PKI Foundations for Security Pros

Jake Hildreth
w/ Tyler Jacobs

Anti-Cast, 2025-11-12

semperis

`Get-ADUser -Identity 'Jake Hildreth'`

- Husband, Dad, Recovering Sysadmin
- Principal Security Consultant @ Semperis
- Open-source Toolmaker:
  Locksmith, BlueTuxedo, PowerPUG!
- Microsoft MVP:
  PowerShell + Identity & Access

```
@{
    Name = 'Tyler Jacobs'
    Alias = 'Poolmanjim'
    About = @(
        'Husband',
        'Boy Dad++',
        'Health Care Sysadmin'
    )
    ExperienceInYears = Get-Random -Min 15 -Max 20
    JobTitle = @(
        'Principal Security Engineer',
        'Directory Services'
    )
    Organization = 'Undisclosed Health Care Org'
    Other = 'Lead Moderator of r/ActiveDirectory'
    Interests = @(
        'Identity',
        'Homelab',
        'TTRPG/Gaming'
    )
}
```

Agenda

- History & Goals
- Encoding vs. Encryption
- Symmetric & Asymmetric Encryption
- Hashing Functions
- Signing
- Certificates
- Public Key Infrastructure (PKI)
- Real-World Uses

Non-Agenda

- Cryptocurrency
- Key Exchange
- MAC/HMAC
- Algorithms
- Cryptanalysis
- Key Space
- Steganography
- AD CS

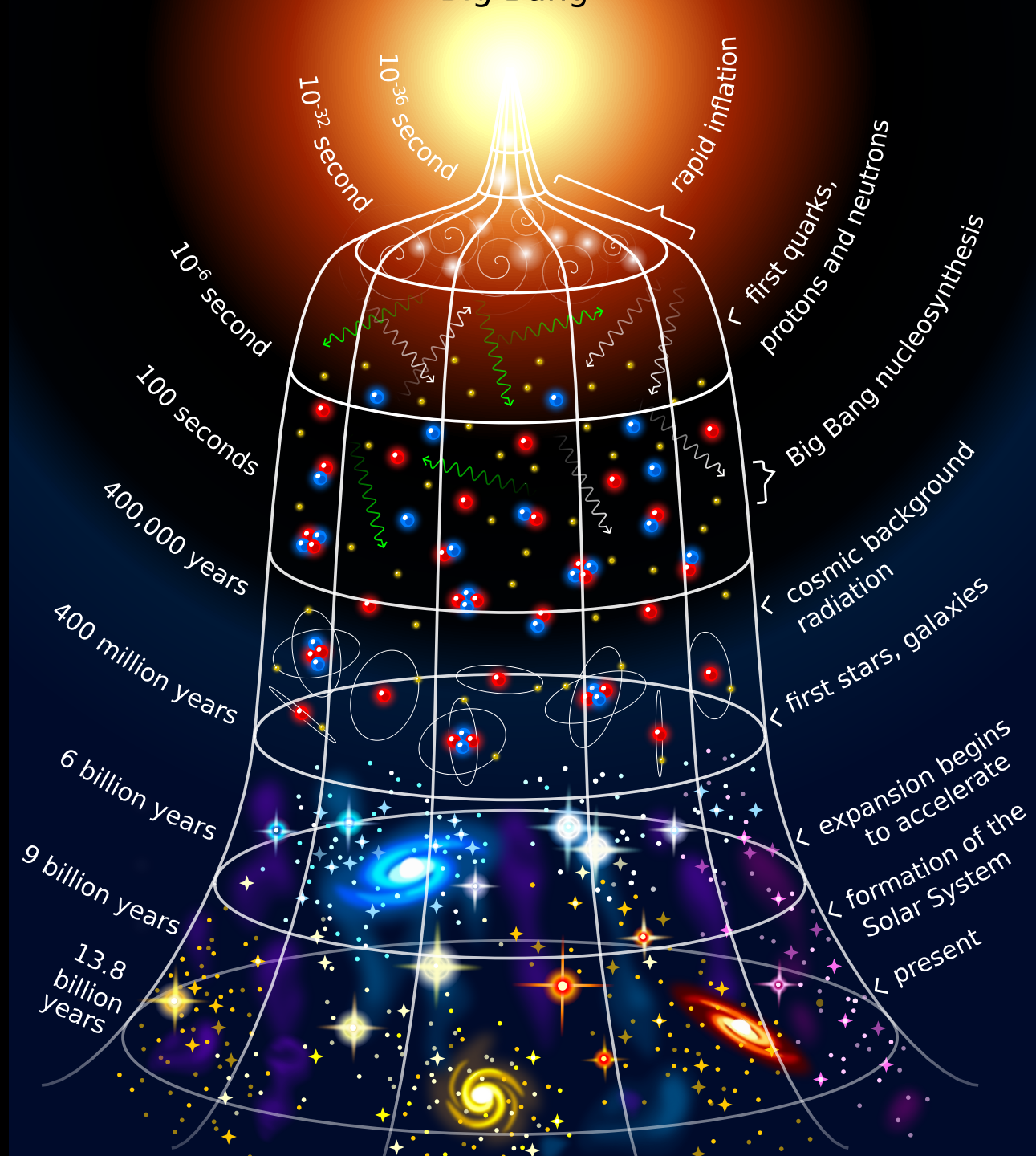
History

In the beginning was the word...

semperis

History

In the beginning was the **pass**word...



Antiquity

- 1900 BCE: Non-Standard Hieroglyphs
- 1500 BCE: Clay Tablets
- 600 BCE: Hebrew Scholars
- 400 BCE: Polybius Square
- 300 BCE: Kama Sutra
- 196 BCE: Rosetta Stone
- 75 BCE: Caesar Cipher

Medieval

- 750-800 CE: Arabic Scholars
- 800-1100 CE: English Scribes
- 1400 CE: Arabic Scholars
- 1508 CE: Tabula Recta
- 1626 CE: Antoine Rossignol

Pre-WWII

- Focus on Cryptanalysis
- Many Medieval Systems Broken
- First Cryptographic Machines
- First One Time Pad

WWII

- Allies:
  - TypeX, SIGABA, Lacida
- Axis:
  - Enigma, Purple, SG-41

Modernity

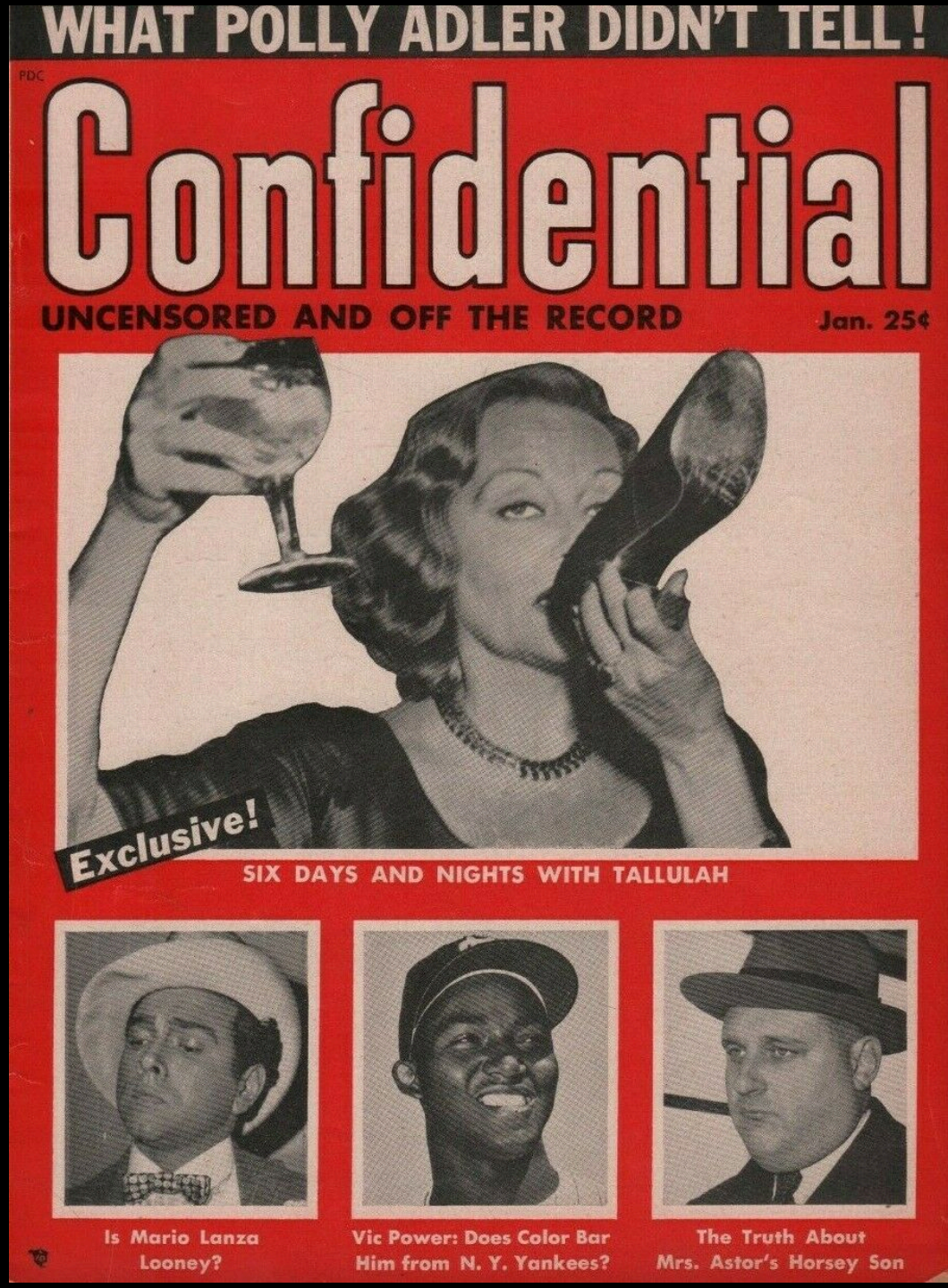
- Claude Shannon
- Data Encryption Standard (DES)
- Diffie-Hellman (DH)
- Rivest, Shamir, Adelman (RSA)
- Message Digest Algorithms (MD*)
- Digital Signature Algorithm (DSA)
- Secure Hash Algorithms (SHA)
- Elliptic Curve Cryptography (ECC)
- Advanced Encryption Standard (AES)
- Post-Quantum Cryptography (PQC)

Goals of Cryptography

semperis

Confidentiality

Ensures a message is only readable by its intended recipient



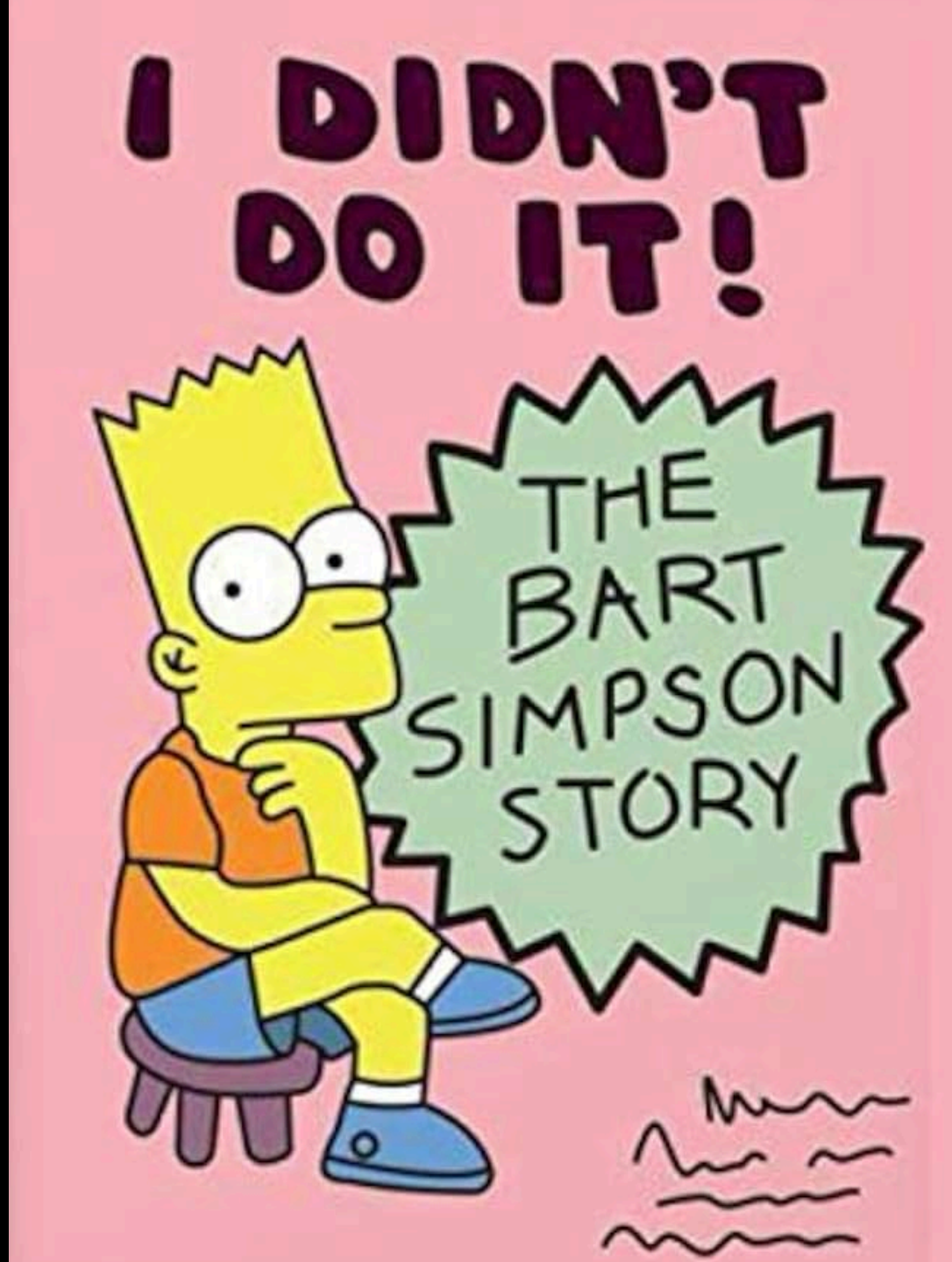
Integrity

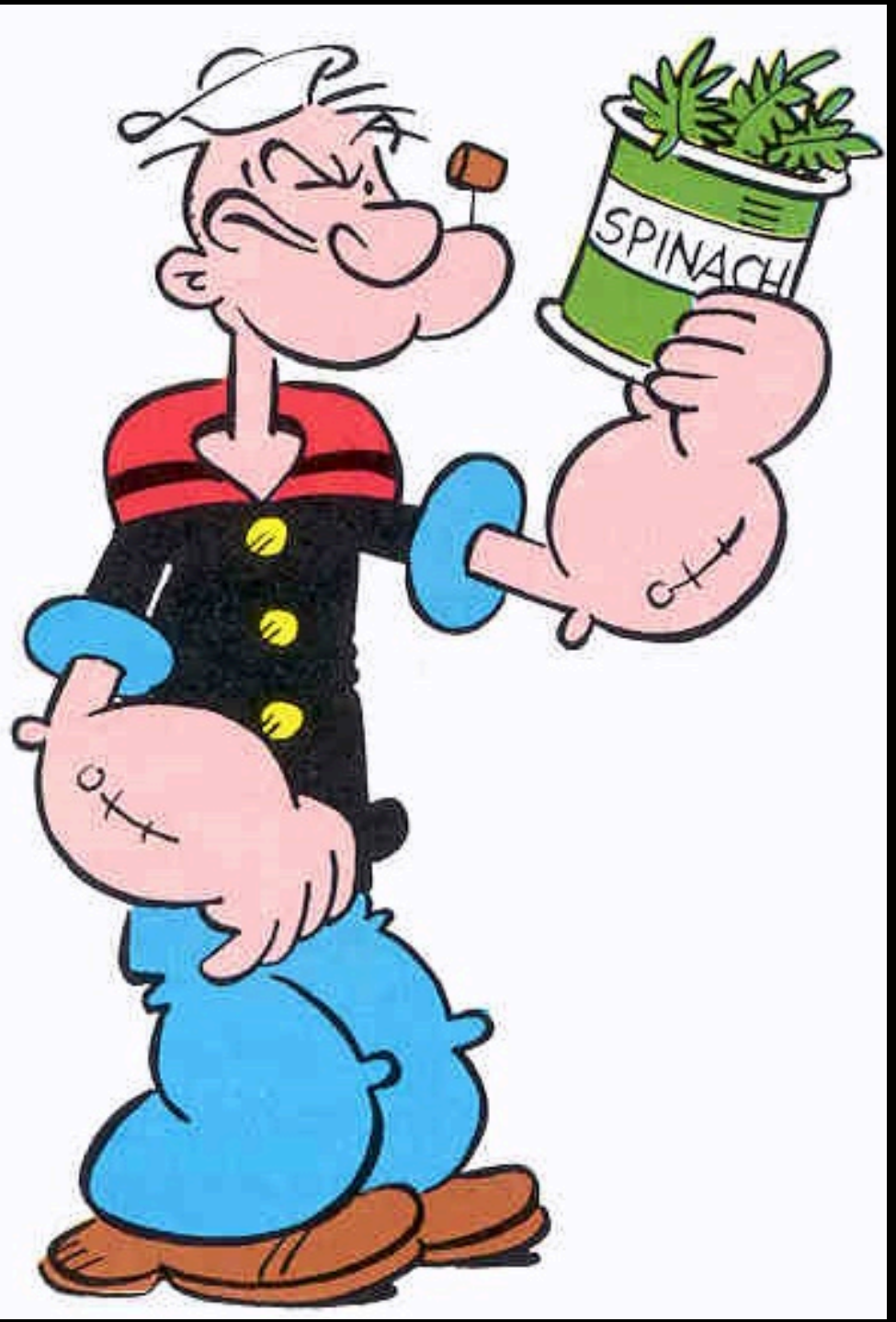
Ensures a message has not been tampered with or changed

semperis

Non-Repudiation

Ensures an author cannot refute authorship of a message





Authentication

Provides proof an author of a message is who they claim

Encoding vs. Encryption

Secret or nah?

Encoding

- Not intended to remain secret
- that is, no **Confidentiality**

Common Schemes

- ASCII
- UTF-8/16/32
- Braille
- Morse Code
- Base64

semperis

Example: Cetacean Cipher

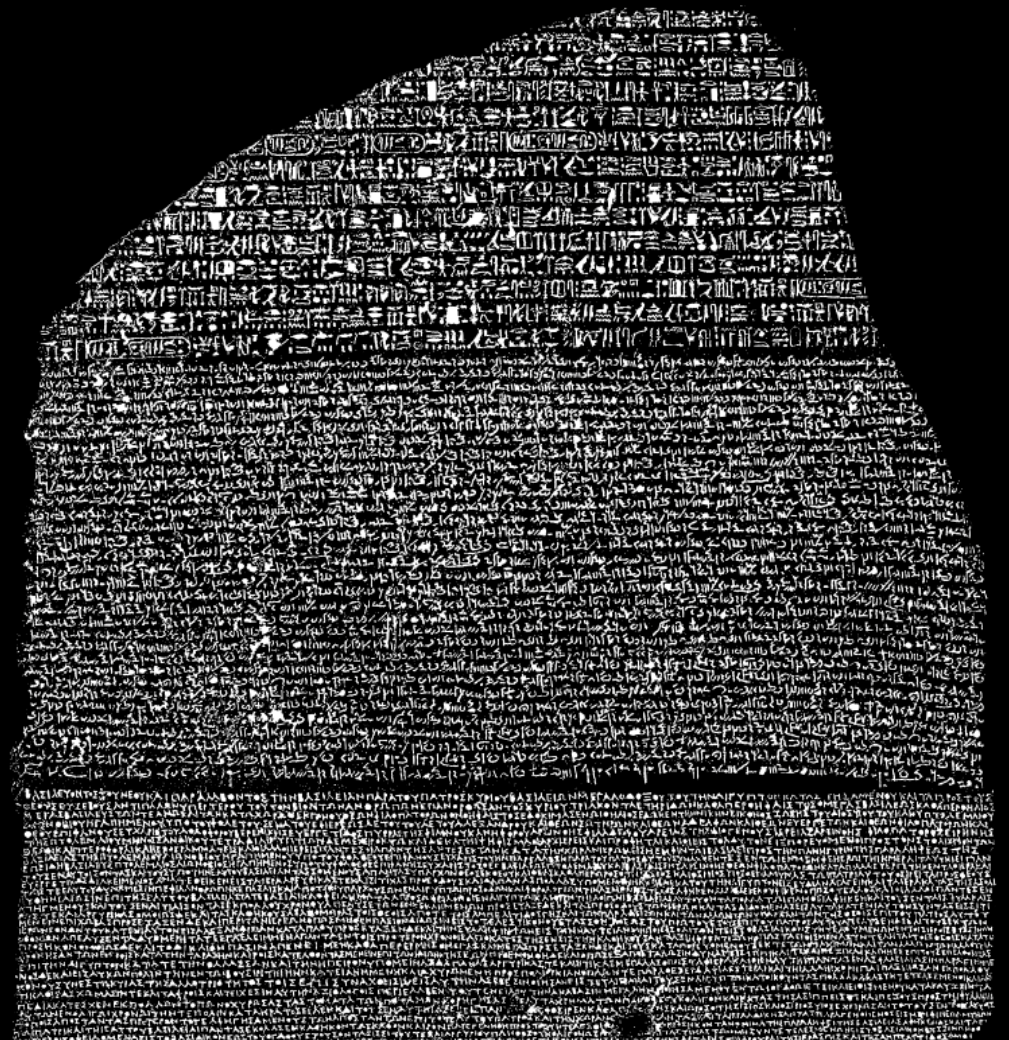
- "Hello, Anti-Cast!" encoded:
- EEEEEEEEEeEEeEEEEEEEEEEEEeeEEeEe
  EEEEEEEEEeeEeeEEEEEEEEEEEeeEeeEE
  EEEEEEEEEeeEeeeeEEEEEEEEEEeEeeEE
  EEEEEEEEEeEEEEEeEEEEEEEEEeeEeeeE
  EEEEEEEEEeeeEeEEEEEEEEEEEeeEeEEe
  EEEEEEEEEEeEeeEeEEEEEEEEEeEEEEee
  EEEEEEEEEeeEEEEeEEEEEEEEEeeeEEee
  EEEEEEEEEeeeEeEEEEEEEEEEEeEEEEe

Try it yourself: https://www.a.tools/Tool.php?Id=389

semperis

Example: Rosetta Stone

- Ancient Egyptian was indecipherable
- Rosetta Stone showed Egyptian next to Greek
- Allowed for translating / "decoding" the Egyptian.
- Encoding can be mistaken for encryption - but once the method is known, it is trivial to break.



Encryption

- Intended to remain secret
- Yes! **Confidentiality**
- Requires 1 or more "keys"
- Keys are **something** used to encrypt/decrypt
- In modern cryptography, keys are **really** big numbers
- Kerckhoffs's principle

Common Schemes

- Symmetric - 1 key
  - DES
  - AES
- Asymmetric - 2+ keys
  - RSA
  - ECC

Encrypted data is

indistinguishable from random noise

Encrypted data ~~is~~ should be

indistinguishable from random noise


Anti-Cast 2025.11.12

Symmetric Encryption

Pretty much all cryptography before the 1970s

Symmetric Encryption

Pros

- Same key to encrypt/decrypt message
  = Easy to understand
- Simple algorithms
  = Very fast

Cons

- Keys must be pre-shared via a secure channel before communication
  = Difficult!
- One key per communication group
  = Key management becomes untenable
- Does not protect Integrity, ensure Non-repudiation, or enable Authentication

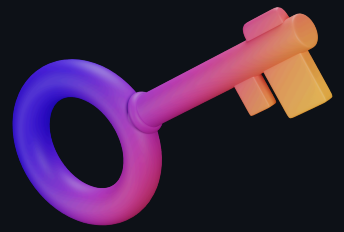
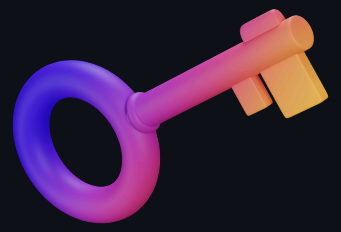
Example: Caesar Cipher (Symmetric)

- Also known as: Shift cipher or ROT*
- Symmetric Encryption ~= ROT13
- A(1) + Shift(13) = N(14)
- N(14) + Shift(13) = A(1)

Original	A	B	C	D	E	F	G	H	I	J	K	L	M
Value	1	2	3	4	5	6	7	8	9	10	11	12	13
"Encrypted"	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Original	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value	14	15	16	17	18	19	20	21	22	23	24	15	26
"Encrypted"	A	B	C	D	E	F	G	H	I	J	K	L	M

Example: Caesar Cipher (Symmetric)

Original Message	Alice Encrypts (ROT13)	Encrypted Message	Bob Decrypts (ROT13)	Received Message
Hello, Anti-Cast!		Olssv, Huap-Jhza!		Hello, Anti-Cast!

Try it yourself: https://rot13.com

Asymmetric Encryption

Encrypt with one & Decrypt with the other

Asymmetric Encryption

Pros

- One **key pair** (public + private) per user
  = Easy(ish) to manage
- Public keys can be shared over insecure channel
  = Much easier than secure channel
- Provides **Confidentiality**, **Integrity**, **Non-repudiation**

Cons

- Complicated algorithms
  = Not fast
- No verification of other side
  = Impersonation possible

Example: Caesar Cipher (Asymmetric)

- Private Key ~= ROT7
- Public Key ~= ROT19
- A(1) + Shift(7) = H(8)
- H(8) + Shift(19) = A(1)

Original	A	B	C	D	E	F	G	H	I	J	K	L	M
Value	1	2	3	4	5	6	7	8	9	10	11	12	13
"Encrypted"	H	I	J	K	L	M	N	O	P	Q	R	S	T

Original	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value	14	15	16	17	18	19	20	21	22	23	24	15	26
"Encrypted"	U	V	W	X	Y	Z	A	B	C	D	E	F	G

Example: Caesar Cipher (Asymmetric)

Original Message	Alice Encrypts (ROT7)	Encrypted Message	Bob Decrypts (ROT19)	Received Message
Hello, Anti-Cast!		Olssv, Huap-Jhza!		Hello, Anti-Cast!

Try it yourself: https://rot13.com

Hash Functions

Neither potatoes not pipes

Hash Functions

- Maps any data to a (probably) unique fixed-length value
- Can't be easily reversed to discover source data
- Changing even a single bit of source data changes the hash
- Ensures **Integrity**

Common Schemes

- MD4/5
- SHA-0/1/2/3
- DSA

Silly Example: Addition

Original	A	B	C	D	E	F	G	H	I	J	K	L	M
Value	1	2	3	4	5	6	7	8	9	10	11	12	13

Original	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value	14	15	16	17	18	19	20	21	22	23	24	15	26

- HELLOANTICAS**T**
  - 8+5+12+12+15+1+14+20+9+3+1+19+**20**
  - Hash: **0139** or **0ACI**
- HELLO
  - 8+5+12+12+15
  - Hash: **0052** or **00EB**
- HELLOANTICAS**H**
  - 8+5+12+12+15+1+14+20+9+3+1+19+**8**
  - Hash: **0127** or **0ABG**

Note: This is a terrible hash function full of collisions. Do not use this for anything.

Real Examples: SHA1 & 2

Original	SHA1	SHA2-256
Hello, Anti-Cast!	94bc685b3657b730ed72696e036260d3fea8ab23	a993f8f095c0c43348cd39fead7ec3fd4a26a53e889d2a89e42adbdfde093398
Hello!	69342c5c39e5ae5f0077aecc32c0f81811fb8193	334d016f755cd6dc58c53a86e183882f8ec14f52fb05345887c8a5edd42c87b7
Hello, Anti-Cash!	6d1ceba487b84474d554599b24dea1fed95264ab	ea37a553bb16967a4545b9f4fb11c19e9dfcdaf5ac3f7fbdfa93a93f7cca145b

Try it yourself: https://gchq.github.io/CyberChef

Real Example #2: Variable-Length -> Fixed Length

Input	Output (MD5)	Output SHA1
The quick brown fox jumps over the lazy dog.	1c6d98786bea70b9c34ce7f33201120c	22b759d30862cc7c7eb3ce9616a9d4e853b1e14d
The quick brown fox jumps over the lazy dog. (x100)	18b94cc7a461d8774a384d8b82345e51	d04c682f53e42e09abfd8a34e810ae9555be8ca4
The quick brown fox jumps over the lazy dog. (x200)	49f1e3a5cc2130d1b5698f5e220598e7	49a0f16e0a6c13f90269cfd4f2e7edc0c95fdf22

Signing

Hashing + Asymmetric Encryption = Better Than Ink

Signing Process

- Alice creates a hash of a message
- Alice encrypts **the hash** with her private key
- Alice sends the message + the encrypted hash to Bob
- Bob decrypts the hash of the message using Alice's public key
- Bob hashes the original message himself
- If hashes match, Alice is the sender!

Common Uses

- Used for **Non-repudiation**
- Software distribution
- Financial transactions
- Contract management
- Network protocols

Example: Caesar Cipher (Asymmetric) + Addition

Original Message + Hash	Alice Encrypts Hash (ROT7)	Original Message + Encrypted Hash	Bob Decrypts Hash (ROT19)	Received Message + Hash	Bob Hashes Message HImself
Hello, Anti-Cast! **0ACI**		Hello, Anti-Cast! **0HJP**		Hello, Anti-Cast! **0ACI**	**0ACI** matches!

Certificates

(Public Key + Attributes) Signed

Certificates

- Public keys don't include identifying information
- Certificates tell who owns a public key
- Certificates provide basic **Authentication**
- **If** you trust the issuer, you can trust the public key
- Self-signed certs still permit encryption!

Typical Contents

- Subject
- Issuer
- Public Key
- Not Before/Not After
- Key Usage/Extended Key Usage
- Signature Algorithm/Signature
- Serial Number

Generating a Self-Signed Certificate

Create a Key Pair	Create CSR	Hash the CSR	Encrypt CSR Hash with Alice's Private Key	Package Certificate
 	Subject: Alice Public Key: 	0BHF	0UAY	Subject: Alice Issuer: Alice Public Key:  Signature Algorithm: Silly Signature: 0UAY

Certification Authority (CA) Certificate Generation

Create a Key Pair	Create CSR	Hash the CSR	Encrypt CSR Hash with CA's Private Key	Package Certificate
 	Subject: Alice Public Key: 	0BHF	0KQO	Subject: Alice Issuer: CA Public Key:  Signature Algorithm: Silly Signature: 0KQO

Public Key Infrastructure (PKI)

Formalized trust - all the way down to the root

PKI Basics

- Not **just** technology:
  - Hardware
  - Software
  - Policies
  - Procedures
- "Chains" of trust lead back to a "root" of trust
- **If** you trust the PKI (big if), you can trust others that use the PKI

Common Components

- Authorities:
  - Root/Intermediate/Issuing
  - Validation
  - Registration
  - Policy
  - Timestamp
- Users:
  - End Entities
  - Relying Parties

How Do I Know If I Should Trust a PKI?

semperis

How Do I Know If I Should Trust a PKI?

Anti-Cast, 2025-11-12

How Do I Know If I Should Trust a PKI?

- Mostly handled for you
- OSes include lists of trusted roots
  - They're also usually out of date - Server 2025 GA included 7 Root CAs that were expired.
- Active Directory (AD) networks have PKI: AD Certificate Services
  - Drink!
- Reputable PKIs publish their configurations and procedures
- Honestly, something I need to dig into...

Real-World Uses

Hybrid Cryptosystems - **VERY** Simplified

Pretty Good Privacy (PGP)/GNU Privacy Guard (GPG)

Encryption Process

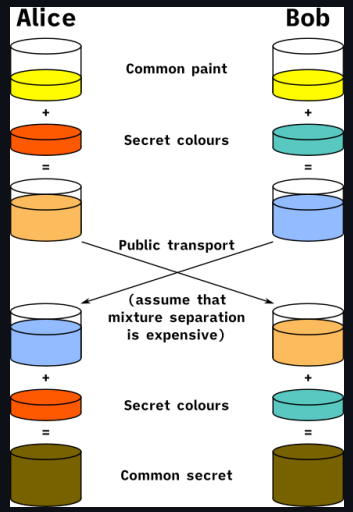
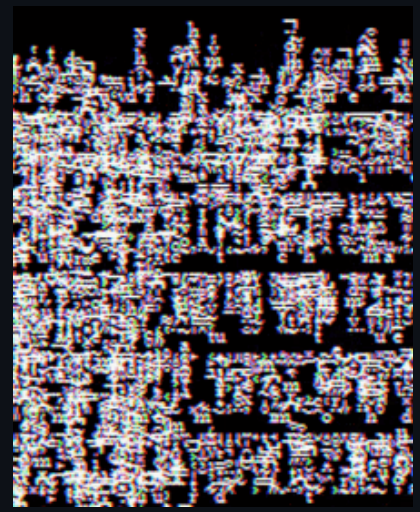
Alice Writes a Message & Creates a Random Key	Alice Encrypts The Random Key With Bob's Public Key	Alice Encrypts the Message with the Random Key	Alice Combines the Encrypted Message and Key And Sends to Bob
Message: Hello, Anti-Cast! Key: 4dyea1wo	Encrypted Key: 7tjtf57d	Encrypted Message: Fcjjm, Ylrg-Ayqr!	Encrypted Message: Fcjjm, Ylrg-Ayqr! Encrypted Key: 7tjtf57d

PGP/GPG

Decryption Process

Bob Receives the Encrypted Package	Bob Decrypts The Random Key With His Private Key	Bob Decrypts the Message with the Random Key
Encrypted Message: Fcjjm, Ylrg-Ayqr! Encrypted Key: 7tjtf57d	Decrypted Key: 4dyea1wo	Decrypted Message: Hello, Anti-Cast!

Secure Sockets Layer (SSL)/Transport Layer Security (TLS)

Client Contacts Server via HTTPS	Server Sends Certificate for Client to Validate	Server and Client Agree on a Session Key (Diffie-Helman)	Data Stream is Encrypted with Session Key
https://dotdot.horse	Name: dotdot.horse Issuer: CA Signature: Signature		

Quick Review

Quick Review

- Humans have wanted to hide things **forever**
- Encoding is reversible **w/o keys**
- Encryption is reversible **with keys**
  - Symmetric Encryption:
    - Uses 1 key
    - FAST
  - Asymmetric Encryption:
    - Uses 2+ keys
    - SLOW
- Hashing is not reversible
- Signing combines hashing and encryption to provide non-repudiation
- Certificates bind an identity to a public key
- PKI solves a lot of problems… but at a cost
- Most modern cryptosystems combine symmetric and asymmetric encryption, hashing, signing, and certificates!

Resources

https://github.com/jakehildreth/PKIFoundations

Links

Cetacean Cipher: https://www.a.tools/Tool.php?Id=389

CyberChef: https://gchq.github.io/CyberChef

ROT13: https://rot13.com

Image Credits

https://github.com/jakehildreth/PKIFoundations/blob/main/images/images.md

Thanks!

Find	Jake	Tyler
LinkedIn	/in/jakehildreth	/in/thetylerjacobs/
GitHub	jakehildreth	ActiveDirectoryKC
Reddit	**no.**	/u/poolmanjim
Site	jakehildreth.com	activedirectorykc.net