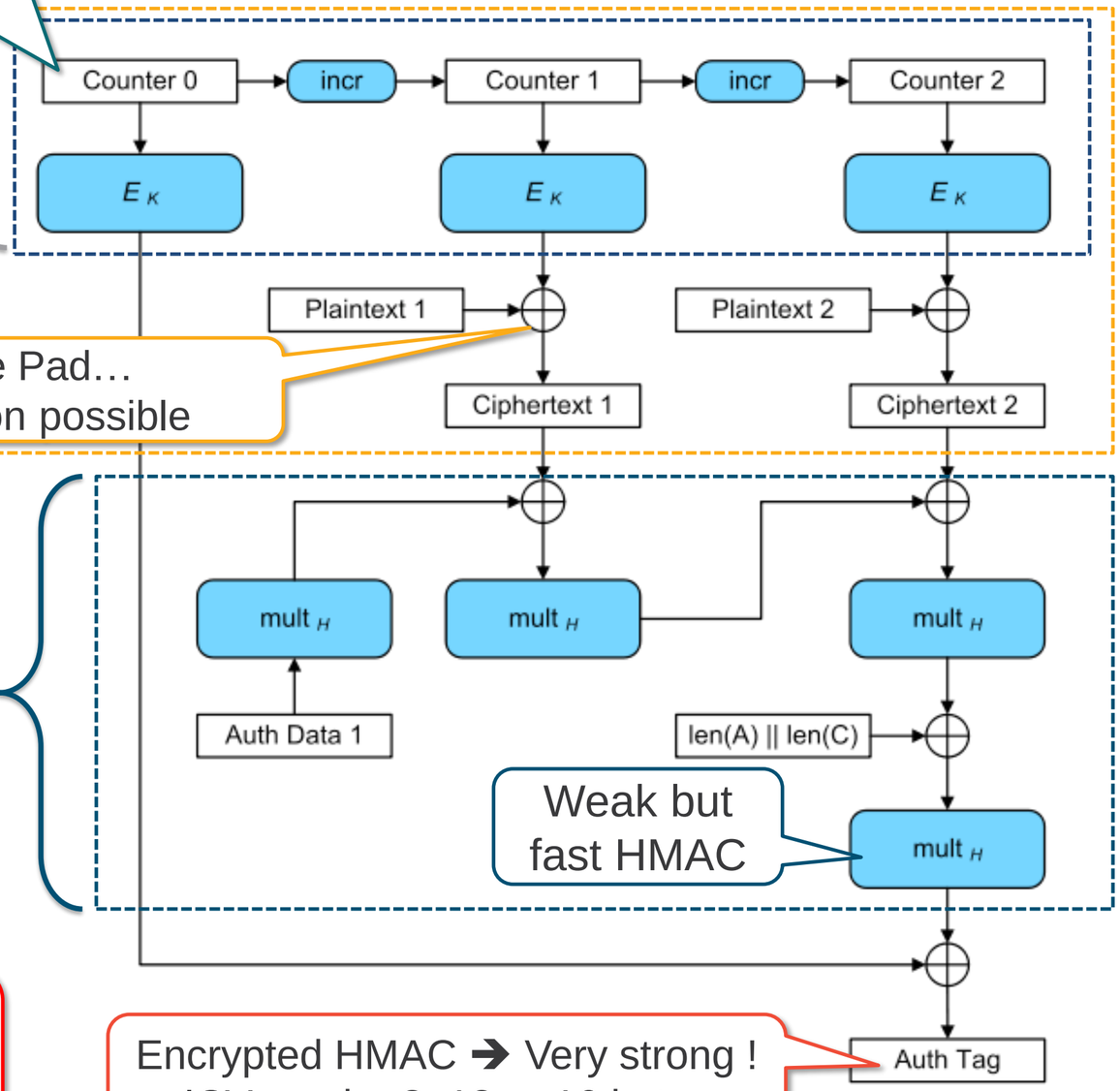
Penguin source: Wikipedia

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

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

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

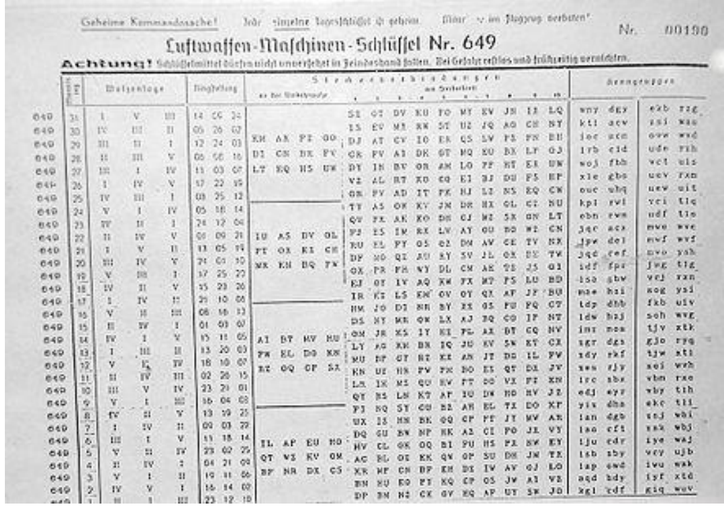
Weak but fast HMAC

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)

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

MODP
Multiplicative Group of Integers Modulo P

The Dawn of Public Key Cryptography


Geheime Kommandosache! Jede einzelne Tagesschlüssel ist geheim. Nur … im Flugzeug verboten!

Nr. 00190

Luftwaffen-Maschinen-Schlüssel Nr. 649

Achtung! Schlüsselmittel dürfen nicht unverschlüsselt in Feindeshand fallen. Bei Gefahr restlos und früchtzeitig vernichten.

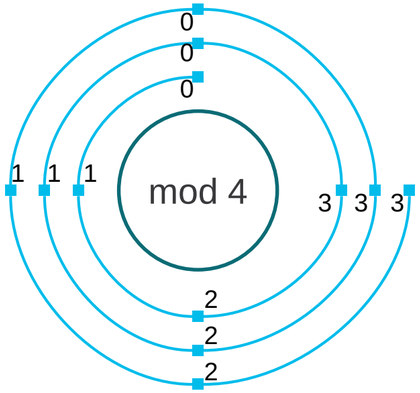
- In the past, (shared) keys had to be exchanged ahead
  - Enigma code books stolen… the rest is history
- Public key cryptography offers a relief
- Multiplicative Group of Integers Modulo P: mod p
- Challenge to the attacker:
  - Computing discrete logarithm in a group. I.e find **k** in $\mathbf{b^k = g \bmod p}$
  - Factor very large numbers (hundreds or thousands of bits)

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 mod 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

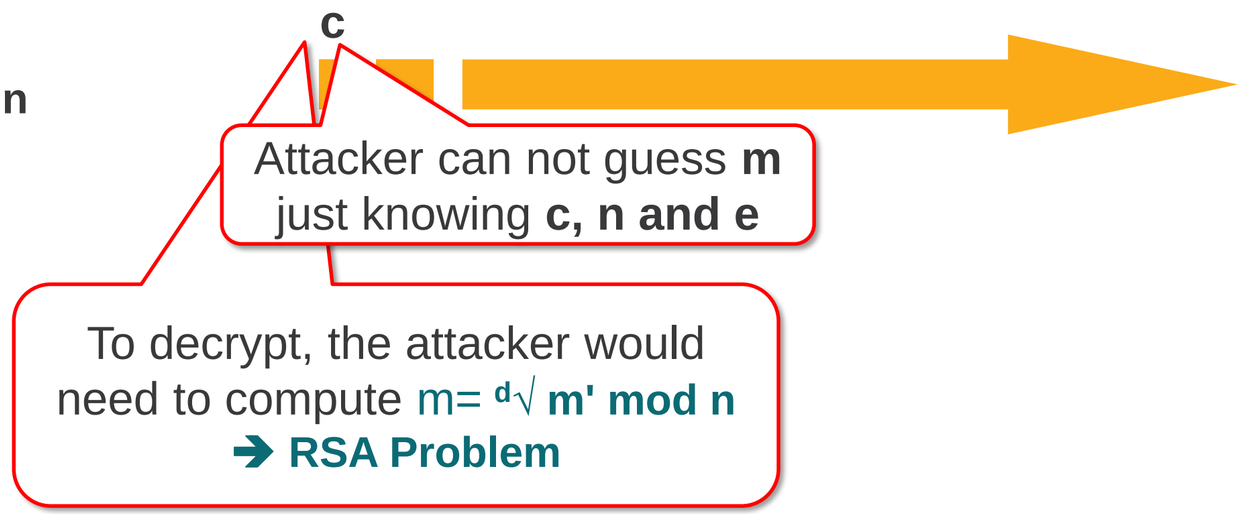
unless there are trapdoors

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**



Bob

Selects three numbers **n, d & e**
n & **e** are **public**, **d** is **secret**
e, d are chosen such as **ed ≡ 1 mod n**

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

m' = c^d mod n
= $(m^e)^d$ mod n
= m^{ed} mod n
= m^1 mod n
= m

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

Signature with Modular Arithmetic

Alice

Bob

Selects three numbers **n, d & e**
n & **e** are **public**, **d** is **secret**
e, d are chosen such as **ed ≡ 1 mod n**

Must send a signed message **m**

Computes **c = m^d mod n**
(we assume m < n)

c, m

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

Computes **m' = c^e mod n**
m' = c^e mod n
= $(m^d)^e$ mod n
= m^{de} mod n
= m^1 mod n
= m mod n
= m

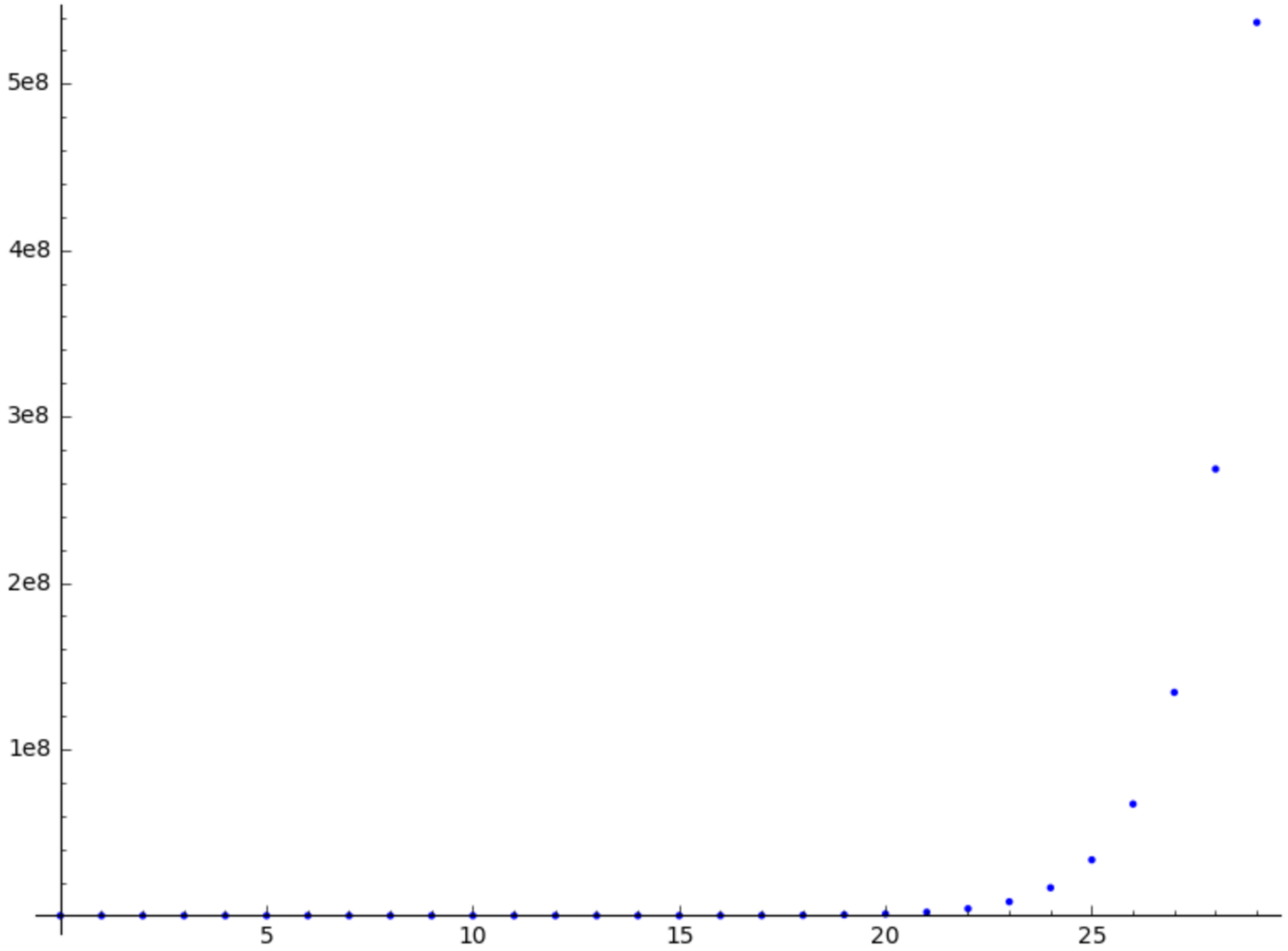
➔Bob must have sent the **c,m**

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

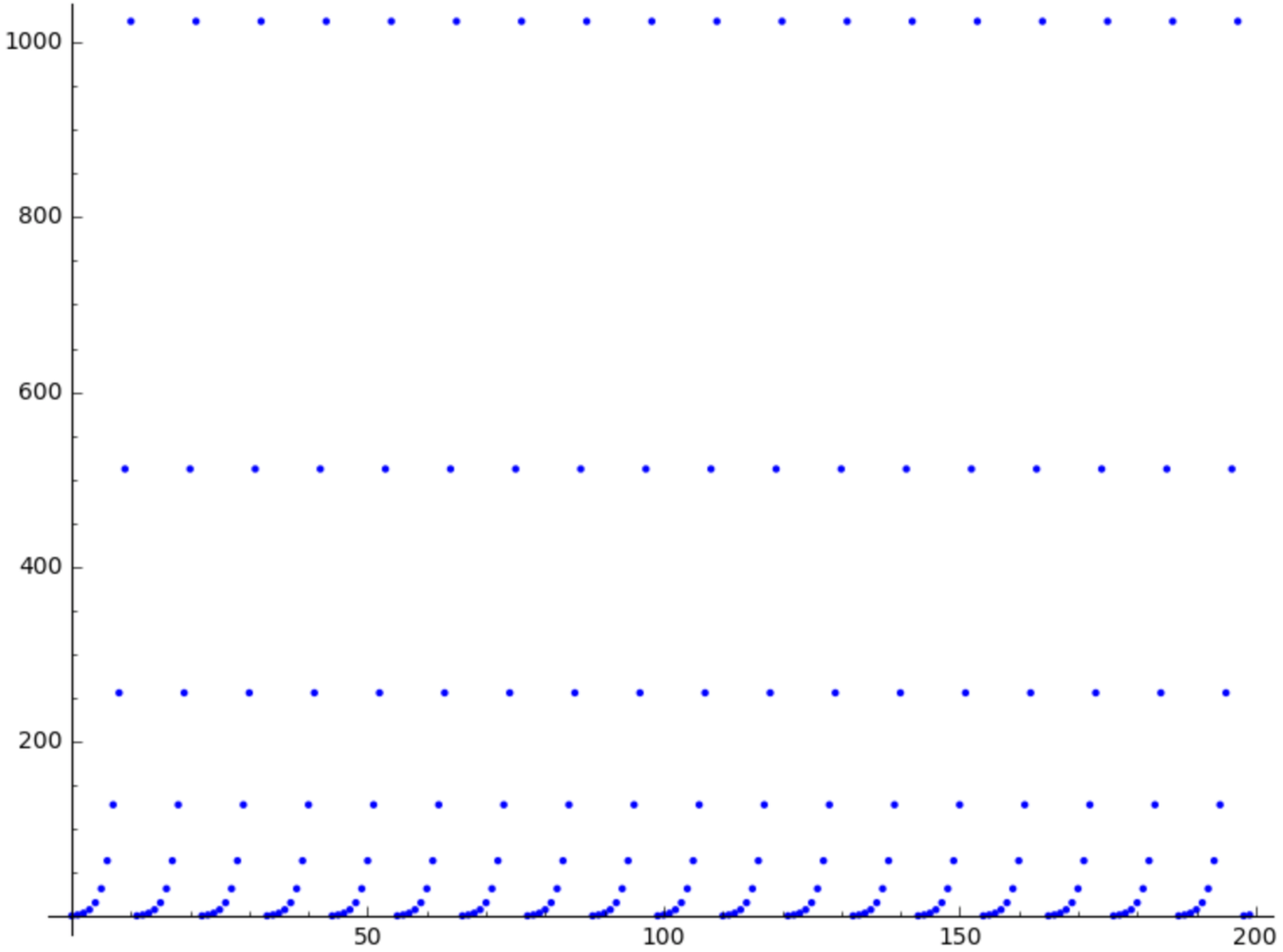
To forge the signature, the attacker would need to compute
d = $\log_e(m')$ mod n
➔ **Discrete Logarithm Problem**

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

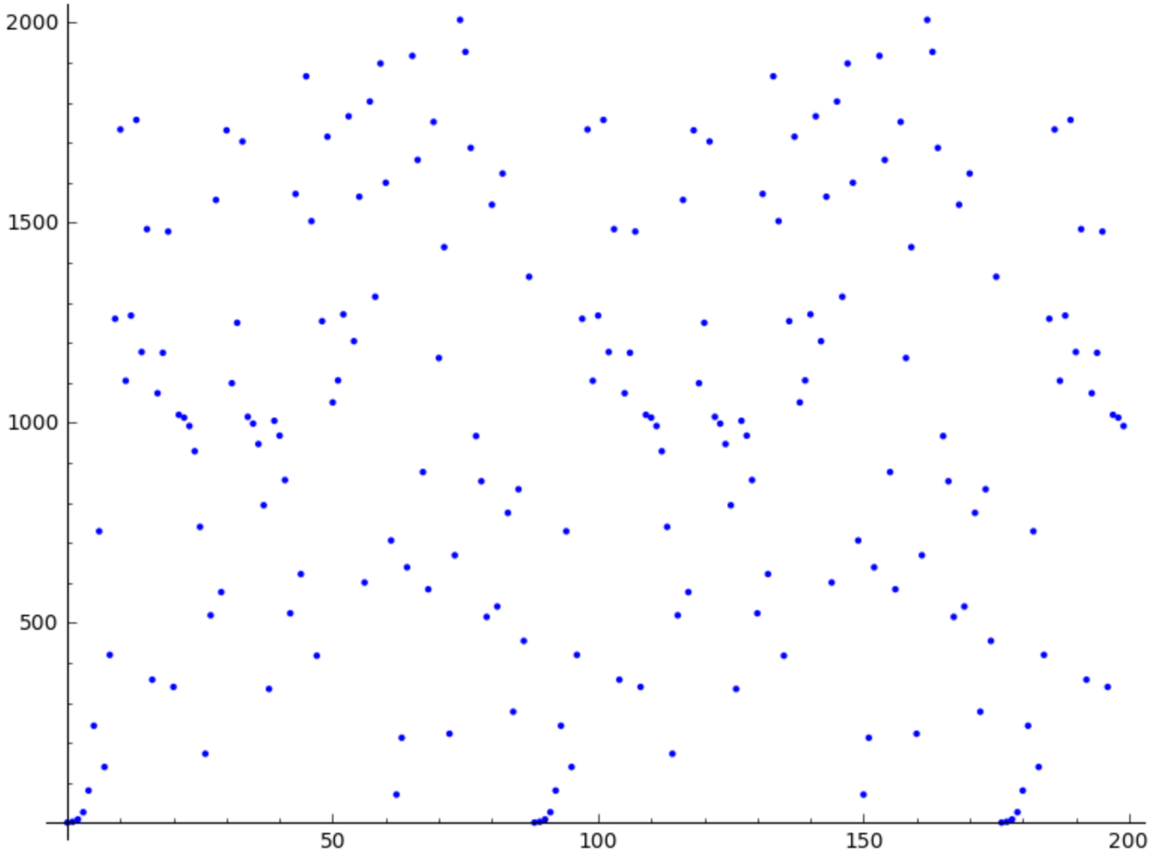
Regular Exponentiation – Use Dichotomy to Reverse

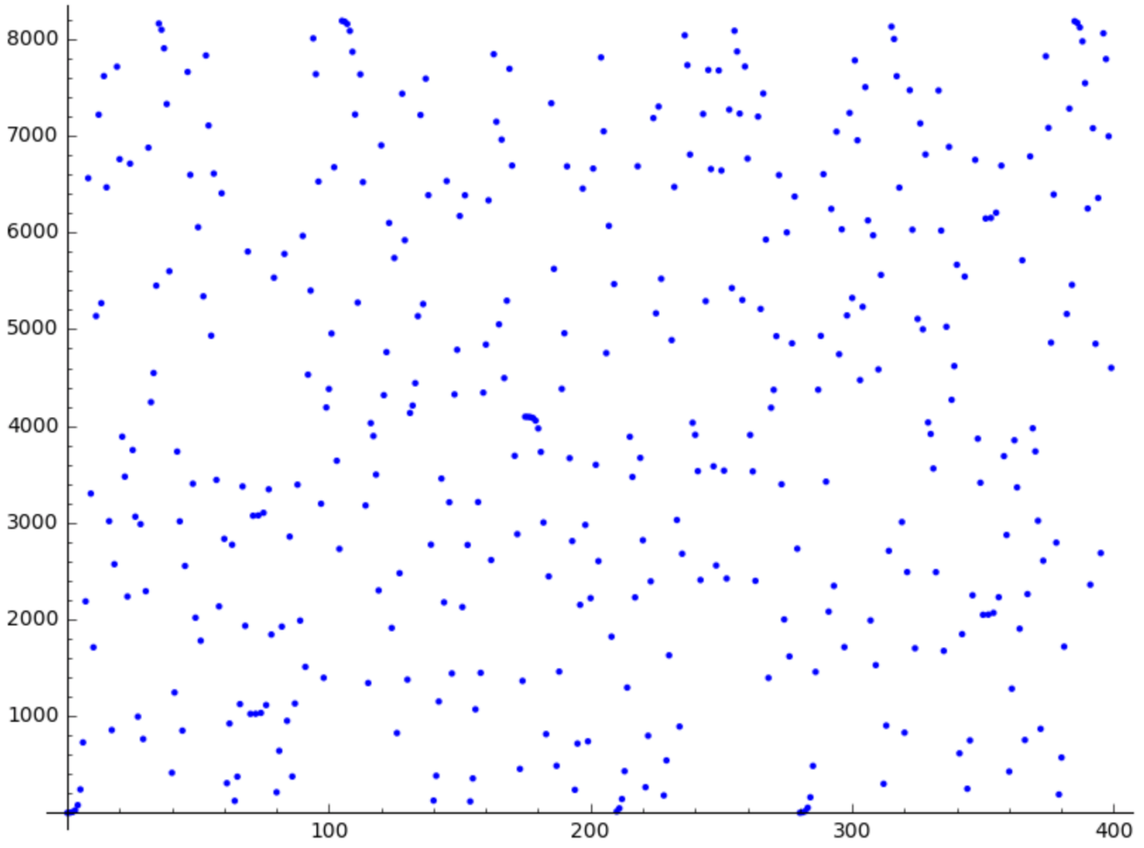


MODP Exponentiation – dichotomy is broken

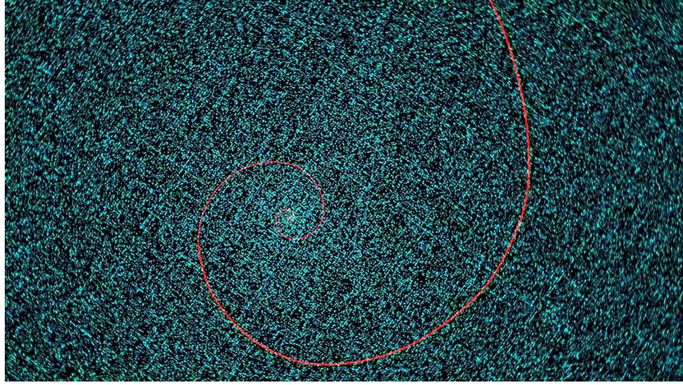


Where Quantum Computers Come In





About Prime Numbers



picture: Khan Academy

- A number is prime if it can be divided by one or itself
- A number is composite is it can be divided by 2 or more prime numbers
  - **Factorization is a hard problem**. Best algorithm yields $O\left(\exp\left(\left(\frac{64}{9}b\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right)$.
- Fundamental Theorem of Arithmetic: a given number has a single factorization
- Euclid's theorems: there are **infinitely many primes**
  - prime density (ratio of primes per composite up to x) is 1/ln(x)
  - density drops off rapidly in the beginning but very slowly after a few powers of 10
  - π(x) = x/ln(x) : number of primes < x
- Euler's φ(n) function or Euleur's totient
  - # of integers in [1,n] that are relatively prime to n: |k ∈ [2,n] | GCD(k,n) = 1|, {1}
    - 2 numbers are coprime if they share no factor other than 1.
  - Property: **n1, n2, GCD(n1,n2)=1 ➔ φ(*n1*n2*)= φ(*n1*)* φ(*n2*)** – Totient is multiplicative
  - **if x is prime ➔ φ(*n*) = n-1** since 1…n-1 coprime with n and n divisible by itself
- Euler's theorem: **$m^{\varphi(n)} \equiv 1 \pmod{n}$ if m and n are co-prime.**

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

- Choose two distinct prime numbers ***p, q*** and hide them forever!
- $n = p.q$ → 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
  - $1^k = 1$ ➔ $(m^{\varphi(n)})^k \equiv 1^k \pmod n$ ➔ $m^{k\varphi(n)} \equiv 1 \pmod n$
  - $1m = m$ ➔ $m\, m^{k\varphi(n)} \equiv m \pmod n$ ➔ $m^{k\varphi(n)+1} \equiv m \pmod n$
  - we look for e,d,n such that $m^{ed} \equiv m^{k\varphi(n)+1} \equiv m \pmod n$ ➔ **$ed = k\,\varphi(n) + 1$**
  - ➔ $\mathbf{d = \frac{k\,\varphi(n)+1}{e} = \frac{k\,(n-(p+q-1))+1}{e}}$
- 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

Euler theorem…

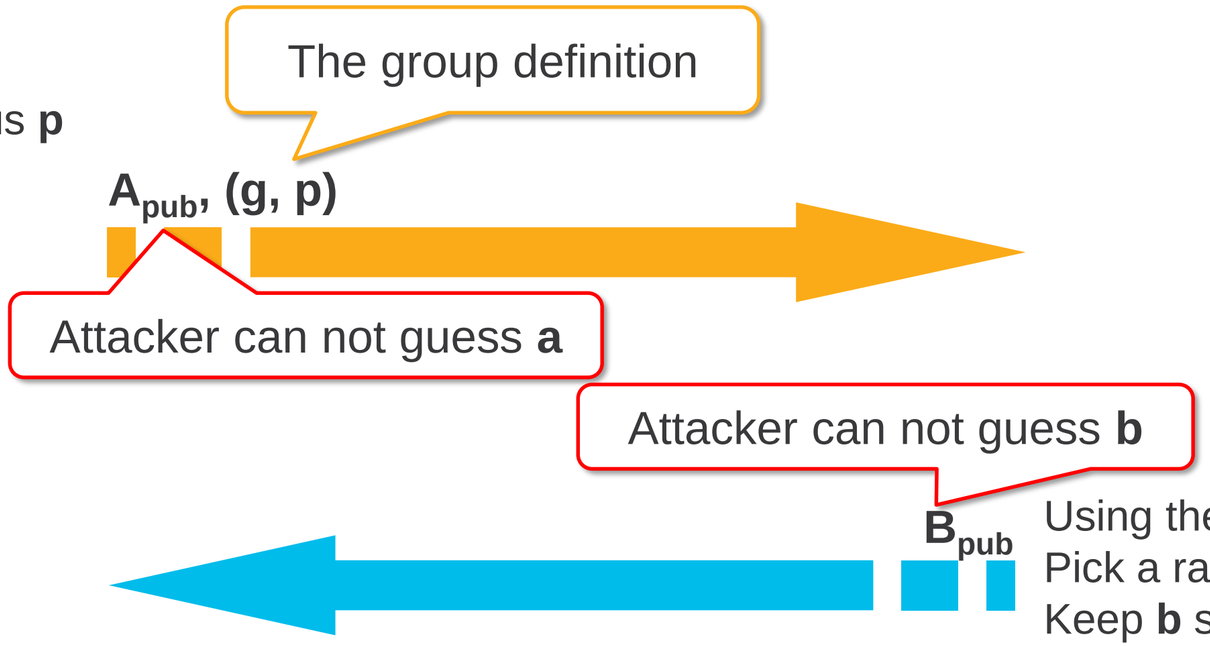
m – arbitrary message
n – the modulus
e – the public key
d – the private key

DH –Diffie-Hellman

Alice

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

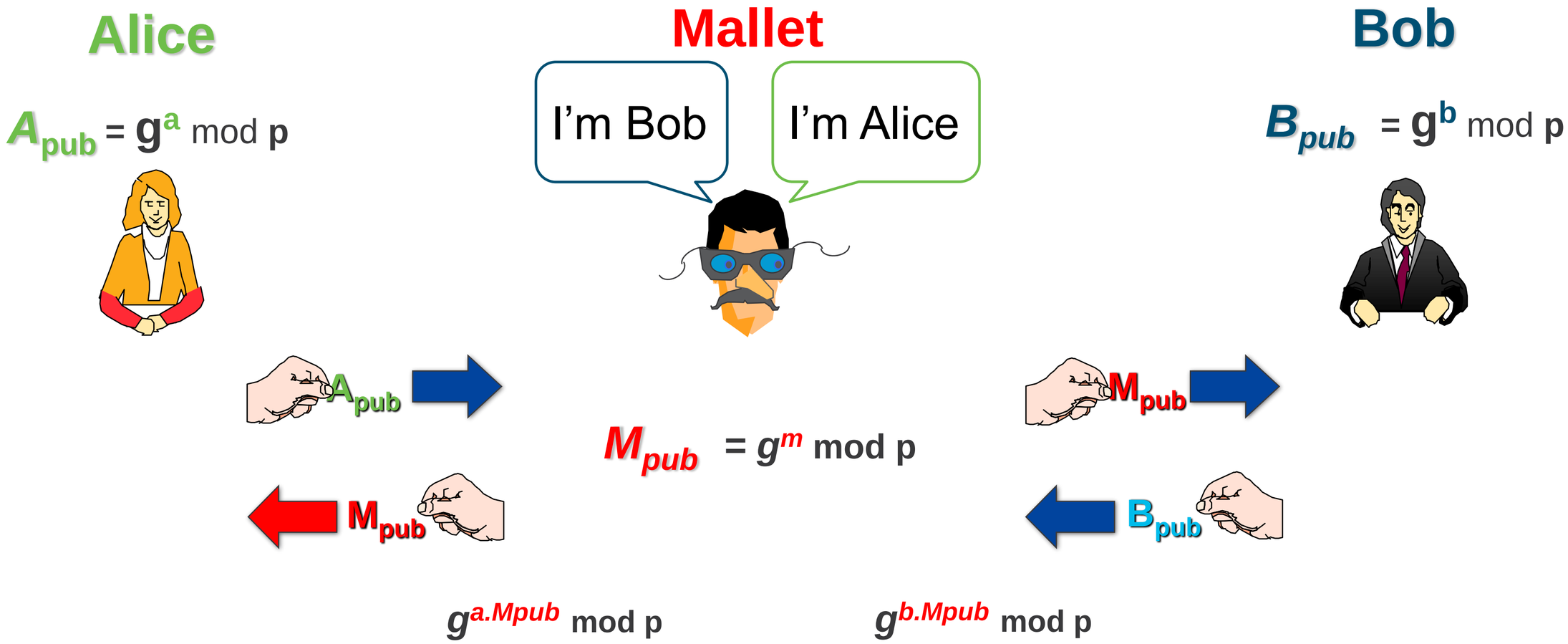
Bob



Using the same generator **g** and modulus **p**
Pick a random number **b**
Keep **b** secret!!
Compute $\mathbf{B_{pub} = g^b \bmod p}$

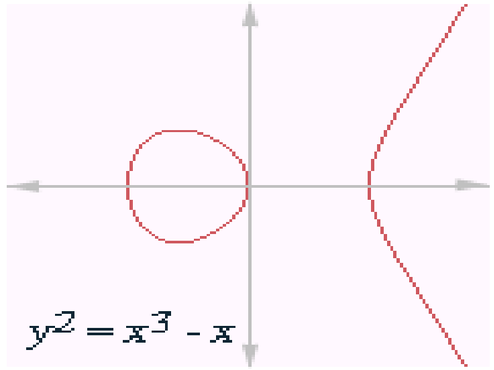
$Secret_{Init} = (B_{pub})^a \bmod p$ = $Secret = g^{a.b} \bmod p$ = $Secret_{Resp} = (A_{pub})^b \bmod p$

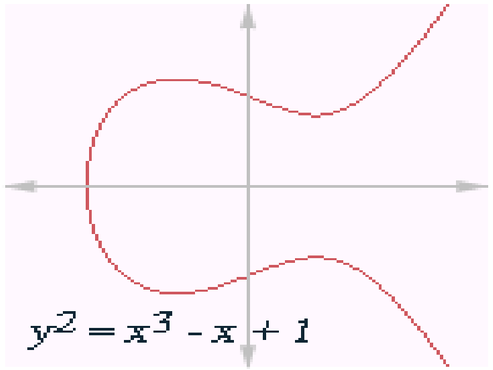
DH is sensitive to a Man-in-the-Middle Attack



ECC
Elliptic Curve Cryptography

What is an elliptic curve ?

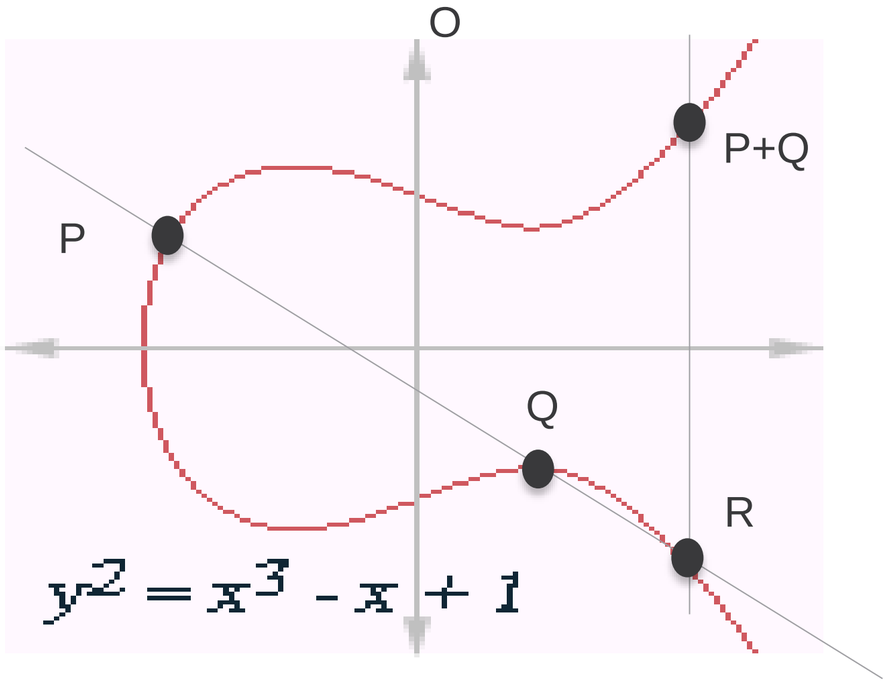


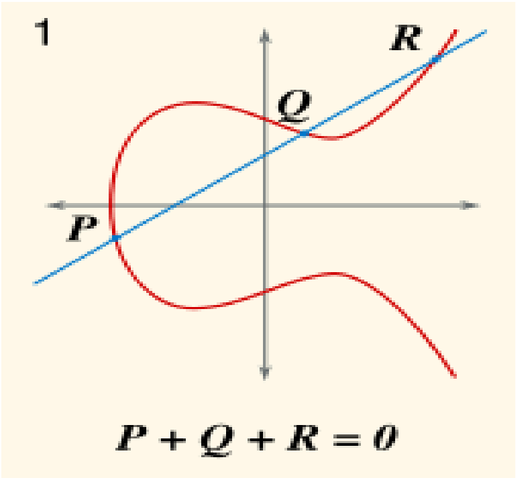


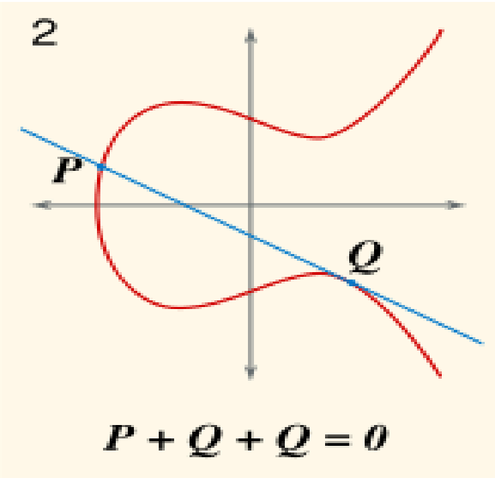
- A curve of general equation **$y^2=x^3+ax+b$**
  - It MUST be a smooth curve
  - Its discriminant MUST BE NON ZERO: $D = 4A^3 + 27B^2$
- **The Elliptic Curve is the set of points**
  - that satisfy the equation of the curve (ie. that “belong” to the curve)
  - Plus special point at infinity that we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

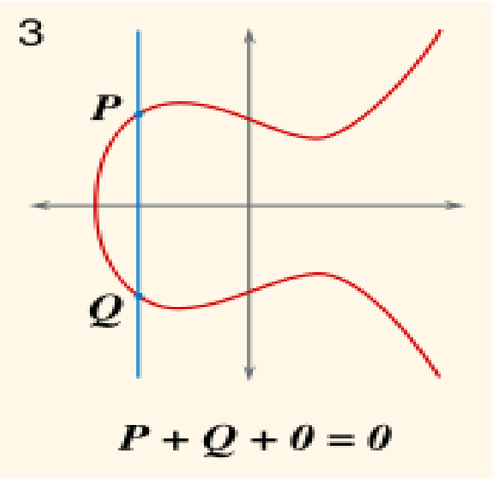
Elliptic Curve Addition

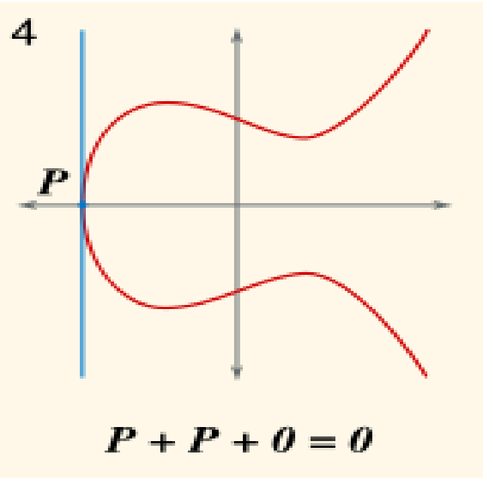
- Let P and Q be two points on the curve
- A line (P,Q) cuts the curve at a third point R
  - If the line is parallel to the Y axis, this point is O
  - If the line is tangent to the curve, the tangent point is counted twice
- The group operator + is defined such as
  - P+Q+R = O; O is the identity
-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



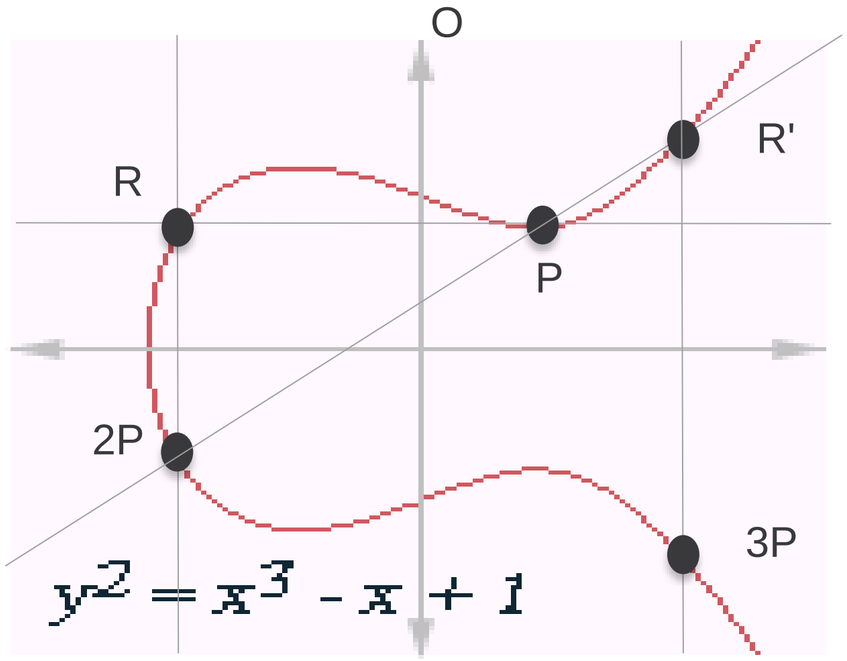






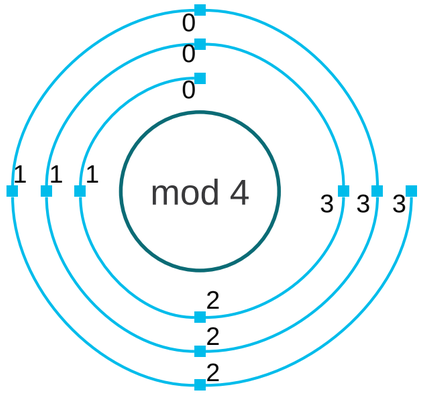


The scalar multiplication n*P



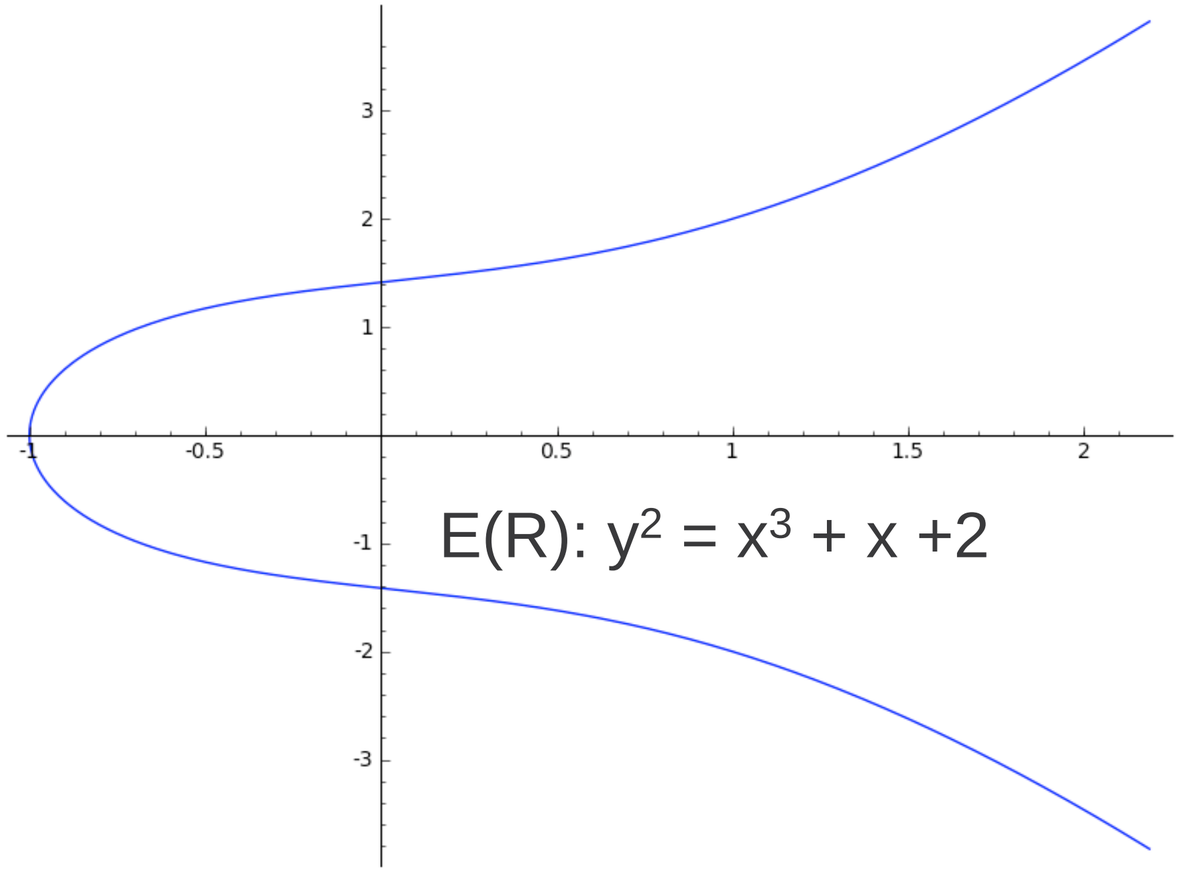
- Let’s start with P+P = 2*P
- For drawing (P,P)
  - draw a tangent to the curve → R
  - (O,R) cuts in P+P=2P
- This is a scalar multiplication
  - One can derive 3P = 2P+P, 4P = 3P+P,…nP = (n-1)P+P

Fast Forward – the finite fields F_m & F_2^k

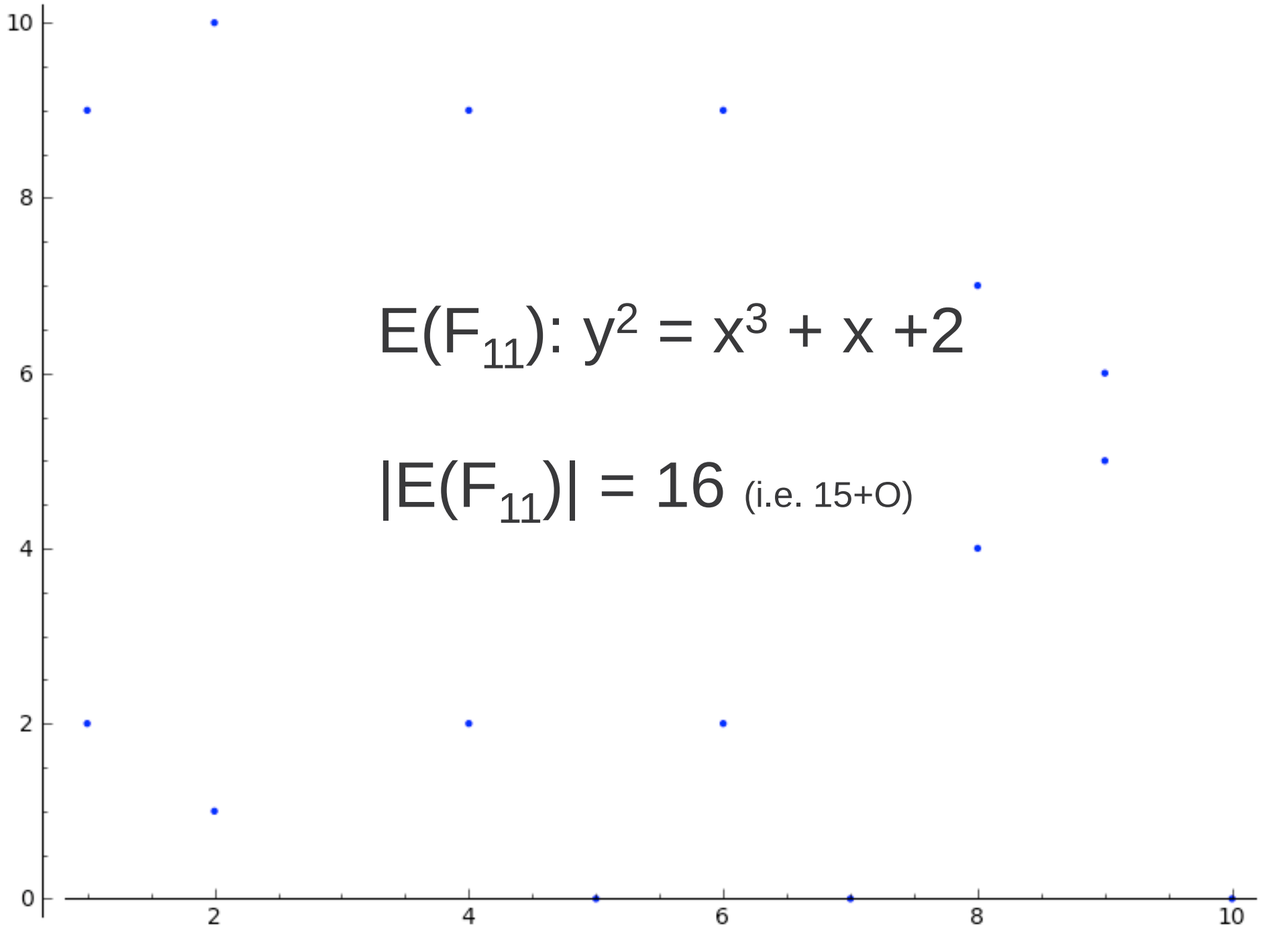


- Remember… modulo arithmetic
- Galois Field = Finite Field
- Let E be an elliptic curve defined over a finite field F_m (modulo m):
  - **$E(Fm):\{\infty\} \cup \{(x,y) \text{ in } F_m x F_m \mid y^2=x^3+ax+b , a,b \text{ in } F_m\}$**
  - $E(F_m)$ is the set of points whose coordinates belong to $F_m x F_m$ 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



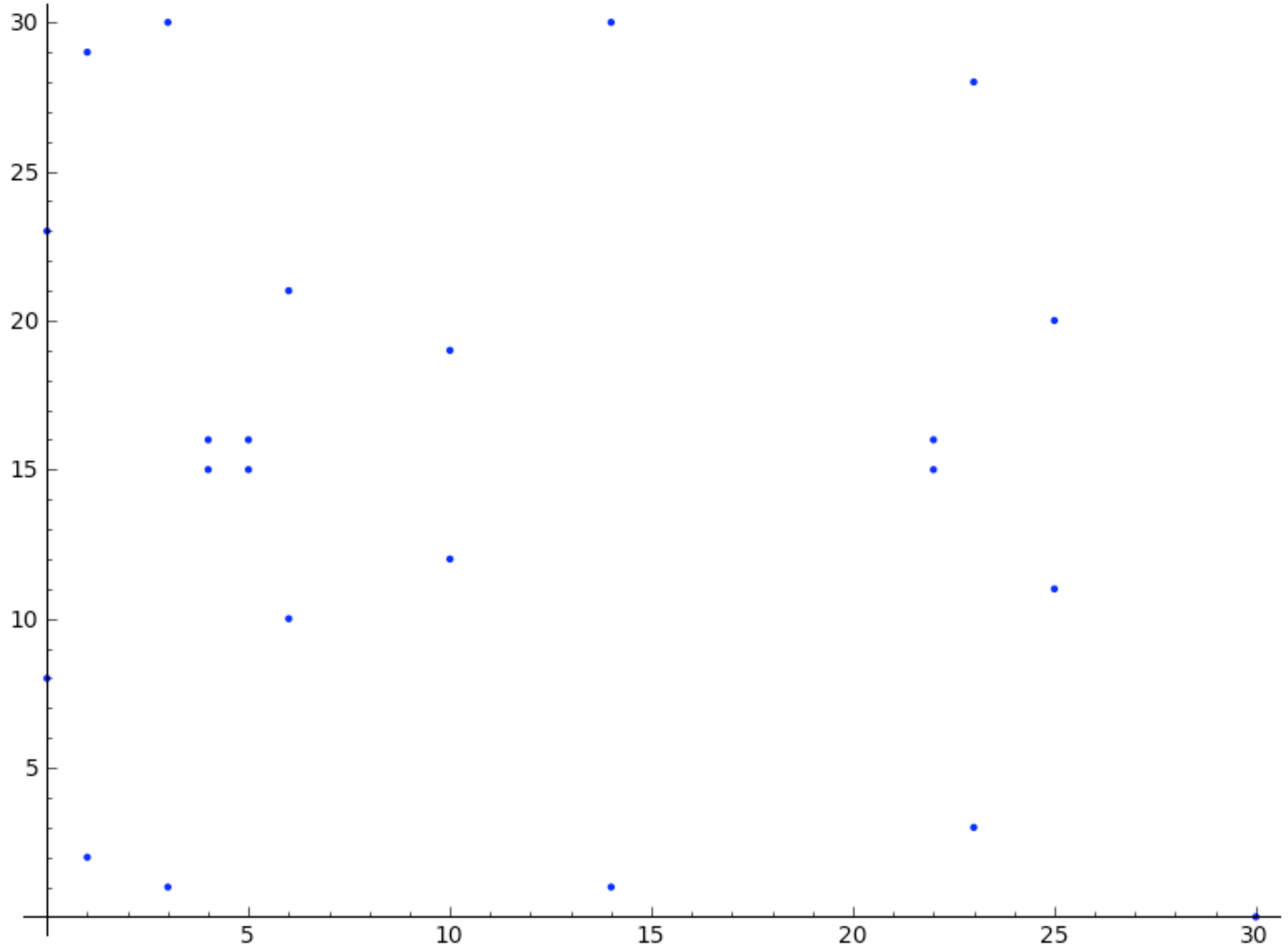
$E(R): y^2 = x^3 + x + 2$



$E(F_{11}): y^2 = x^3 + x + 2$

$|E(F_{11})| = 16$ (i.e. 15+O)

Example on F_{31} – Complexity Increases

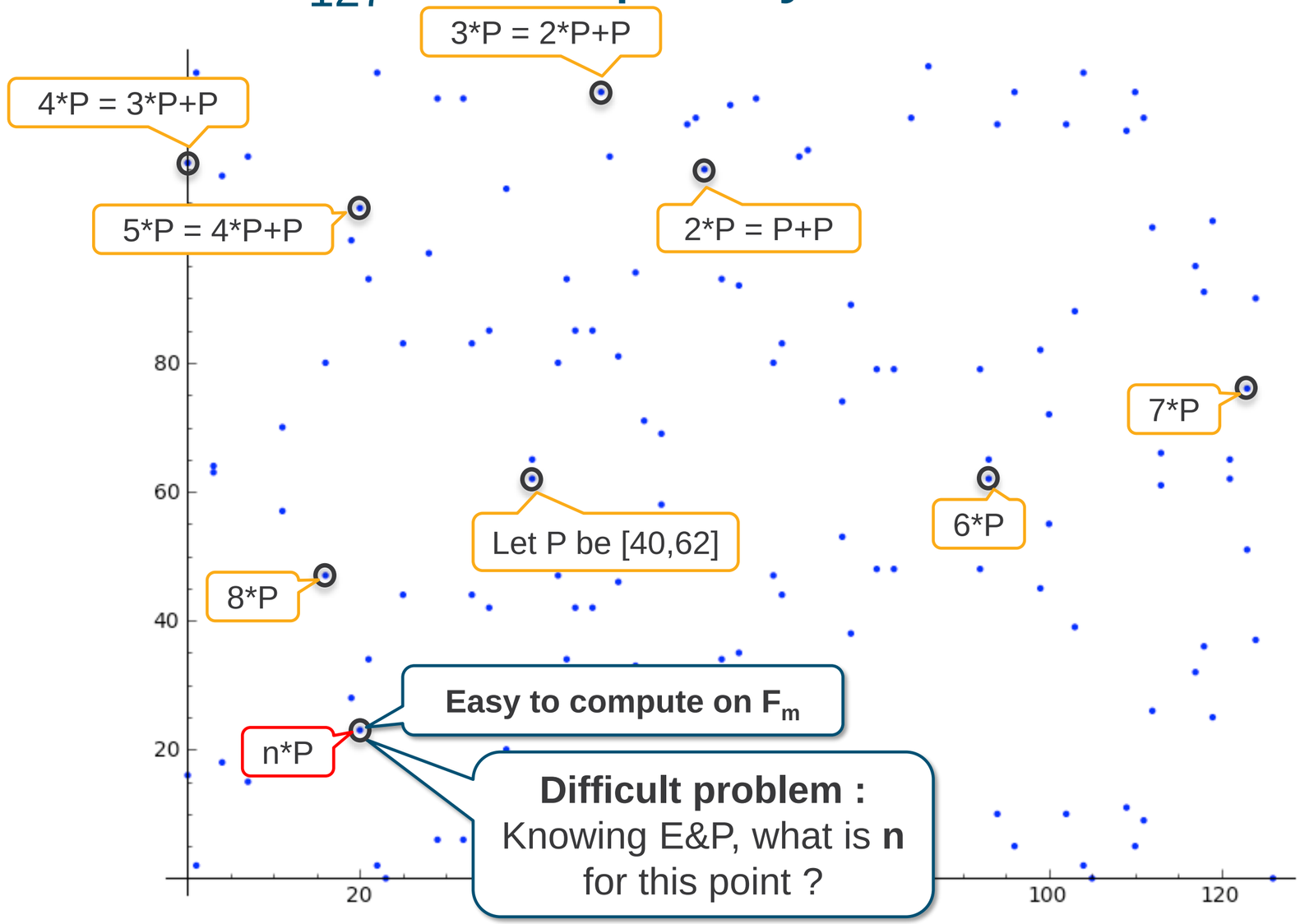


$m = 2^5-1 = 31$

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

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

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

The curve definition **f** and point **P**

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

$\mathbf{A_{pub}}$, **(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 $\mathbf{B_{pub} = b*P}$

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

Representation of Elliptic Curves

- Elliptic curve domain parameters
  - (p, a, b, G, n, h) for a curve over a prime field F_p
  - (m, f(x), a, b, G, n, h) for a curve over a binary field F_2^m
- Where
  - **p** is the prime modulus
  - **G** is the generator (base point) of the curve
  - **n** is the order of G. i.e **n*G=O**
  - **a, b** are the coefficient of $\mathbf{y^2 + xy = x^3 + ax + b \pmod p}$
- Who defines elliptic curves ?
  - National Institute of Standards and Technology (NIST)
  - American National Standard Institute (ANSI)
  - Agence Nationale pour la Securité des Systèmes Informatiques(ANSSI)
  - Institute of Electrical and Electronics Engineers (IEEE)
  - Certicom
  - Brainpool ECC

P-256 from "NIST routines"

4.3 Curve P–256

4.3.1 Parameters

The curve P–256 is given by the following parameters (see also [FIPS186-2]). The prime $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$:

$p_{256} =$ 1157920892103562487626974469494075735300\
86143415290314195533631308867097853951

in hexadecimal form:

$p_{256} =$ ffffffff 00000001 00000000 00000000 00000000 ffffffff\
ffffffff ffffffff

The parameter $a = p_{256} - 3$:

$a =$ 1157920892103562487626974469494075735300\
86143415290314195533631308867097853948

in hexadecimal form:

$a =$ ffffffff 00000001 00000000 00000000 00000000 ffffffff\
ffffffff fffffffc

the parameter b:

$b =$ 41058363725152142129326129780047268409 11\
44410159937255548352563140394674012 91

in hexadecimal form:

$b =$ 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6\
3bce3c3e 27d2604b

Base point G:

$x_G =$ 48439561293906451759052585252797914202 76\
29495260417479958440807170824046352 86

$y_G =$ 36134250956749795798585127919587881956 61\
11066729850150718771982535684144051 09

in hexadecimal form:

$x_G =$ 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0\
f4a13945 d898c296

$y_G =$ 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece\
cbb64068 37bf51f5

in hexadecimal form with X9.63 compression (lead byte 02 if y_G is even, 03 if y_G is odd):

$\overline{G} =$ 00000003 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81\
2deb33a0 f4a13945 d898c296

Order q of the point G (and of the elliptic curve group E):

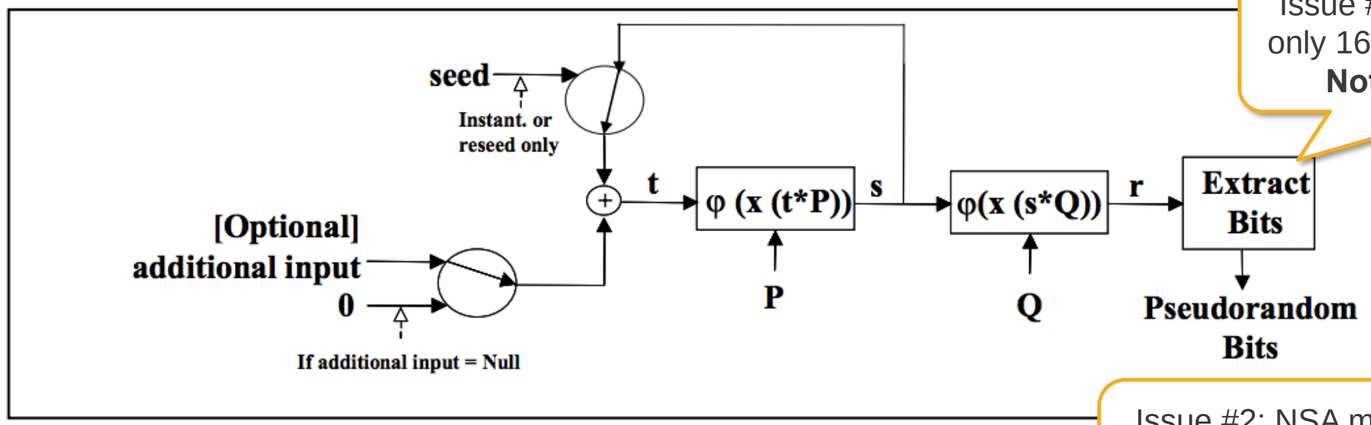
$q =$ 1157920892103562487626974469494075735299\
96955224135760342422259061068512044369

hexadecimal form:

$q =$ ffffffff 00000000 ffffffff ffffffff bce6faad a7179e84\
f3b9cac2 fc632551

A back door'ed PRNG: Dual EC DRBG

source: NIST 800-90A



Issue #1: extract too many bits – only 16 bits to guess leads to s*Q. **Not a problem in itself…**

A.1.1 Curve P-256

p = 115792089210356248762697446949407573530086143415290314195533631308867097853951

n = 115792089210356248762697446949407573529996955224135760342422259061068512044369

b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b

Px = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296

Py = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5

Qx = c97445f4 5cdef9f0 d3e05e1e 585fc297 235b82b5 be8ff3ef ca67c598 52018192

Qy = b28ef557 ba31dfcb dd21ac46 e2a91e3c 304f44cb 87058ada 2cb81515 1e610046

Issue #2: NSA managed to make baked P&Q values a standard. **The last nail in the coffin**

if P=eQ, knowing sQ means sP=seQ ➔ once you know sQ, you know what is going to be the next sP: just compute e*sQ !!

Then you know sQ effortlessly ➔ compute the next sP etc. forever!!

P & Q are supposed to be truly random point but the NSA cheated the process and cooked P & Q to have a relationship P=eQ. Of course, e is super secret and very hard (impossible) to find.

By breaking the PRNG, the attacker can break all other crypto mechanisms. New asymmetric or symmetric keys, DH secrets, pre-master secrets, etc.

NSA did NOT break cryptography. NSA abused a PROCESS !!

Performance and Security Comparisons

Security Level of Symmetric Crypto Algorithms

Security Level	Work Factor	Algorithms
Weak	$O(2^{40})$	DES, MD5
Legacy	$O(2^{64})$	RC4, SHA1
Minimum	$O(2^{80})$	3DES, SEAL, SKIPJACK
Standard	$O(2^{128})$	AES-128, SHA-256, GHASH
High	$O(2^{192})$	AES-192, SHA-384, GHASH
Ultra	$O(2^{256})$	AES-256, SHA-512, GHASH

Quantum Strength (for comparison)

Algorithm	Key Length	Classical Bit Strength	Quantum Bit Strength
RSA/DH 1024	1024 bits	80 bits	0 bits
RSA/DH 2048	2048 bits	112 bits	0 bits
ECC/ECDH 256	256 bits	128 bits	0 bits
ECC/ECDH 521	521 bits	256 bits	0 bits
AES 128	128 bits	128 bits	64 bits
AES 256	256 bits	256 bits	128 bits
SHA 256	256 bits	256 bits	128 bits

ECDH Gains in Security

The Table Below Shows the Comparable Key Lengths Required in DH/RSA as Compared to ECC Based DH to Secure a Symmetric Key of a Given Length

Symmetric Key Length	ECC Key Length	DH/RSA Key Length
80	163	1024
112	233	2048
128	283	3072
192	409	7680
256	571	15360

Reference: draft-ietf-ipsec-ike-ecc-groups-05.txt with Further Reference Contained Therein

IOS IKEv2 New Smart Defaults

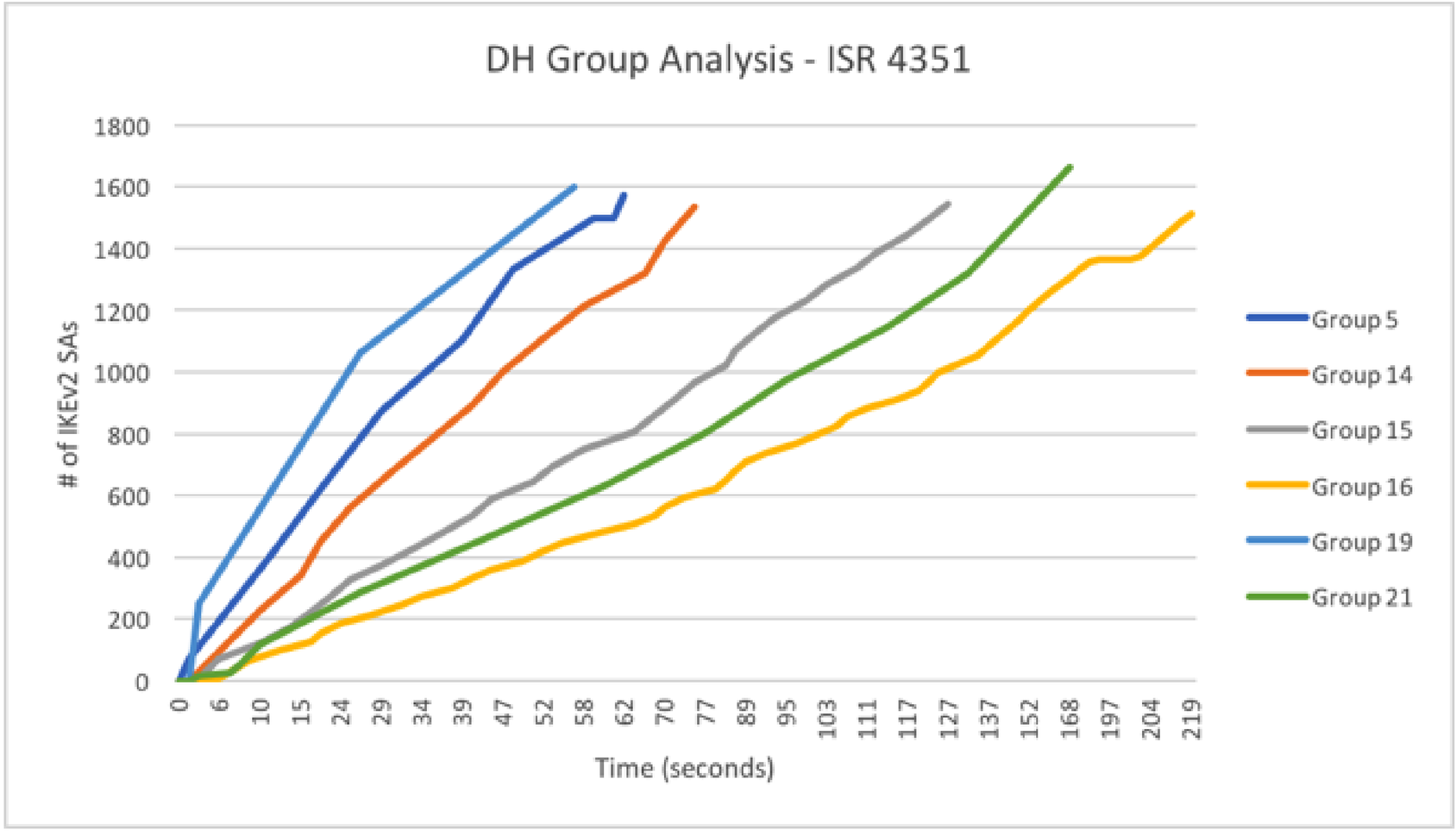
```
Router#show crypto ikev2 proposal default
IKEv2 proposal: default
Encryption : AES-CBC-256 AES-CBC-192 AES-CBC-128
Integrity  : SHA512 SHA384 SHA256 SHA96 MD596
PRF        : SHA512 SHA384 SHA256 SHA1 MD5
DH Group   : DH_GROUP_1536_MODP/Group 5 DH_GROUP_1024_MODP/Group 2
```

Today

```
Router#show crypto ikev2 proposal default
Encryption : AES-CBC-256
Integrity  : SHA512 SHA384
PRF        : SHA512 SHA384
DH Group   : DH_GROUP_256_ECP/Group 19 DH_GROUP_2048_MODP/Group 14
DH_GROUP_521_ECP/Group 21 DH_GROUP_1536_MODP/Group 5
```

CSCuy44786
16.8.1

IOS IKEv2 Smart Defaults Performance



Rough Performance Comparison

Very rough comparison – orders of magnitude only (*)

Strength	AES-CBC			DES / 3DES CBC			RSA			ECDSA		
	Key size	16 B	8192 B	Key size	16 B	8192 B	Modulus	Sign	Verify	Field	Sign	Verify
< 80 bits	-	-	-	56 (DES)	64 MBps	66 MBps	512 bits	18K /s	194K /s	-	-	-
80 bits	-	-	-	168 (3DES)	25MBps	25MBps	1024 bits	6.5K /s	90K /s	p160	13K /s	4K /s
112 bits	-	-	-	-	-	-	2048 bits	1.5K /s	30K /s	p256	20K /s	8.5K /s
128 bits	128	116MBps	130MBps	-	-	-	4096 bits	136 /s	8.5K /s	k283	1.6K /s	1K /s
192 bits	192	98.5 MBps	109 MBps	-	-	-	7680 bits	-	-	k409	723 /s	568 /s
256 bits	256	85 MBps	94 MBps	-	-	-	15360 bits	-	-	k571	348 /s	249 /s

Optimizations...

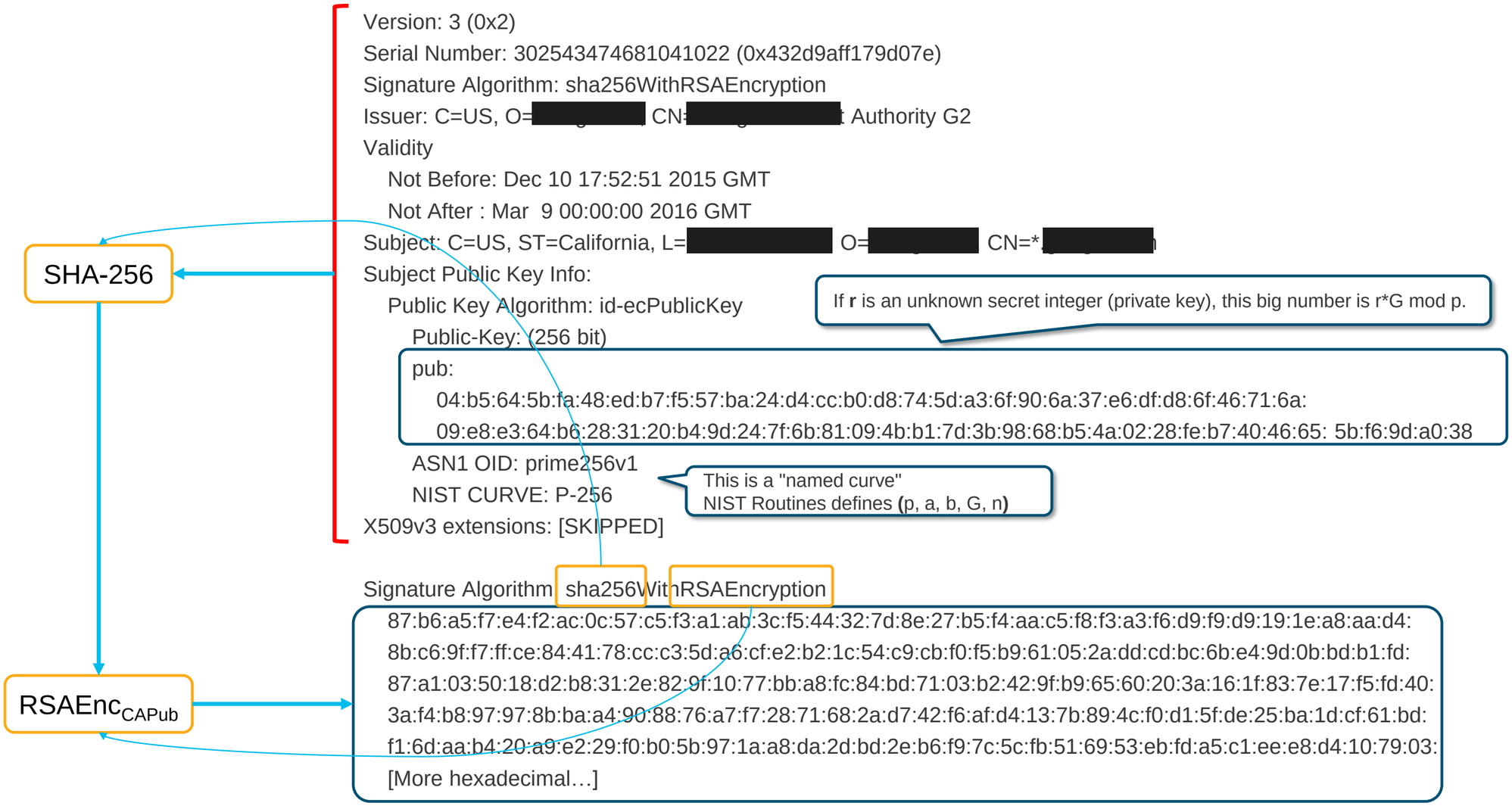
(*) computed on my laptop – whatever that means

See Performances for Yourself 😀

- openssl speed aes
- openssl speed des-cbc
- openssl speed des-ede3 ← 3-DES (Encrypt-Decrypt-Encrypt)
- openssl speed rsa
- openssl speed ecdsa
- openssl speed dh ← does not exist
- openssl speed ecdh

Practical Use...
The Crypto Angle

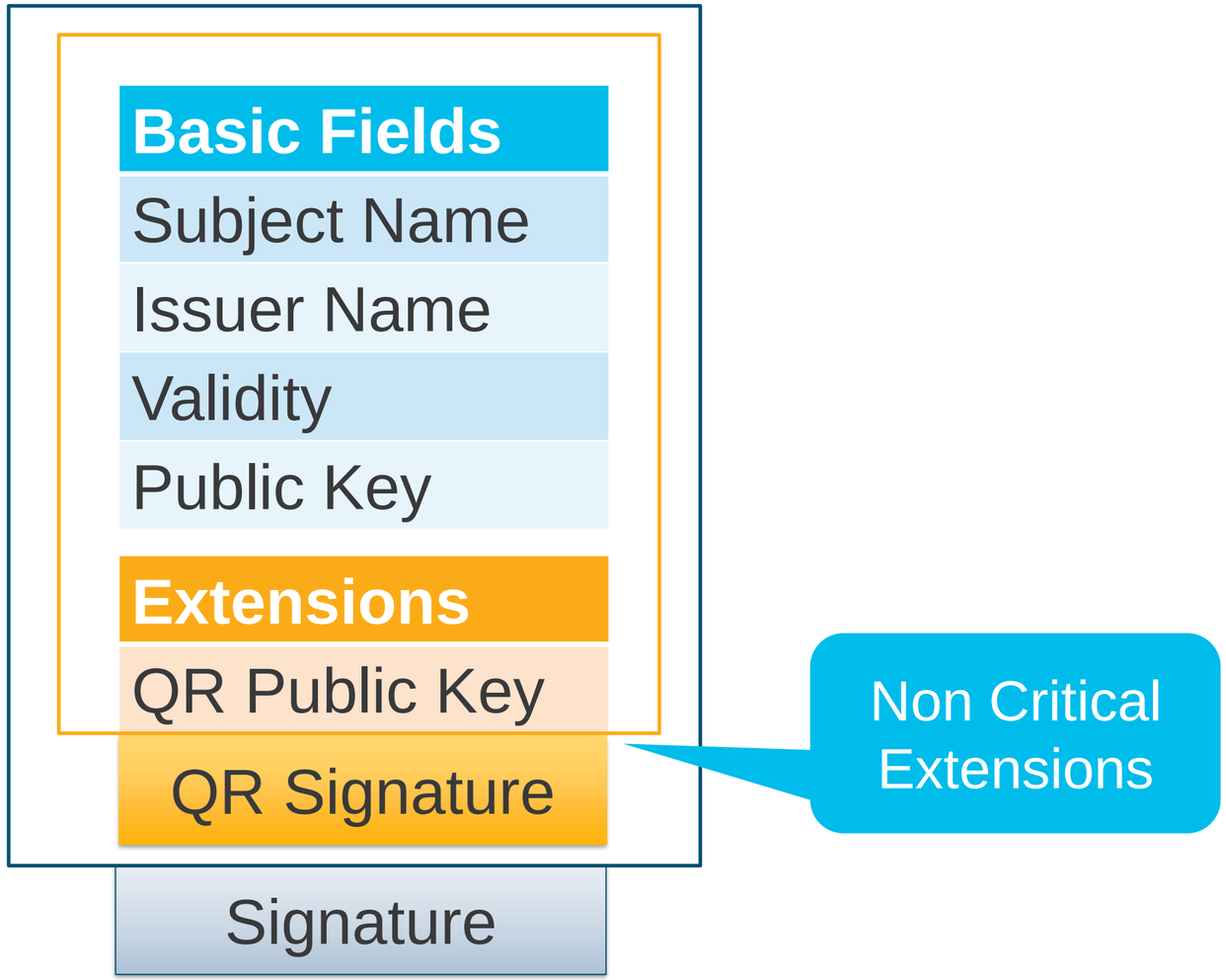
Certificates – Just an example…



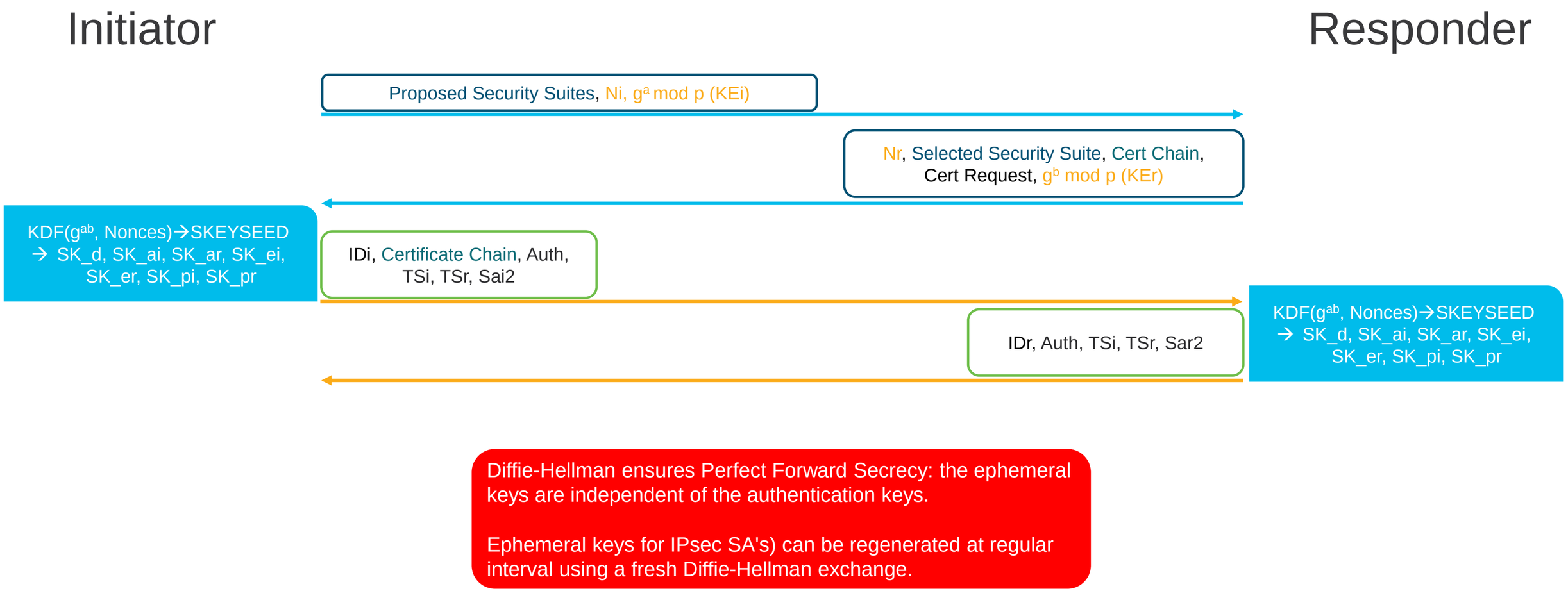
Source: The Viability of Post-QuantumX.509 Certificates

Panos Kampanakis
Peter Panburana
Ellie Daw
Daniel Van Geest

Quantum Resistant Hybrid Certificate



IKEv2



Diffie-Hellman ensures Perfect Forward Secrecy: the ephemeral keys are independent of the authentication keys.

Ephemeral keys for IPsec SA's) can be regenerated at regular interval using a fresh Diffie-Hellman exchange.

Hash DRBG for Key Derivation and Authentication

Hash DRBG prf+:

prf+ (K,S) = T1 | T2 | T3 | T4 | ...
where:
T1 = prf (K, S | 0x01)
T2 = prf (K, T1 | S | 0x02)
T3 = prf (K, T2 | S | 0x03)
T4 = prf (K, T3 | S | 0x04)
...

Key Generation:

Key Derivation Function

SKEYSEED = prf (Ni | Nr, g^ir)
{SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr} = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)

Authentication:

RSA_{Priv} prf

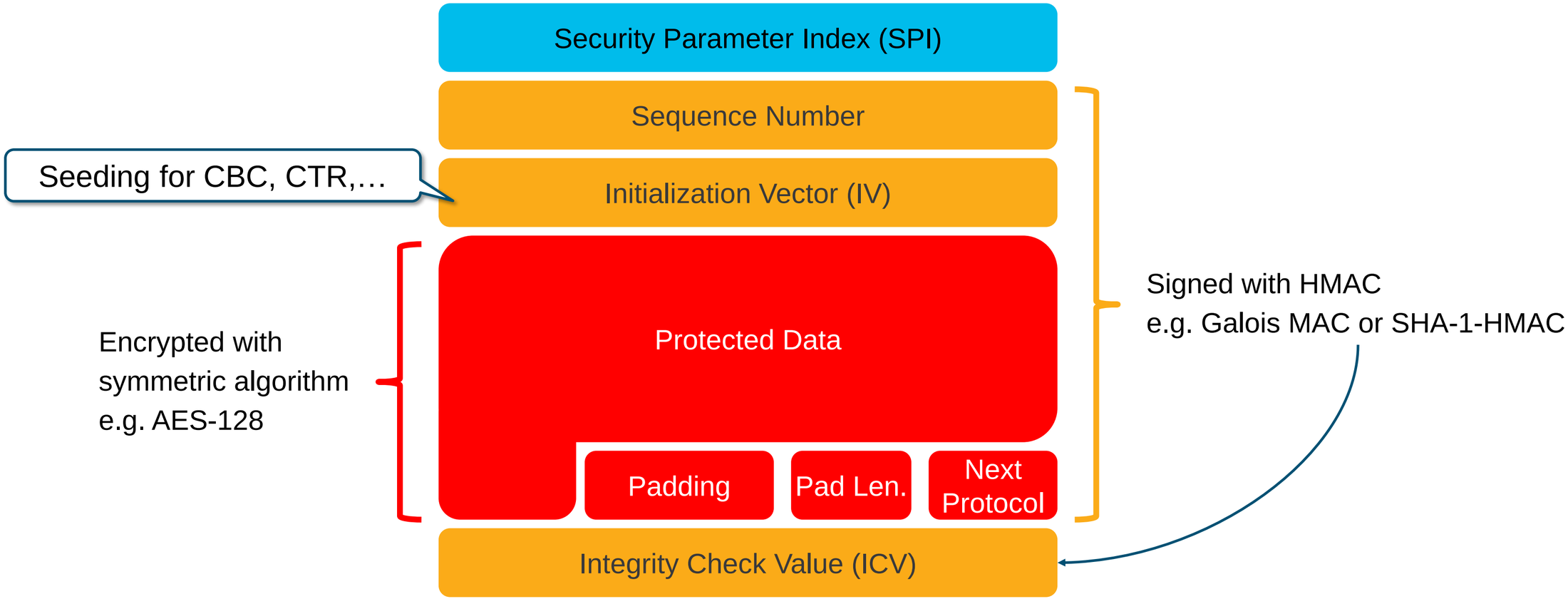
Let's call all this the "conversation"

InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [four octets 0 if using port 4500] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)

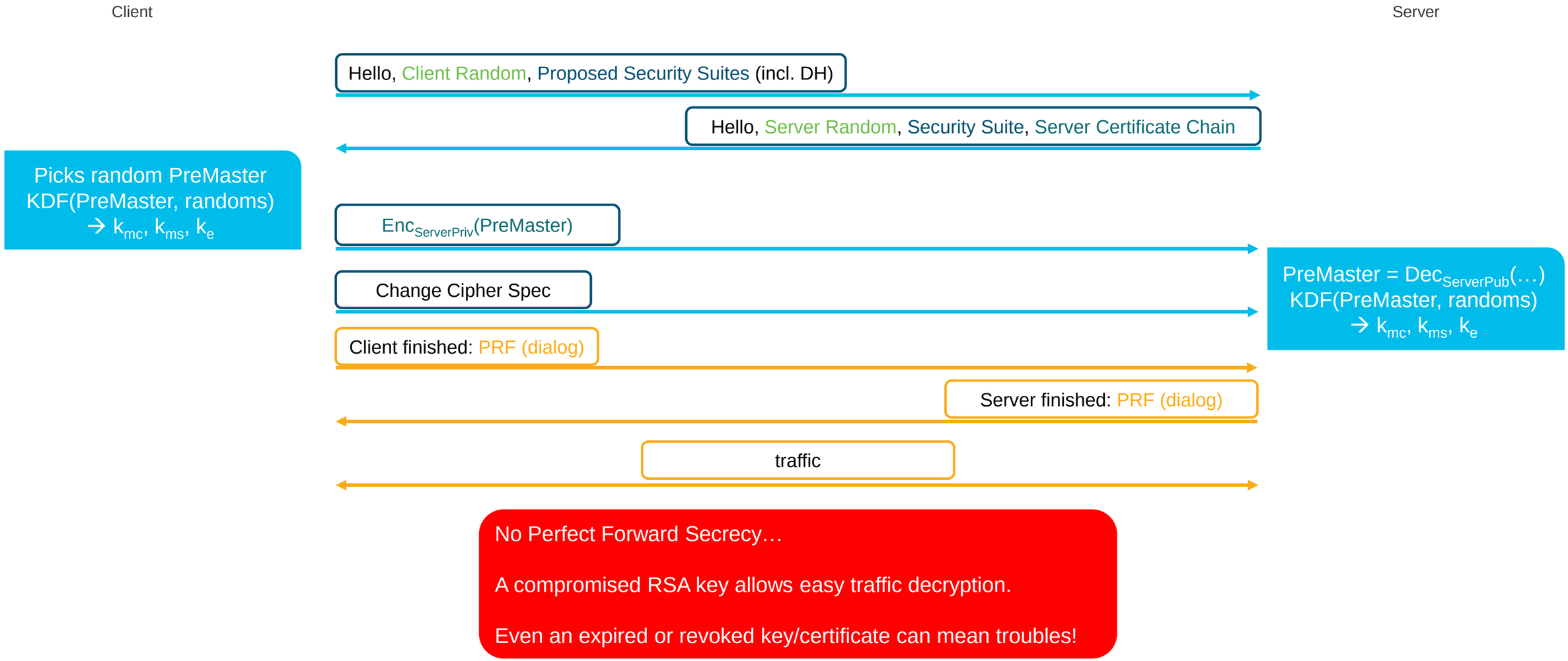
IPsec: ESP packet format

IPsec HMAC and Encryption keys independent of IKE sessions keys.

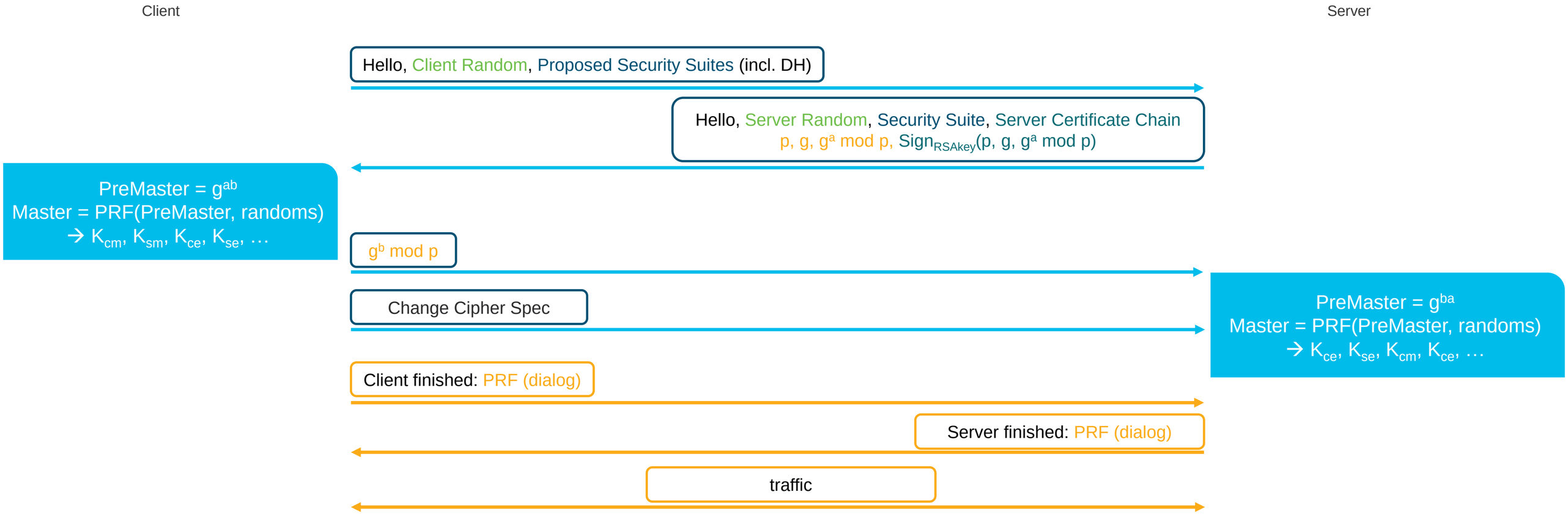
IPsec keys can be derived from a fresh DH exchange (aka PFS)



SSL/TLS with Pre-Master Secret (no DH)



SSL/TLS with Ephemeral Diffie-Hellman



Offers Perfect Forward Secrecy since SSLv3.

In everyday practice many implementations have refused to offer forward secrecy or only provide it with very low encryption grade. [Wikipedia]

While TLS offers and recommends regular rekey, tickets span for the lifetime of the application [Blackhat.com US-13-Daigniere-TLS-Secrets-Slides]

TLS Key Derivation and Authentication

Hash DRBG prf:

P_hash(secret, seed) = HMAC_hash (secret, A(1) + seed) +
HMAC_hash (secret, A(2) + seed) +…

A() is defined as

A(0) = seed

A(i) = HMAC_hash(secret, A(i-1))

PRF(secret, label, seed) = P_<hash>(secret, label + seed)

Key Derivation Function

Key Generation:

key_block = PRF(master_secret, "key expansion", server_random, client_random)

{K_{cm}, K_{sm}, K_{ce}, K_{se},…} = key_block

Server Authentication:

RSA_{Priv} (prf (client_random[32] | server_random[32] | server_DH_parms))

the conversation

Recommendations and Conclusion

Reassurance

- Crypto is not broken but it has to be used intelligently
  - Post quantum cryptography
  - Supersingular isogeny key exchange
  - Learning With Errors (LWE) challenge
  - Lattice Based Cryptography
- All the problems cited were known or expected
- Snowden's revelations only showed how dedicated attackers are
- We should not be scared, we should simply act!

Recommendations

- Protocols
  - IKEv2 is cool – use it if you can (not always possible)
  - Keep an eye on TLS1.3 for improvements
- Public Key Infrastructure (PKI)
  - Prefer ECDSA (> 256) or RSA (modulus size >> 1024 ; 1536 or 2048 preferred)
  - SHA-256-HMAC or better is a must
- Key Exchange
  - **Use PFS**
  - Prefer ECDH 263 bits for mid term security (~15 years) or MODP 3184 bits for 15+ years
  - If MODP, use DH group >> 1024 (1536 or 2048 preferred).
    - IKEv2: group 5 (~1500 bits) or better
    - TLS 1.2: FIX YOUR SERVERS!! https://weakdh.org/sysadmin.html
    - Upgrade to TLS 1.3 whenever possible (still draft)
- Symmetric key cryptography
  - AES-128 (or better) CBC or Counter mode
  - SHA-1 is still ok for data crypto but plan moving to GCM, GMAC or SHA-2 (256 or above)

Reputedly Safe Elliptic Curves

		Parameters:			ECDLP security:				ECC security:			
Curve	Safe?	field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True✔	True✔	True✔	True✔	False	False	True✔	False	False	False	False
M-221	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
E-222	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
NIST P-224	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False	False
Curve1174	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Curve25519	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
BN(2,254)	False	True✔	True✔	True✔	True✔	False	False	True✔	False	False	False	False
brainpoolP256t1	False	True✔	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False
ANSSI FRP256v1	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False	False
NIST P-256	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	True✔	False	False
secp256k1	False	True✔	True✔	True✔	True✔	True✔	False	True✔	False	True✔	False	False
E-382	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
M-383	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Curve383187	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
brainpoolP384t1	False	True✔	True✔	True✔	True✔	True✔	True✔	True✔	False	True✔	False	False
NIST P-384	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	True✔	False	False
Curve41417	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Ed448-Goldilocks	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
M-511	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
E-521	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔

source: http://safecurves.cr.yp.to

Some Random Software… FlexVPN (IOS)

Keypair for certificates

crypto key generate ec keysize 256

crypto key generate rsa 1536 (or better)

IKEv2 Profile

crypto ikev2 proposal default

encryption aes-256 aes-cbc-192 aes-cbc-128

integrity sha512 sha384 sha256

prf sha512 sha384 sha256

group 14 5

Defaults are Top!

show crypto ikev2 proposal

IKEv2 proposal: default

Encryption : AES-CBC-256
Integrity : SHA512 SHA384
PRF : SHA512 SHA384
DH Group : **DH_GROUP_256_ECP/Group 19** DH_GROUP_2048_MODP/Group 14 DH_GROUP_521_ECP/Group 21 DH_GROUP_1536_MODP/Group 5

Some Random Software… WSA

Hear it from the horse's mouth

See BRKSEC-3006 – Tobias Mayer on TLS Decryption using the Web Security Appliance

Some Random Software… OpenSSH

/etc/ssh/ssh_config

```
KexAlgorithms curve25519-sha256@libssh.org
```

Generate your own DH

```
ssh-keygen -G moduli-2048.candidates -b 2048
ssh-keygen -T moduli-2048 -f moduli-2048.candidates
```

Some Random Software… Apache (mod_ssl)

/etc/dovecot.conf

```
SSLProtocol all -SSLv2 -SSLv3

SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

SSLHonorCipherOrder on
```

Point to custom DH parameters

```
SSLOpenSSLConfCmd DHParameters "{path to dhparams.pem}"
```

openssl dhparam -out dhparams.pem 2048

Some Random Software… Dovecot

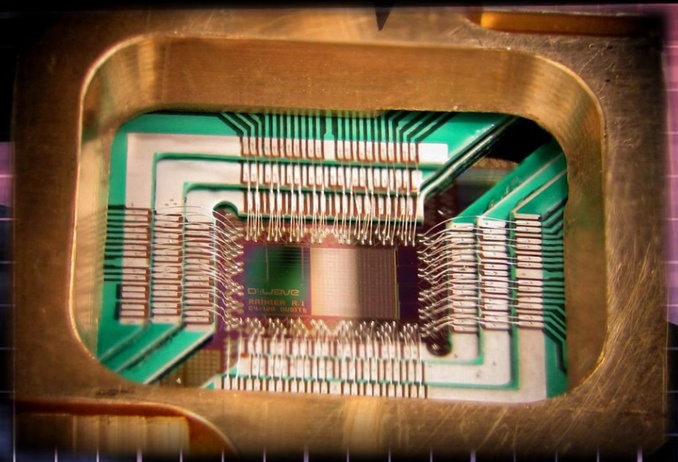
/etc/dovecot.conf

```
ssl_cipher_list=ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

ssl_prefer_server_ciphers = yes (Dovecot 2.2.6 or greater)
```

Regenerate DH Parameters

```
#regenerates every week ssl_dh_parameters_length = 2048
```



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

Call to Action

- Visit the World of Solutions for
  - Cisco Campus – (speaker to add relevant demos/areas to visit)
  - Walk in Labs – (speaker to add relevant walk in labs)
  - Technical Solution Clinics
- Meet the Engineer (Speaker to specify when they will be available for meetings)
- Lunch and Learn Topics
- DevNet zone related sessions

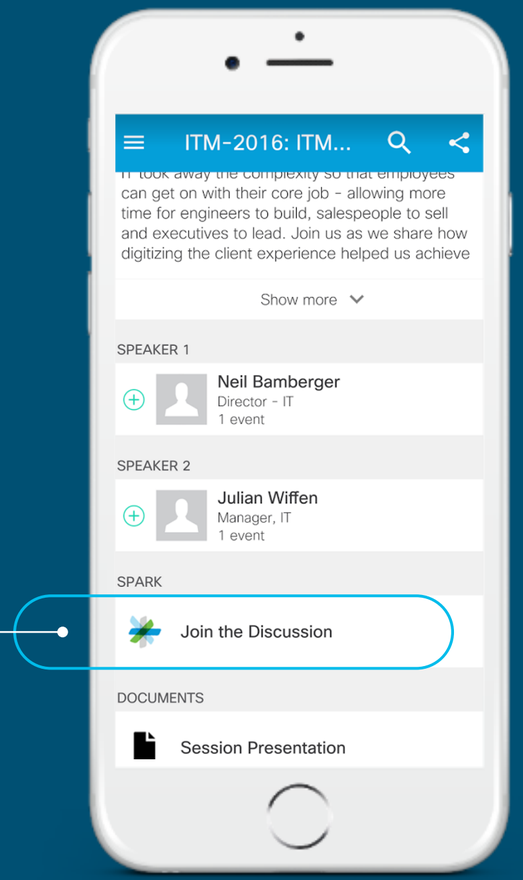
Cisco Spark

Questions?

Use Cisco Spark to communicate with the speaker after the session

How

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

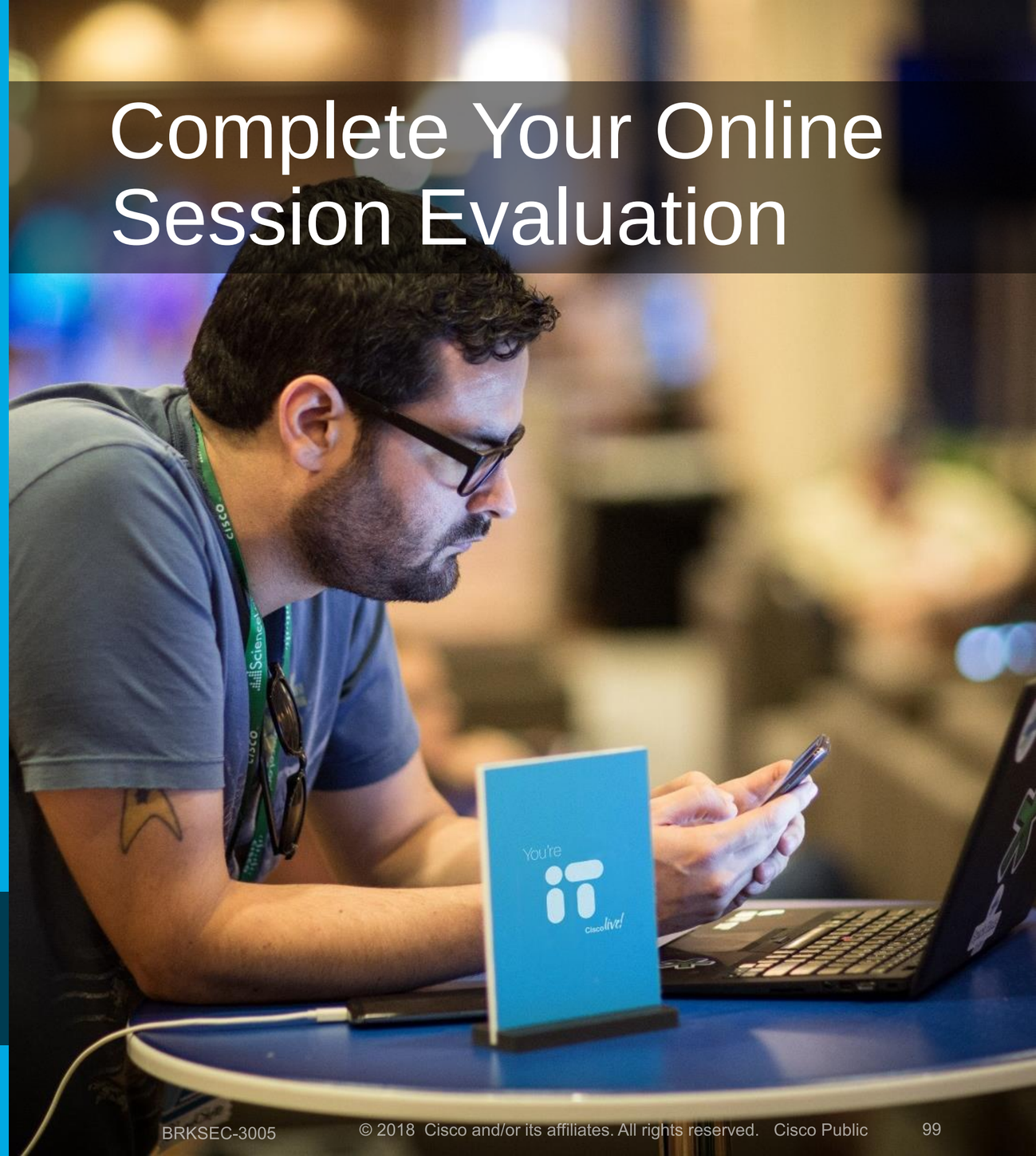


cs.co/ciscolivebot#BRKSEC-3005

Complete Your Online Session Evaluation

- Please complete your Online Session Evaluations after each session
- Complete 4 Session Evaluations & the Overall Conference Evaluation (available from Thursday) to receive your Cisco Live T-shirt
- All surveys can be completed via the Cisco Live Mobile App or the Communication Stations

Don’t forget: Cisco Live sessions will be available for viewing on-demand after the event at www.ciscolive.com/global/on-demand-library/.

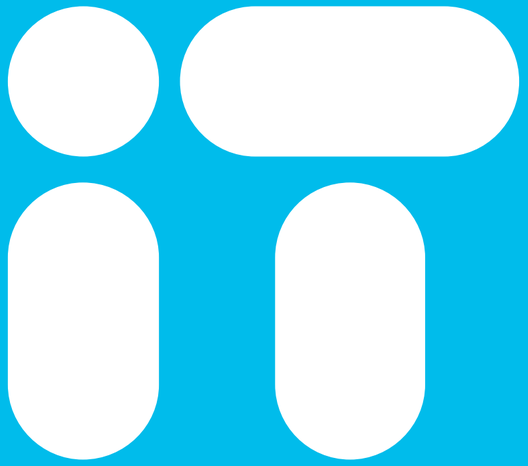


Continue Your Education

- Demos in the Cisco campus
- Walk-in Self-Paced Labs
- Tech Circle
- Meet the Engineer 1:1 meetings
- Related sessions

Thank you

You're



Cisco live!

Demo

Cisco live!

January 29 - February 2, 2018 · Barcelona