AMERICAN UNIVERSITY OF BEIRUT

Applied Cryptography

CMPS 297AD/396AI

Fall 2025

Part 1: Provable Security

1.1: Introduction

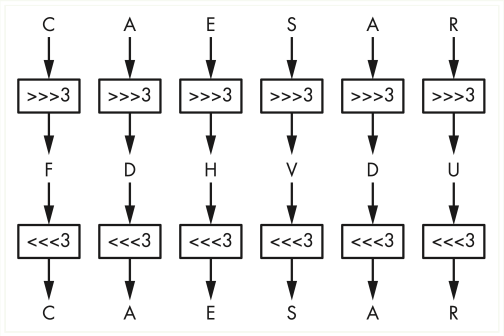
Nadim Kobeissi

https://appliedcryptography.page

Defining cryptography

What is Cryptography?

"The science of enabling secure and private computation, communication, verification, and delegation in the presence of untrusted parties, adversarial behavior, and mutually distrustful participants."

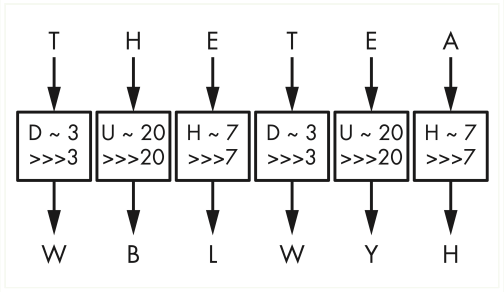


Source: Serious Cryptography, 2nd Edition

Defining cryptography

What is Cryptography?

"The science of enabling secure and private computation, communication, verification, and delegation in the presence of untrusted parties, adversarial behavior, and mutually distrustful participants."

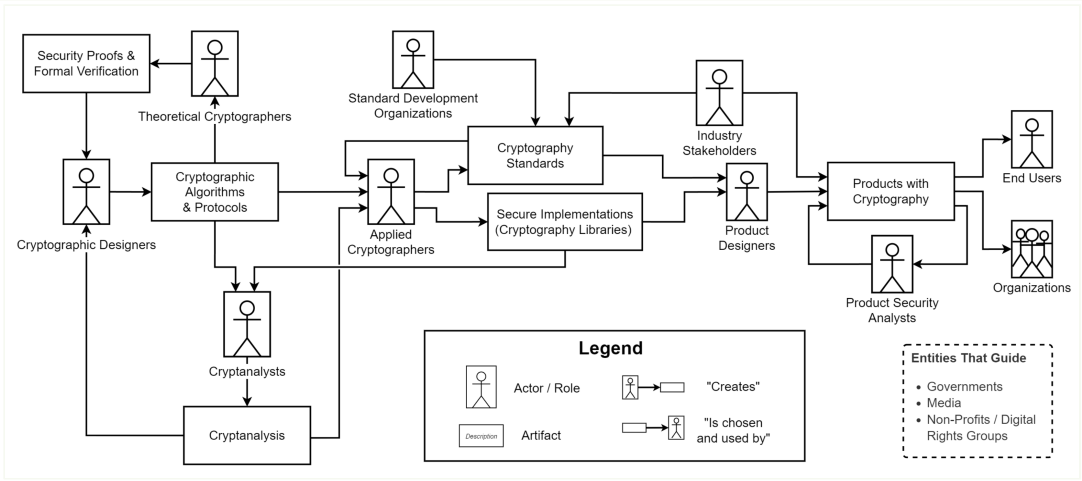


Source: Serious Cryptography, 2nd Edition

Cryptography is everywhere

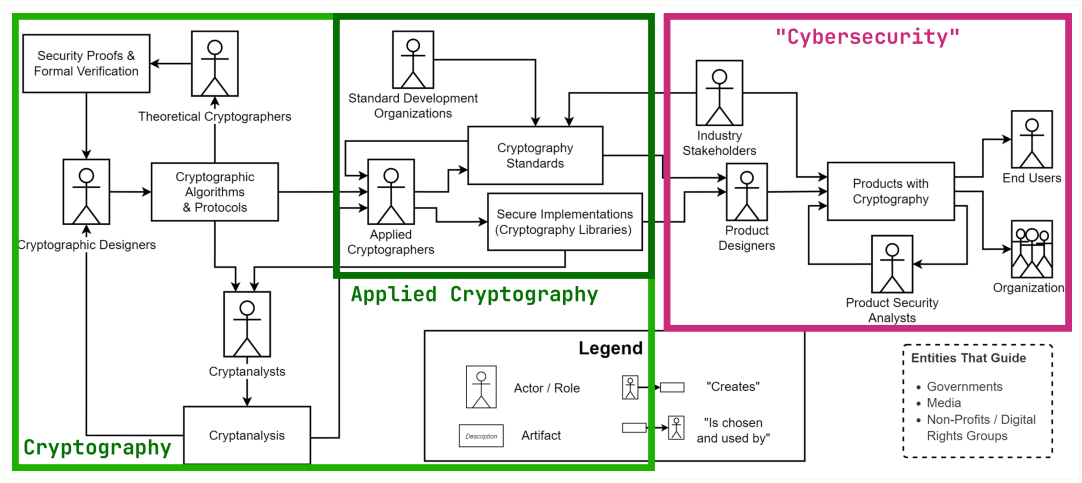
- Banking
- Buying stuff from the store
- Any digital payment system
- Messaging (WhatsApp, Signal, iMessage, Telegram)
- Voice calls
- Government and military systems
- SSH
- VPN access
- Visiting most websites (HTTPS)
- Disk encryption
- Cloud storage
- Video conferencing
- Unlocking your (newer) car
- Identity card systems
- Ticketing systems
- DRM solutions
- Private contact discovery
- Cryptocurrencies
- That iPhotos feature that detects similar photos

How it's made



Fischer et al., The Challenges of Bringing Cryptography from Research Papers to Products: Results from an Interview Study with Experts, USENIX Security 2024

How it's made



Fischer et al., The Challenges of Bringing Cryptography from Research Papers to Products: Results from an Interview Study with Experts, USENIX Security 2024

Cryptographic building blocks

Components

- Cryptography manifests as a set of primitives, from which we build protocols intended to accomplish well-defined security goals.
- **Primitives**: AES, RSA, SHA-2, DH...
- **Protocols**: TLS, Signal, SSH, FileVault 2, BitLocker...

Examples

- **AES**: Symmetric encryption
  - $\mathsf{Enc}(k, m) = c$, $\mathsf{Dec}(k, c) = m$.
- **SHA-2**: Hash function
  - $\mathsf{H}(m) = h$.
- **Diffie-Hellman**: Public key agreement
  - Allows two parties to agree on a secret key k.

Cryptographic building blocks

Security goals

- **Confidentiality**: Data exchanged between Client and Server is only known to those parties.
- **Authentication**: If Server receives data from Client, then Client sent it to Server.
- **Integrity**: If Server modifies data owned by Client, Client can find out.

Examples

- **Confidentiality**: When you send a private message on Signal, only you and the recipient can read the content.
- **Authentication**: When you receive an email from your boss, you can verify it actually came from them.
- **Integrity**: Your computer can verify that software update downloads haven't been tampered with during transmission.

Security goals: more examples

- **TLS (HTTPS)** ensures that data exchanged between the client and the server is confidential and that parties are authenticated.
  - Allows you to log into gmail.com without your ISP learning your password.
- **FileVault 2** ensures data confidentiality and integrity on your MacBook.
  - Prevents thieves from accessing your data if your MacBook is stolen.
- **Signal** implements post-compromise security, an advanced security goal.
  - Allows a conversation to “heal” in the event of a temporary key compromise.
  - More on that later in the course.

Why bother?

- Can’t we just use access control?
- Strictly speaking, usernames and passwords can be implemented without cryptography...
- Server checks if the password matches, or if the IP address matches, etc. before granting access.
- What’s so bad about that?

The Problem with Traditional Access Control

- Requires trusting the server completely
- No protection during transmission
- No way to verify integrity
- No way to establish trust between strangers

The magic of cryptography

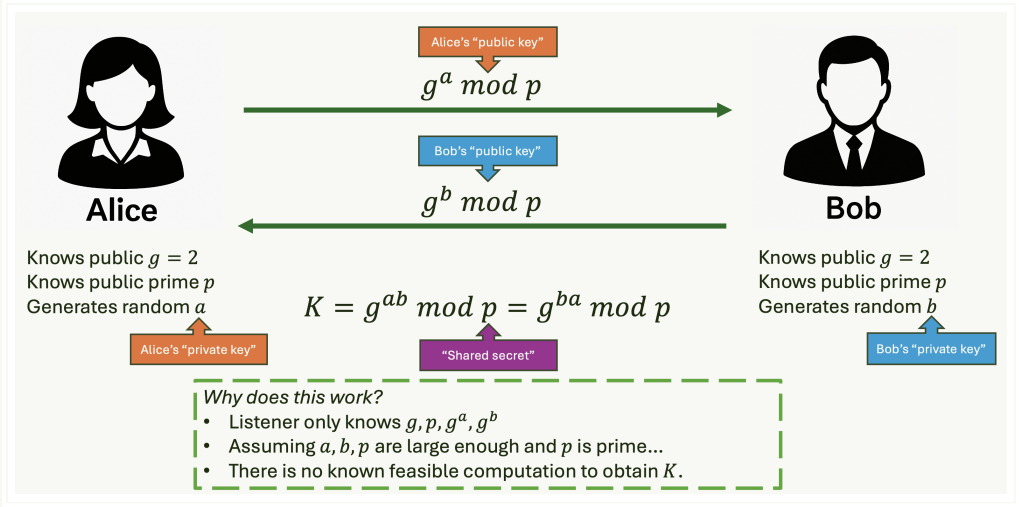
Cryptography lets us achieve what seems impossible

- Secure communication over insecure channels
- Verification without revealing secrets
- Proof of computation without redoing it

Hard problems

- Cryptography is largely about equating the security of a system to the difficulty of solving a math problem that is thought to be computationally very expensive.
- With cryptography, we get security systems that we can literally mathematically prove as secure (under assumptions).
- Also, this allows for actual magic.
  - Alice and Bob meet for the first time in the same room as you.
  - You are listening to everything they are saying.
  - Can they exchange a secret without you learning it?

Time for actual magic



No known feasible computation

- The discrete logarithm problem:
  - Given a finite cyclic group G, a generator $g \in G$, and an element $h \in G$, find the integer x such that $g^x = h$
- In more concrete terms:
  - Let p be a large prime and let g be a generator of the multiplicative group $\mathbb{Z}_p^*$ (all nonzero integers modulo p).
  - Given:
    - $g \in \mathbb{Z}_p^*$, $h \in \mathbb{Z}_p^*$
    - Find $x \in \{0, 1, \ldots, p-2\}$ such that $g^x \equiv h \pmod{p}$
  - This problem is believed to be computationally hard when p is large and g is a primitive root modulo p.
    - "Believed to be" = we don't know of any way to do it that doesn't take forever, unless we have a strong, stable quantum computer (Shor's algorithm)

Hard problems

Asymmetric Primitives

- Diffie-Hellman, RSA, ML-KEM, etc.
- “Asymmetric” because there is a “public key” and a “private key” for each party.
- Algebraic, assume the hardness of mathematical problems (as seen just now.)

Symmetric Primitives

- AES, SHA-2, ChaCha20, HMAC...
- “Symmetric” because there is one secret key.
- Not algebraic but unstructured, but on their understood resistance to n years of cryptanalysis.
- Can act as substitutes for assumptions in security proofs!
  - Example: hash function assumed to be a “random oracle”

Kerckhoff's principle

- *"A cryptosystem should be secure even if everything about the system, except the key, is public knowledge."* — Auguste Kerckhoffs, 1883
- **Why it matters**:
  - No "security through obscurity"
  - The key is the only secret: the rest can be audited, tested, trusted
  - Encourages open standards and peer review
  - If your system's security depends on nobody knowing how it works, it's not secure.

Symmetric primitive example: hash functions

Hash Function Properties

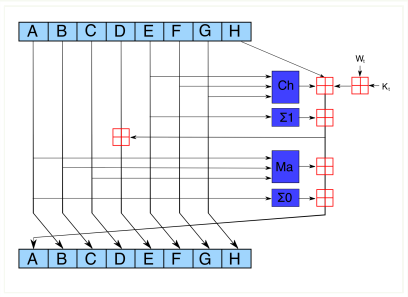
- Takes input of **any size**[<+->]
- Produces output of **fixed size**
- Is **deterministic** (same input → same output)
- Even a **tiny change** in input creates completely different output
- Is **efficient** to compute

```
SHA256(hello) =
2cf24dba5fb0a30e26e83b2ac5
b9e29e1b161e5c1fa7425e7304
3362938b9824
SHA256(hullo) =
7835066a1457504217688c8f5d
06909c6591e0ca78c254ccf174
50d0d999cab0
```

Note: One character change → completely different hash!

Expected properties of a hash function

- **Collision resistance**: computationally infeasible to find two different inputs producing the same hash.
- **Preimage resistance**: given the output of a hash function, it is computationally infeasible to reconstruct the original input.
- **Second preimage resistance**: given an input and an output, it's computationally infeasible to find another different input producing the same output.



SHA-2 compression function. Source: Wikipedia

Hash functions: what are they good for?

- **Password storage**: Store the hash of the password on the server, not the password itself. Then check candidate passwords against the hash.
- **Data integrity verification**: Hash a file. Later hash it again and compare hashes to check if the file has changed, suffered storage degradation, etc.
- **Proof of work**: Server asks client to hash something a lot of times before they can access some resource. Useful for anti-spam, Bitcoin mining, etc.
- **Zero knowledge proofs**: time for more actual magic

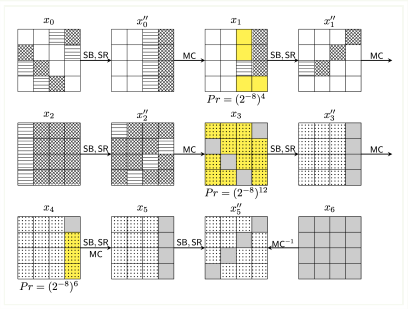
Time for more actual magic

- **Zero-knowledge proofs** allow you to prove that you know a secret without revealing any information about it.
- They built “zero-knowledge virtual machines” where you can execute an entire program that runs as a zero-knowledge proof.
- ZKP battleship game: server proves to the players that its output to their battleship guesses is correct, without revealing any additional information (e.g. ship location).

Battleship board game. Source: Hasbro

Evaluating a hash function's quality

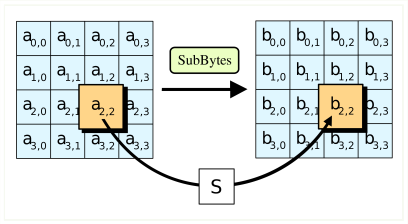
- **Recall**:
  - **Asymmetric primitives** are based on mathematical problems, can be mathematically proven secure (given assumptions!)
  - **Symmetric primitives** (encryption, hashing...) are statistically, empirically, heuristically shown to be secure, not proven secure.
  - The more cryptanalysis they survive, the higher confidence we have in their security.



Cryptanalysis of AES.

What about encryption?

- Symmetric primitive of choice for encryption: **AES**.
- Not that far off in terms of design process from hash functions, but:
  - AES is a PRP (pseudorandom permutation)
  - HMAC-SHA256 is a PRF (pseudorandom function)



AES's SubBytes operation. Source: Wikipedia

PRF versus PRP

Pseudo-Random Function (SHA-2)

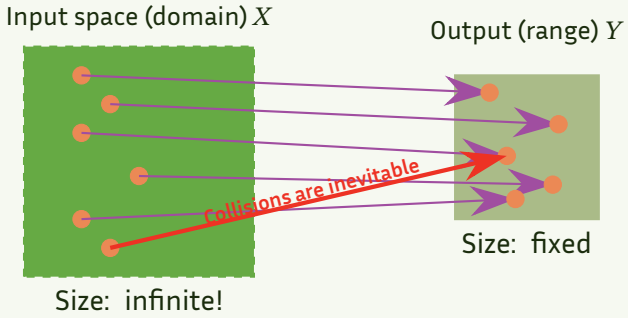
- **Input** is arbitrary-length,
- **Output** is fixed-length, looks random (as discussed earlier).
- Indistinguishable from a truly random function by an adversary with limited computational power.

Pseudo-Random Permutation (AES)

- **Input and output** are the same length, forming a permutation.
- Each input maps uniquely to one output, allowing invertibility.
- Indistinguishable from a truly random permutation by an adversary with limited computational power.

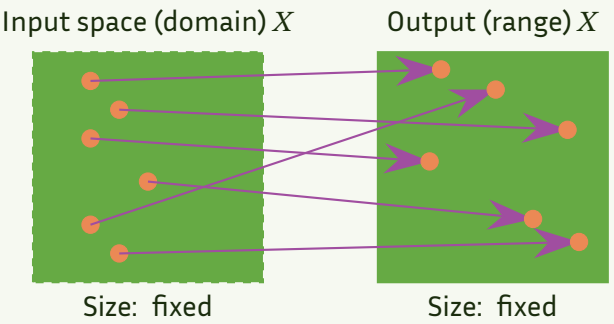
PRF : $F_k = X \rightarrow Y$

- We want the mapping to be:
  - One-way
  - “Randomized”
  - Relations between inputs not reflected in outputs



PRP : $F_k = X \rightarrow X$

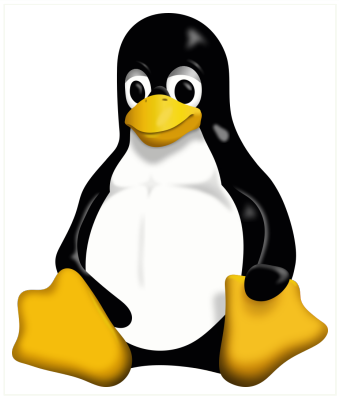
- **Bijective** (two-way)
  - **Injective**: no two inputs map to same output (no collisions)
  - **Surjective**: Every output has one corresponding input
- "Randomized"
- Relations between inputs not reflected in outputs



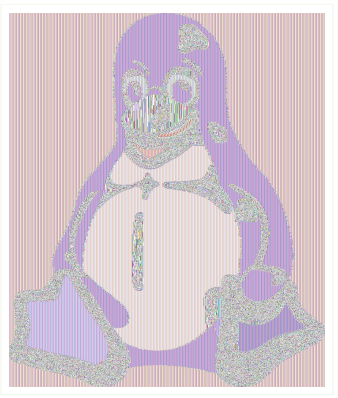
AES is a block cipher

- AES takes a 16-byte input, produces a 16-byte output.
- Key can be 16, 24 or 32 bytes.
- OK, so what if we want to encrypt more than 16 bytes?
- **Proposal**: split the plaintext into 16 byte chunks, encrypt each of them with the same key.

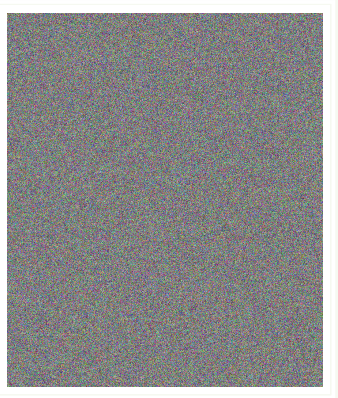
Block cipher examples



What we start with

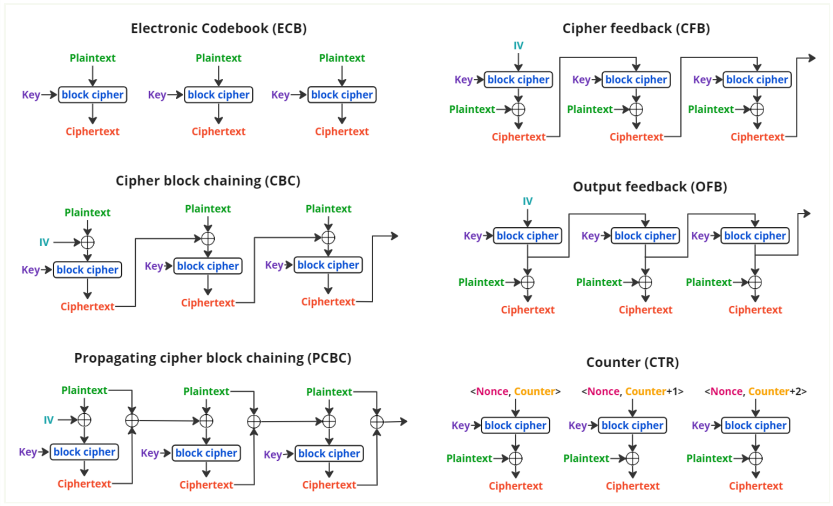


What we get



What we actually want

Block cipher modes of operation



Source: Wikipedia

Cryptographic building blocks

Security goals

- **Confidentiality**: Data exchanged between Client and Server is only known to those parties.
- **Authentication**: If Server receives data from Client, then Client sent it to Server.
- **Integrity**: If Server modifies data owned by Client, Client can find out.

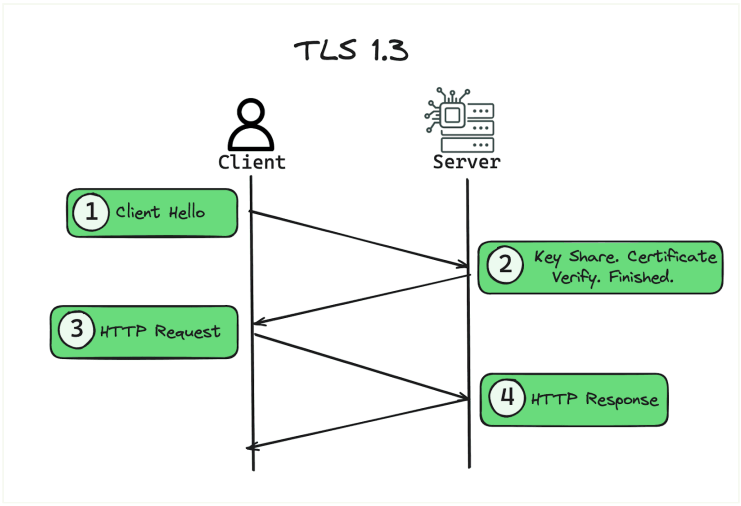
Examples

- **Confidentiality**: When you send a private message on Signal, only you and the recipient can read the content.
- **Authentication**: When you receive an email from your boss, you can verify it actually came from them.
- **Integrity**: Your computer can verify that software update downloads haven't been tampered with during transmission.

Security goals: more examples

- **TLS (HTTPS)** ensures that data exchanged between the client and the server is confidential and that parties are authenticated.
  - Allows you to log into gmail.com without your ISP learning your password.
- **FileVault 2** ensures data confidentiality and integrity on your MacBook.
  - Prevents thieves from accessing your data if your MacBook is stolen.
- **Signal** implements post-compromise security, an advanced security goal.
  - Allows a conversation to “heal” in the event of a temporary key compromise.
  - More on that later in the course.

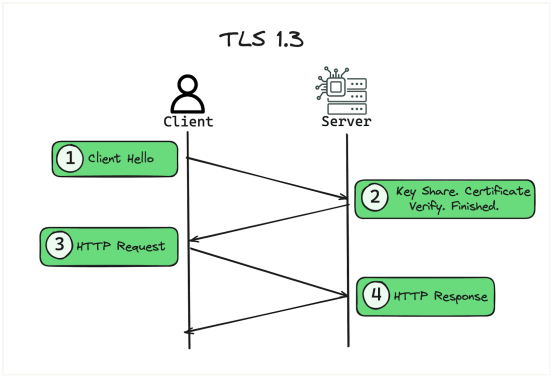
TLS 1.3: high-level sketch



Source: Mostafa Ibrahim

TLS 1.3: high-level sketch

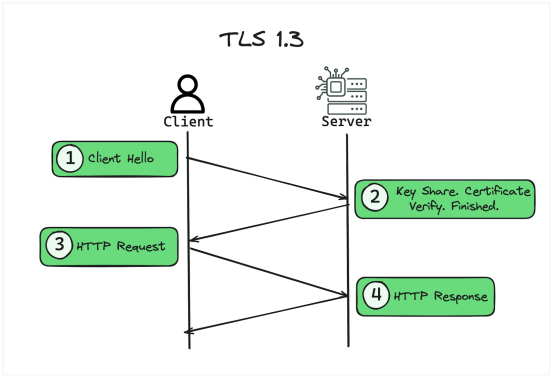
- **Public key agreement** (eg. Diffie-Hellman) is used to establish a shared secret between the client and the server.
- **AES** is used for encrypting data in transit.
- **SHA-2** is used for hashing (checking certificates, etc.)



Source: Mostafa Ibrahim

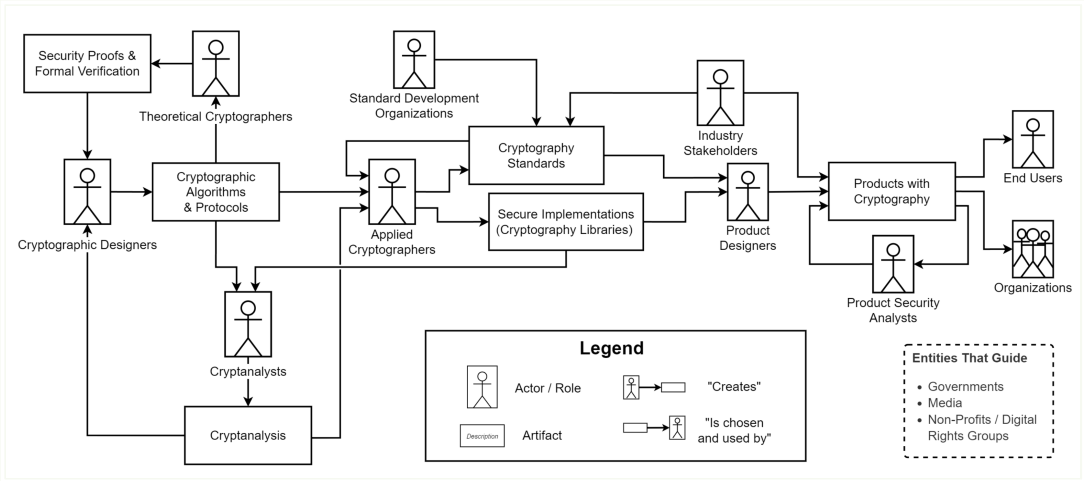
TLS 1.3: high-level sketch

- Through the design, we accomplish the desired **security goals** under a well-specified **threat model**:
- **Security goals**: confidentiality of data, authentication of the server towards the client...
- **Threat model**: malicious Internet Service Provider (ISP), etc.

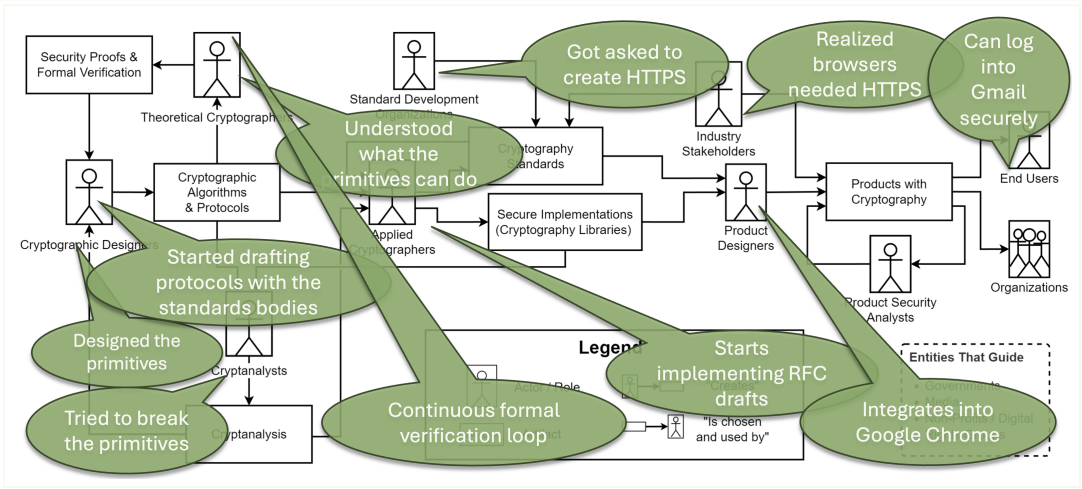


Source: Mostafa Ibrahim

How TLS 1.3 was made



How TLS 1.3 was made



From hard problems to real-world security

The journey we'll trace

1. **Mathematical insight**: Discrete logarithm is hard to compute.
2. **Cryptographic innovation**: Diffie-Hellman key exchange leverages this hardness.
3. **Real-world impact**: Secure communication for billions of people daily.

This is the power of applied cryptography: transforming abstract mathematical problems into tools that help people and protect our digital lives.

Course goals

- Understand the reasoning behind the math of modern cryptography.
- Analyze and prove the security of cryptographic constructions.
- Understand how cryptographic constructions can be composed to build real-world secure protocols and systems.
- Discern between theoretical cryptography and applied cryptography from an engineering perspective.
- Critically assess security implementations and evaluate real-world cryptographic protocols.
- Gain an understanding of the future of cryptography and its role in emerging technologies.

Course prerequisites

- Good but optional: CMPS 215 (Theory of Computation)
- If you want to understand whether you have the sufficient background for this course, review this revision chapter and try to do all the exercises: `https://joyofcryptography.com/pdf/chap0.pdf`

Class materials

- **Joy of Cryptography**: learn how to reason about and prove systems secure.
- **Attack papers, codebases, labs**: hard engineering perspective.

- **Always keep an eye on the website:** Course news, updates, materials, slides will all be posted there. `https://appliedcryptography.page`
- I am aiming for the most engaging course possible!

AMERICAN UNIVERSITY OF BEIRUT

Applied Cryptography

CMPS 297AD/396AI

Fall 2025

Part 1: Provable Security

1.1: Introduction

Nadim Kobeissi

https://appliedcryptography.page