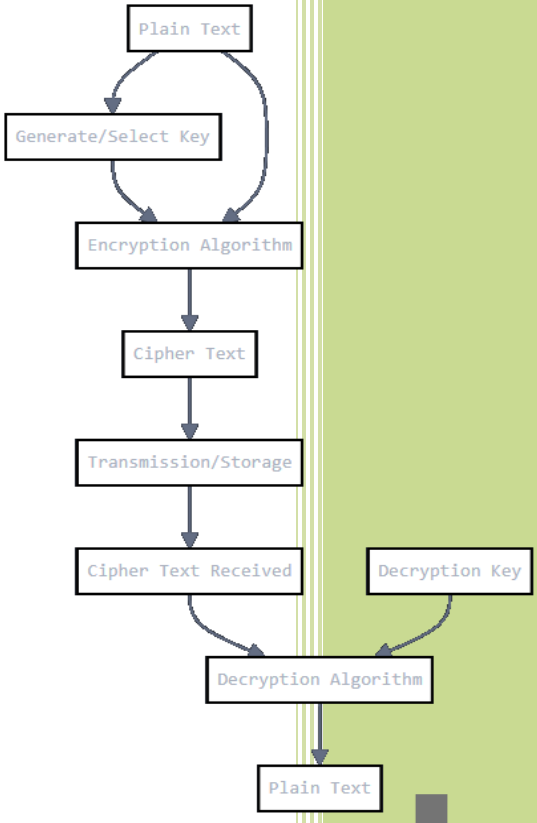
CRYPTOGRAPHY ALGORITHMS

CRYPTOGRAPHY ALGORITHMS

For the non-technical people



leonel pedroza

CRYPTOGRAPHY ALGORITHMS

For the non-technical people

Contents

INTRODUCTION

Let's talk about cryptography. No, really. Stay with me here.

It sounds as if it belonged to some math nerds; but when you actually send a text, buy anything online, or check your bank balance, cryptography is quietly at work in the background. Without it, our digital lives never existed. I would bet my life against it that most of us would not know how it works.

Try considering that data breaches are everywhere. Just headlines? No. They're affecting companies we all know about-now some "maybe" actually employing us. Millions fine. Lost reputations. All because somebody, somewhere, failed to properly lock the digital door. So what does cryptography really do to make us feel safe? Well, let's say you are trying to carry a secret note across a crowded room.

You could fold it into a paper airplane and hope for the best (spoiler: bad idea). Or you could write it in a code only your friend understands. That's cryptography in a nutshell—except instead of paper airplanes, we're dealing with credit card numbers, medical records, and those embarrassing photos from college.

The beauty of modern cryptography? It's not just one trick. We've got symmetric encryption—fast, efficient, perfect for encrypting your entire hard drive. Then there's asymmetric encryption, which solved a problem that stumped people for centuries: how do you share a secret key with someone you've never met? Brilliant stuff. Add in hash functions (think digital fingerprints) and digital signatures (like notarized documents, but cooler), and you've got a pretty solid toolkit.

Oh, and quantum computers? They're coming. Not tomorrow, but soon enough that smart people are already working on "quantum-proof" encryption. Because apparently, regular encryption wasn't complicated enough.

This guide breaks down eight major types of cryptography. No PhD required. Just real-world explanations of how these technologies work, where they shine, where they stumble, and why you should care. Because whether you're protecting customer data or just trying to understand why your IT department keeps talking about "key rotation," this stuff matters.

Ready? Let's dive in.

SYMMETRIC KEY CRYPTOGRAPHY

Think of symmetric key encryption like having a single house key. You and your roommate both need copies of the exact same key to get in. Simple? Yes. But here's the catch.

These algorithms are blazingly fast. We're talking about encrypting gigabytes of data without breaking a sweat. That's why your VPN doesn't slow to a crawl when you're streaming Netflix through it. The same shared key locks and unlocks everything.

The good stuff:

- Speed. Seriously, it's like comparing a sports car to a bicycle.
- Perfect for those massive data dumps you need to secure.

The not-so-good stuff:

- How do you safely give someone that key? Can't exactly email it.
- Lose the key? Game over. Everything's compromised.

Where you'll find it:

Your Wi-Fi uses AES (that's what WPA2/WPA3 is all about). Some banks still run DES in their dusty old systems—though they really shouldn't.

ASYMMETRIC KEY CRYPTOGRAPHY

Now this is where things get clever. Instead of one key, you get two: one public, one private. It's like having a mailbox where anyone can drop letters in (public key), but only you have the key to open it (private key).

This solves that pesky key-sharing problem. Want to send me a secret? Use my public key. I'll decrypt it with my private one. Magic? Almost.

Why it rocks:

- No more secret key exchanges in dark alleys.
- Digital signatures! Prove you're really you.

Why it sometimes doesn't:

- Slow. Like, really slow compared to symmetric.
- Your computer will feel it, especially on older hardware.

Real-world examples:

Every time you see that little padlock in your browser? That's RSA working behind the scenes. Your banking app? Probably using ECC to keep things snappy on your phone.

HASH FUNCTIONS

Hash functions are the unsung heroes of cybersecurity. Feed them any data—a password, a file, your entire music collection—and they'll spit out a fixed-length string. Change even one bit of the input? Completely different output.

Think of it as a digital fingerprint. Unique, consistent, and impossible to reverse-engineer.

The perks:

- Lightning fast and predictable.
- Perfect for checking if someone tampered with your files.

The downsides:

- Old ones like MD5? Broken. Don't use them for anything serious.
- One-way street—you can't get your original data back.

In the wild:

Bitcoin wouldn't exist without SHA-256. And MD5? Still hanging around for simple file checks, though it's basically retired from security duty.

BLOCK CIPHERS

Picture this: You're encrypting a novel, but instead of doing it letter by letter, you're working with whole paragraphs. That's block ciphers—they chop your data into neat chunks (usually 128 bits) and encrypt each one.

These are the workhorses of modern encryption. AES, the current champion, has been battle-tested for decades.

What's great:

- Rock-solid algorithms that have survived years of attacks.
- Flexible—pick your mode based on what you need.

What's tricky:

- Data doesn't always fit perfectly into blocks. Hello, padding problems.
- Choose the wrong mode? You might accidentally weaken your security.

Where they live:

Your laptop's full-disk encryption (BitLocker, FileVault)? Block ciphers. Those encrypted containers you use for sensitive files? Yep, block ciphers again.

STREAM CIPHERS

Stream ciphers are the sprinters of encryption. While block ciphers process chunks, these encrypt bit by bit, byte by byte. It's like the difference between sending packages versus a continuous stream of water.

Real-time applications love these. Video calls, live streams—anywhere latency matters.

The advantages:

- Near-zero delay. Perfect for "right now" encryption.
- No awkward padding to deal with.

The gotchas:

- Reuse a key? Catastrophic failure. Seriously, don't.
- Harder to get right than they look.

Examples:

RC4 used to be everywhere (looking at you, old SSL), but it's retired now. Modern apps often use Salsa20 or ChaCha20—same speed, better security.

PKI (PUBLIC KEY INFRASTRUCTURE)

PKI is... complicated. Imagine trying to verify everyone's identity in a city of millions. That's what PKI does for the internet. It's a whole ecosystem of certificates, authorities, and trust chains.

When you visit a website and your browser says "secure," that's PKI doing its thing. A certificate authority vouched for that site, and your browser trusts the authority.

The good:

- Scales beautifully. Millions of certificates, no problem.
- Makes secure communication possible with strangers.

The headaches:

- Setting it up? Not for the faint of heart.
- Revoking compromised certificates is messier than it should be.

Daily encounters:

Every HTTPS website. Encrypted emails with S/MIME. Code signing that proves your software isn't malware.

DIGITAL SIGNATURES

Here's something cool: digital signatures do what your handwritten signature can't. They prove you signed something AND that nobody changed it afterward. It's like sealing a letter with wax that screams if someone tries to open it.

Using asymmetric crypto under the hood, these create mathematical proof of authenticity.

Why they matter:

- Can't deny you signed it (non-repudiation, if you want the fancy term).
- Tamper-evident by design.

The catches:

- Guard that private key with your life.
- All that math needs processing power.

Everyday uses:

Signing PDFs electronically. Software updates that your computer trusts. Smart contracts that execute themselves.

ONE-TIME PAD (OTP)

The one-time pad is cryptography's holy grail. Unbreakable. Literally. But there's a reason you're not using it to encrypt your emails.

The key must be:

1. Truly random (not computer-generated "random")
2. As long as your message
3. Used exactly once
4. Kept absolutely secret

Meet all those requirements? Congratulations, you've achieved perfect secrecy. Miss one? You've got expensive noise.

The dream:

- Mathematically proven unbreakable.
- No supercomputer can touch it.

The reality:

- Those key requirements? Nearly impossible at scale.
- A 1GB file needs a 1GB key. Good luck with that.

Who actually uses it:

Spy agencies for their most critical communications. Historical hotlines between superpowers. Anyone else? Probably overkill.

FINAL THOUGHTS

Here's what I've learned after years in this field: cryptography isn't magic. It's math. Really, really good math. But math alone won't save you.

I've seen companies spend fortunes on state-of-the-art encryption, only to store their passwords in a spreadsheet called "passwords.xlsx." (Yes, really.) The best lock in the world won't help if you leave the key under the doormat.

So where are we headed? Quantum computing is the elephant in the room—or maybe the elephant stampeding toward the room. We don't know exactly when quantum computers will crack current encryption, but we know they will. Could be 10 years. Could be 20. The smart money says: prepare now, panic later.

Meanwhile, some genuinely exciting stuff is happening. Homomorphic encryption lets you run calculations on encrypted data without ever decrypting it. Imagine a doctor analyzing your medical data without actually seeing it. Secure multi-party computation allows a number of persons and organizations to jointly compute something but does not let them expose their respective inputs. It's like every chef bringing ingredients for a soup but no one really knows what ingredients the other brought.

Mind-blowing? Perhaps a bit.

But here's my advise: Start way simpler. Get the basics straight. Make sure you use encryption where it matters. Keep your systems up to date-the security patches are not useless at all. Train the people because the human factor is still the weakest link. And for all things secure and good, manage your keys carefully.

The truth? Perfect security doesn't exist. Never has, never will. But good security? That's achievable. It takes work, constant vigilance, and a willingness to adapt as threats evolve.

Stay curious. Join security communities—they're surprisingly welcoming. Read about new developments, even if they seem overwhelming at first. Ask questions. Break things (legally, in test environments). Learn from mistakes—yours and others'.

Because at the end of the day, cryptography isn't about the algorithms or the math or even the technology. It's about protecting what matters: our privacy, our businesses, our digital lives.

And that's worth getting right..

GLOSSARY

AES (Advanced Encryption Standard): A symmetric block cipher algorithm adopted by NIST in 2001, using key sizes of 128, 192, or 256 bits to encrypt data in 128-bit blocks. Currently the most widely used encryption standard worldwide.

Asymmetric Cryptography: Also called public key cryptography, a system using mathematically related key pairs where the public key encrypts and the private key decrypts, solving the key distribution problem.

Block Cipher: An encryption method that processes fixed-size blocks of data (typically 64 or 128 bits) using a symmetric key, forming the basis for many encryption modes and protocols.

Certificate Authority (CA): A trusted entity in PKI that issues and manages digital certificates, verifying the identity of certificate holders and binding public keys to entities.

Cipher: An algorithm for performing encryption or decryption, transforming plaintext into ciphertext or vice versa through mathematical operations.

Cryptographic Hash Function: A one-way mathematical function that produces a fixed-size output (hash) from arbitrary input data, used for data integrity verification and password storage.

Digital Certificate: An electronic document that uses a digital signature to bind a public key with an identity, enabling secure communications and authentication in PKI systems.

Digital Signature: A cryptographic technique using asymmetric keys to verify the authenticity and integrity of digital messages or documents, providing non-repudiation.

Entropy: A measure of randomness or unpredictability in cryptographic systems, essential for generating secure keys and preventing pattern-based attacks.

FIPS (Federal Information Processing Standards): US government standards for cryptographic modules and algorithms, with FIPS 140-2/3 being the primary standard for cryptographic module validation.

Key Derivation Function (KDF): A cryptographic algorithm that derives one or more secret keys from a master key or password, often using techniques like PBKDF2 or Argon2.

Key Exchange: The process of securely sharing cryptographic keys between parties over an insecure channel, commonly implemented using protocols like Diffie-Hellman or ECDH.

Keystream: A stream of random or pseudorandom characters combined with plaintext in stream cipher operations, typically generated from a key and initialization vector.

Mode of Operation: A technique for applying block ciphers to encrypt data larger than the cipher's block size, including ECB, CBC, CTR, GCM, and XTS modes.

NIST (National Institute of Standards and Technology): The US federal agency responsible for developing cryptographic standards and guidelines, including AES, SHA, and post-quantum cryptography standards.

Nonce: A "number used once" in cryptographic protocols to prevent replay attacks and ensure uniqueness in encryption operations, critical for secure communications.

PKI (Public Key Infrastructure): A framework of policies, procedures, and technologies for creating, managing, and revoking digital certificates and public keys.

Post-Quantum Cryptography (PQC): Cryptographic algorithms designed to remain secure against attacks from both classical and quantum computers, addressing the future threat of quantum computing.

RSA: A widely-used asymmetric encryption algorithm based on the difficulty of factoring large prime numbers, named after Rivest, Shamir, and Adleman.

Salt: Random data added to passwords before hashing to prevent rainbow table attacks and ensure identical passwords produce different hash values.

Stream Cipher: An encryption method that encrypts data one bit or byte at a time using a keystream, ideal for real-time communications and streaming data.

Symmetric Cryptography: Encryption where the same key is used for both encryption and decryption, offering high performance but requiring secure key distribution.

APPENDICES

Appendix A: Which Protocol is Used and When - A Quick Guide

Use Case	Recommended Protocol	Key Considerations	Avoid
File Encryption	AES-256 (GCM mode)	Fast, secure, widely supported	DES, 3DES (outdated)
Database Encryption	AES-256 (XTS mode for storage)	Designed for sector-based storage	ECB mode
Web Communications (HTTPS)	TLS 1.3 with AES-GCM or ChaCha20-Poly1305	Forward secrecy, modern ciphers	SSL, TLS 1.0/1.1
Email Encryption	S/MIME or PGP/GPG	Certificate management required	Plain SMTP
Password Storage	Argon2id, bcrypt, or scrypt	Use strong salting, high work factors	MD5, plain SHA
VPN Tunnels	AES-256-GCM with IKEv2/IPSec	Performance vs. security balance	PPTP, outdated ciphers
Digital Signatures	RSA-2048+ or ECDSA P-256+	Non-repudiation, legal compliance	RSA < 2048 bits
Real-time Communications	ChaCha20-Poly1305 or AES-CTR	Low latency requirements	RC4 (broken)
Mobile Applications	ECDSA/ECDH with AES	Battery efficiency important	Large RSA keys
IoT Devices	Lightweight crypto (ASCON, GIFT)	Resource constraints	Heavy algorithms
Blockchain/Cryptocurrency	SHA-256, ECDSA, EdDSA	Consensus and immutability	Weak hash functions
Certificate Generation	RSA-2048+ or ECDSA P-256+	Industry standard compliance	Self-signed for production
API Authentication	HMAC-SHA256 or JWT with RS256	Token expiration, rotation	Basic authentication

Use Case	Recommended Protocol	Key Considerations	Avoid
Cloud Storage	Client-side AES-256-GCM	Zero-knowledge architecture	Provider-only encryption
Messaging Apps	Signal Protocol (Double Ratchet)	End-to-end encryption	Proprietary protocols
Financial Transactions	3DES (legacy) migrating to AES	PCI-DSS compliance	Outdated algorithms
Post-Quantum Readiness	CRYSTALS-Kyber, CRYSTALS-Dilithium	NIST PQC standards	Current algorithms alone

Appendix B: Who Makes These Things? Vendors and Standards Organizations

Standards Organizations

Organization	Role	Key Standards
NIST	US federal cryptographic standards	FIPS 140-3, AES, SHA, PQC standards
ISO/IEC	International standards	ISO/IEC 19790, 18033, 29192 (lightweight)
IETF	Internet standards	TLS, IPsec, S/MIME, JOSE
IEEE	Technical standards	802.11i (WPA2), P1363 (public key)
ETSI	European standards	Quantum Key Distribution standards
ANSI	US private sector	X9 series for financial cryptography

Commercial Vendors

Category	Major Vendors	Notable Products/Services
HSM (Hardware Security Modules)	Thales, Entrust (nCipher), Utimaco	Luna, nShield, SecurityServer
PKI Solutions	DigiCert, Sectigo, GlobalSign	Certificate management, CA services
Encryption Software	Symantec, McAfee, Vormetric	Endpoint encryption, data protection
Cloud Crypto Services	AWS, Microsoft Azure, Google Cloud	KMS, CloudHSM, Key Vault
Post-Quantum Solutions	IBM, ISARA, PQShield	Quantum-safe algorithms, migration tools
Crypto Libraries	OpenSSL, Bouncy Castle, wolfSSL	Open source implementations
Authentication	RSA Security, Yubico, Duo	SecurID, YubiKey, MFA solutions
Blockchain Platforms	Ethereum, Hyperledger, R3 Corda	Smart contracts, DLT infrastructure

Open-Source Projects

Project	Description	Use Case
OpenSSL	Comprehensive crypto library	General purpose encryption
GnuPG	OpenPGP implementation	Email encryption, signing
Bouncy Castle	Java/C# crypto APIs	Application development
libsodium	Modern, easy-to-use crypto	Developer-friendly implementation
OpenVPN	VPN solution	Secure tunneling
Let's Encrypt	Free certificate authority	HTTPS certificates

REFERENCES

Web Resources and References

Standards and Guidelines:

- NIST Cryptographic Standards and Guidelines: https://www.nist.gov/cryptography
- NIST Computer Security Resource Center (CSRC): https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines
- NIST Post-Quantum Cryptography: https://csrc.nist.gov/projects/post-quantum-cryptography
- ISO/IEC JTC 1/SC 27 IT Security Techniques: https://www.iso.org/committee/45306.html

Technical References:

- IETF Transport Layer Security (TLS): https://datatracker.ietf.org/wg/tls/documents/
- OpenSSL Documentation: https://www.openssl.org/docs/
- Cryptographic Algorithm Validation Program: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

Industry Resources:

- SANS Cryptography Resources: https://www.sans.org/cyber-security-courses/cryptography/
- OWASP Cryptographic Storage Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html
- Crypto Museum - Historical Cryptography: https://www.cryptomuseum.com/

Post-Quantum Cryptography:

- PQC Standardization Process: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

- ETSI Quantum Safe Cryptography: https://www.etsi.org/technologies/quantum-safe-cryptography
- Post-Quantum Cryptography Conference (PQCrypto): https://pqcrypto.org/

Tools and Implementations:

- CrypTool Portal: https://www.cryptool.org/
- Bouncy Castle Crypto APIs: https://www.bouncycastle.org/
- libsodium Documentation: https://doc.libsodium.org/

Current Developments:

- Quantum Computing Report: https://quantumcomputingreport.com/
- International Association for Cryptologic Research: https://www.iacr.org/
- Bruce Schneier's Crypto-Gram Newsletter: https://www.schneier.com/crypto-gram/

A Little About Me & Why I Do This

BSc. Electronic engineer

Senior Network Analyst | Cybersecurity Engineer | Telecommunications Specialist

So, what's my deal? I've got this personal mission, you could call it a challenge: to explain how cybersecurity works in plain English, or as I like to say, in "muggle" language. For over 12 years, I was a technical instructor for big corporate clients, and I can't even count the number of internal training sessions I ran for colleagues across Latin America. I've also had the privilege of being a university professor, teaching systems engineering to both undergrads and graduate students.

If there's one thing all those years have taught me, it's this: not everyone speaks "geek." And that's perfectly okay! My goal here, and with other stuff I create, is to cut through the dense technical jargon without losing the important stuff. I genuinely hope that if you're reading this, you feel a bit more confident and can speak up about why understanding cybersecurity is so deeply important these days. It's not just for the IT crowd anymore; it's for everyone.

→ Want to learn more or find some tools? Check out my stuff on **GitHub**:

github.com/leonelpedroza