AMERICAN UNIVERSITY OF BEIRUT

Applied Cryptography

CMPS 297AD/396AI

Fall 2025

Part 1: Provable Security

1.1: Introduction

Nadim Kobeissi

https://appliedcryptography.page

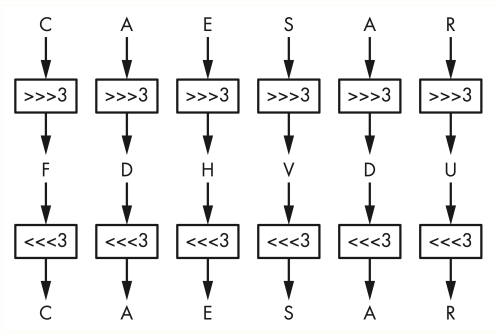
Section 1

Defining Cryptography

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."

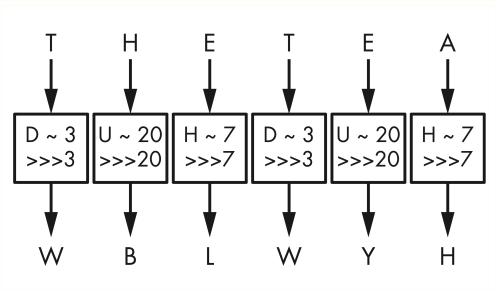


Source: Serious Cryptography, 2nd Edition

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

Pull out your phone!

Let's count the cryptographic operations happening right now:

- WiFi connection (WPA3)
- Cellular connection (5G AES)
- App notifications (TLS)
- Face/Touch ID (Secure Enclave)
- Background app refreshes

Real-time calculation

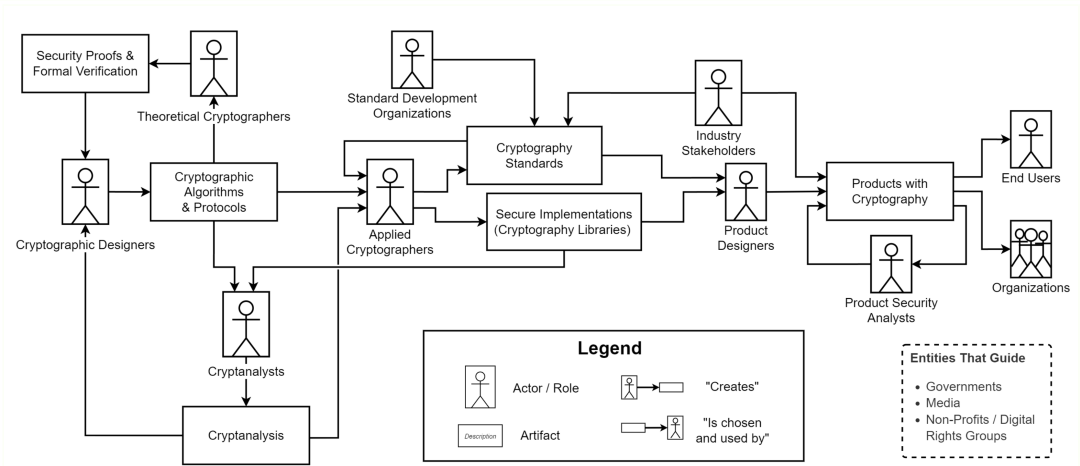
- Average smartphone: 100+ crypto operations/second
- In this 75-minute class: 450,000+ operations
- By end of semester: Billions of operations

You're already a crypto user!

Cryptography is everywhere

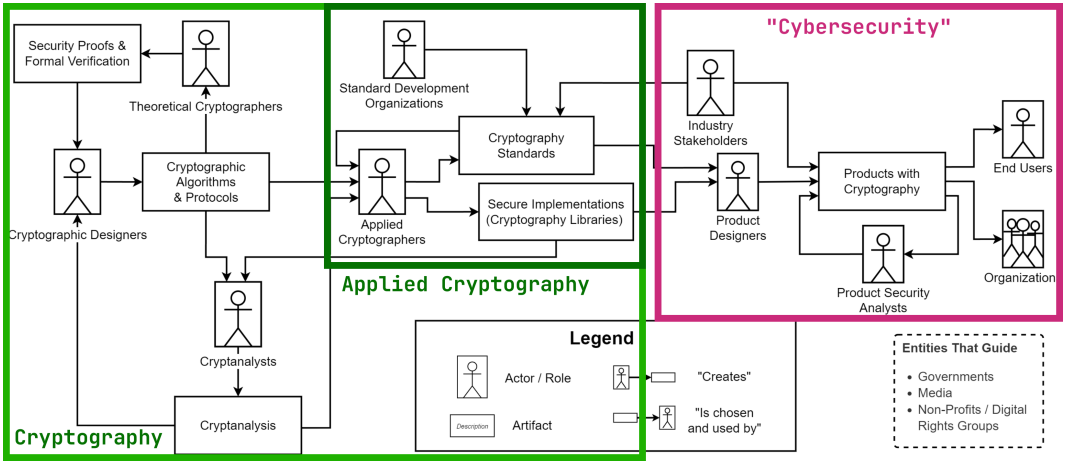
- Banking
- Buying stuff from the store
- Any digital payment system
- Messaging (WhatsApp, Signal, iMessage, Telegram)
- Voice calls
- Government and military systems
- SSH
- VPN access
- Visiting most websites (HTTPS)
- Disk encryption
- Cloud storage
- Video conferencing
- Unlocking your (newer) car
- Identity card systems
- Ticketing systems
- DRM solutions
- Private contact discovery
- Cryptocurrencies
- That Apple Photos 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 and WhatsApp** implement 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
- Prove information is true without revealing it
- Proof of computation without redoing it

Section 2

Examples of the Magic in Cryptography

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

Setup

- Public parameters: $p = 13, g = 2$
- Alice picks secret: $a = 5$
- Bob picks secret: $b = 7$

Public Exchange

- Alice computes: $A = g^a \bmod p = 6$
- Bob computes: $B = g^b \bmod p = 11$
- Alice sends $A = 6$ to Bob
- Bob sends $B = 11$ to Alice

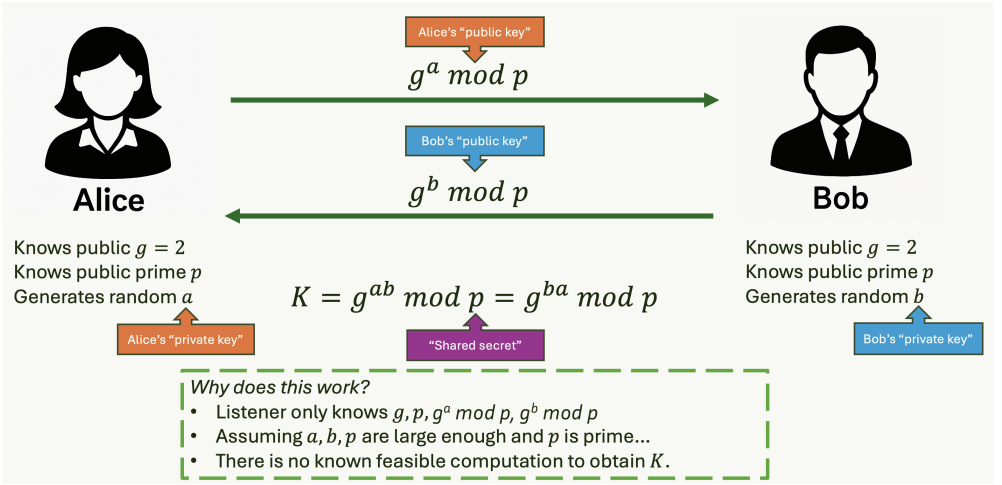
Shared Secret Computation

- Alice computes:
  $s = B^a \bmod p = 11^5 \bmod 13$
  - $= 161051 \bmod 13 = 9$
- Bob computes:
  $s = A^b \bmod p = 6^7 \bmod 13$
  - $= 279936 \bmod 13 = 9$
- **Shared secret:** $s = 9$

Eavesdropper sees only:

- $p = 13, g = 2, A = 6, B = 11$
- (In the real world, p, a and b are much larger numbers)

Time for actual magic

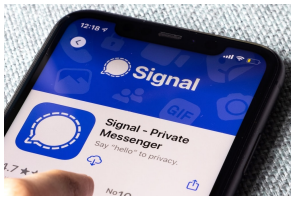


No known feasible computation

- 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)

Signal's double ratchet: DH everywhere

- **Initial key exchange**: Uses X3DH (Extended Triple DH)
  - Combines **three** DH key exchanges for security.
  - Works even when recipient is offline (*"asynchronous"* protocol).[a]
- **Ongoing communication**: Uses Double Ratchet
  - New DH key exchange for every message!
  - Provides "forward secrecy" and "post-compromise security".
  - If your phone gets compromised today, yesterday's messages remain secure.
  - If your phone recovers from compromise, tomorrow's messages are secure again.



Signal uses DH key exchange dozens, hundreds of times per conversation.

[a] Everything on this slide will be covered in much more detail later in the course.

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

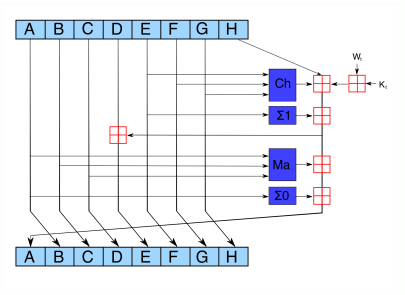
SHA256(hello) =
2cf24dba5fb0a30e26e83b2ac5
b9e29e1b161e5c1fa7425e7304
3362938b9824

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?

- **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

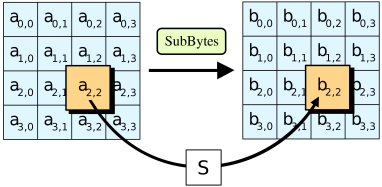
Time for more actual magic

- **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

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)

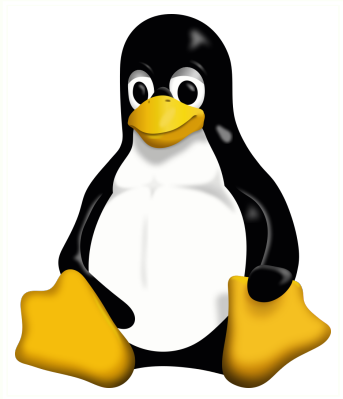


AES's SubBytes operation. Source: Wikipedia

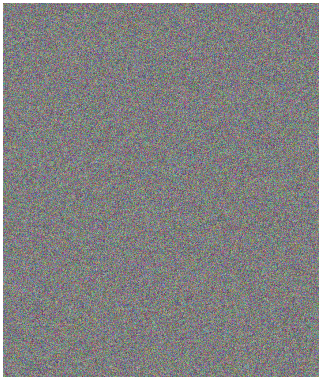
AES is a block cipher

- 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.

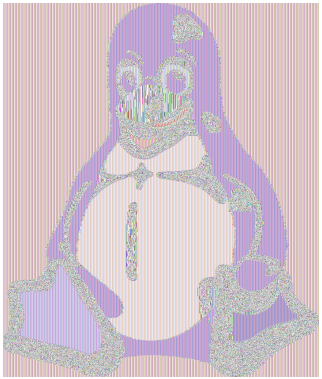
Block cipher examples



What we start with

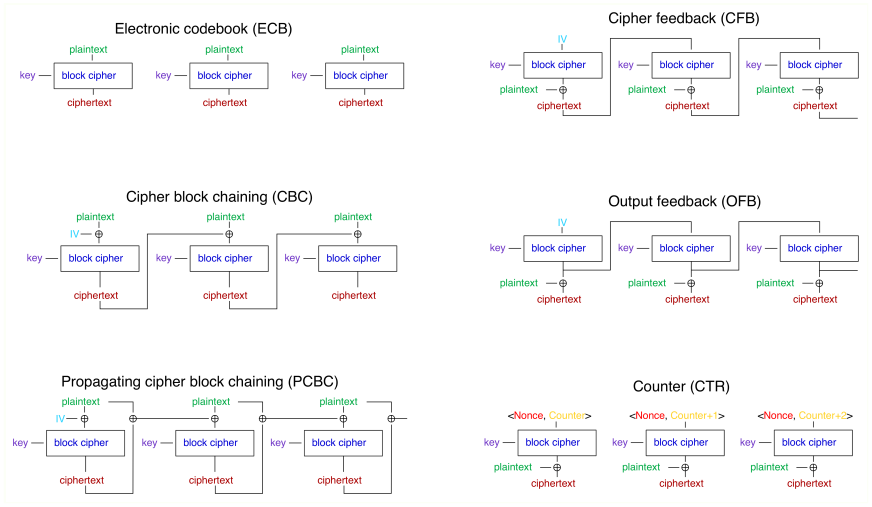


What we want



What we actually get

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.

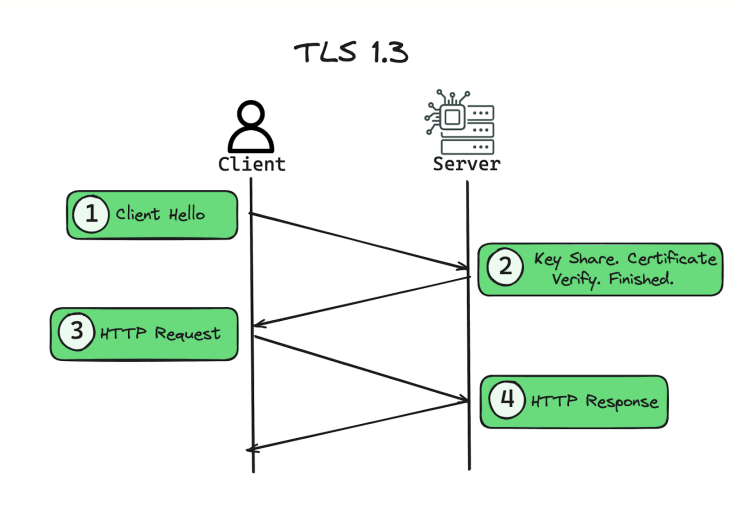
Examples

- **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 and WhatsApp** implement 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.

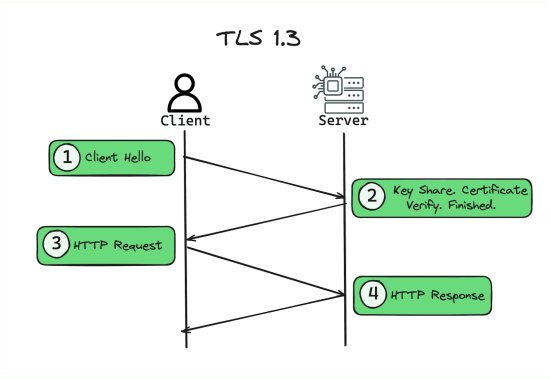
TLS 1.3: high-level sketch



Source: Mostafa Ibrahim

TLS 1.3: high-level sketch

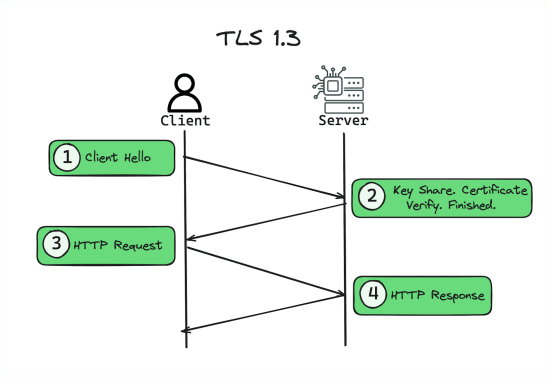
- **Public key agreement** (eg. Diffie-Hellman) is used to establish a shared secret between the client and the server.
- **AES** is used for encrypting data in transit.
- **SHA-2** is used for hashing (checking certificates, etc.)



Source: Mostafa Ibrahim

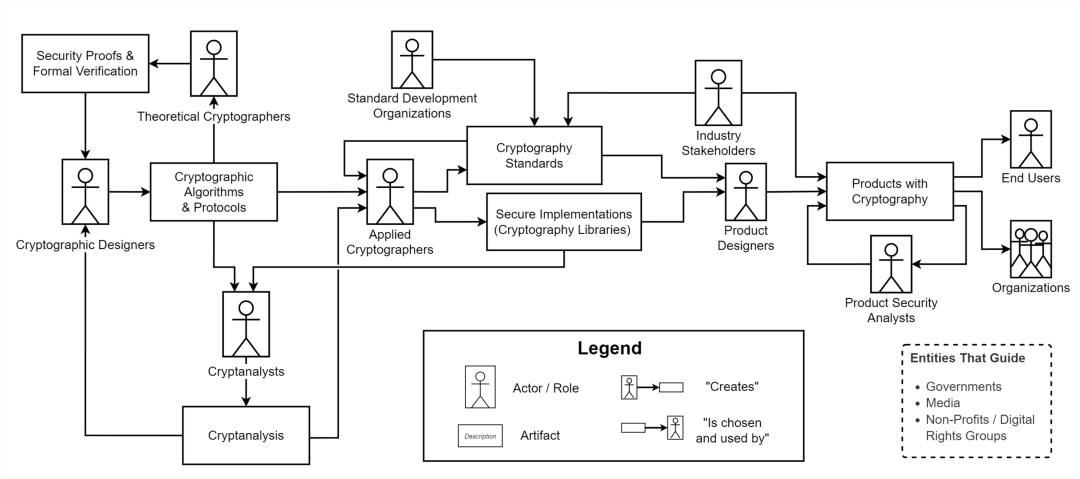
TLS 1.3: high-level sketch

- 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.

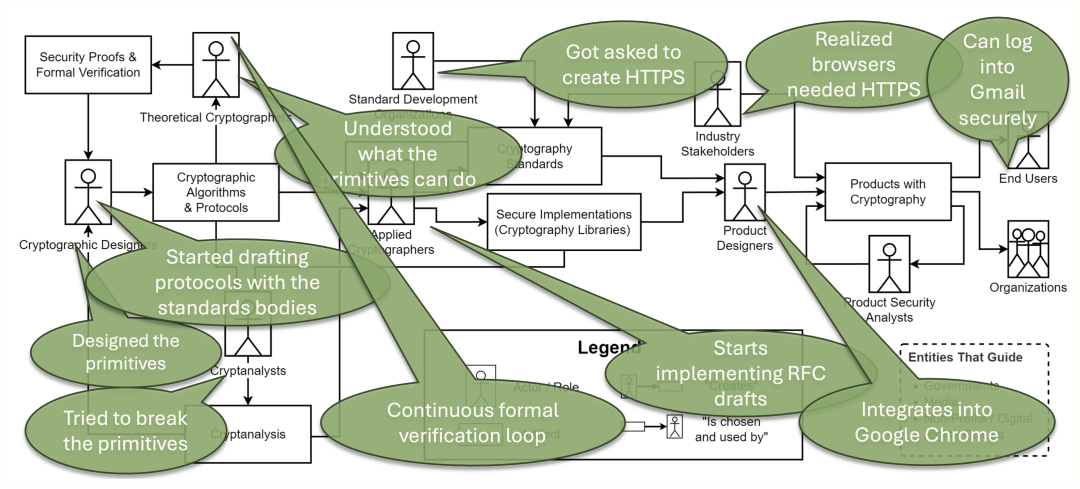


Source: Mostafa Ibrahim

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.

Section 3

Course Goals

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 resources

- **Joy of Cryptography**: learn how to reason about and prove systems secure.
- **Attack papers, codebases, projects**: 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!

Section 3

Course Goals

Subsection 3.1

Projects

Projects: hands-on learning

- **Weekly project sessions**: Hands-on complement to lectures.
- **Team structure**: Groups of 1-3 students.
- **Project selection**: Pick at most **two project topics** to work on throughout the entire course.
- **What you'll do**:
  - Experiment with real-world cryptographic libraries.
  - Simulate attacks and vulnerabilities.
  - Understand why certain security practices are necessary.
  - Use formal analysis tools.

Project sheets available on the course website.
Pick one or suggest your own!

Project A: Designing a Password Manager

Build Your Digital Vault!

Create a fortress for your passwords using real cryptographic techniques!

- Master password protection with key derivation
- Encrypted storage that even you can't break
- Secure password generation algorithms

What you'll learn:

- How password managers *actually* work
- Defend against password cracking attacks
- Memory scraping countermeasures
- Why "forgot master password" button doesn't exist

Never forget a password again... except one!

Project B: Designing a Secure Messenger

Build the Next Signal!

Create your own end-to-end encrypted messaging app from scratch!

- Implement the Double Ratchet protocol
- Perfect forward secrecy by default
- Group messaging that stays private

Real cryptography in action:

- Key exchange protocols that feel like magic
- Deniable authentication (plausible deniability!)
- Metadata protection techniques
- Secure key storage on real devices

Your messages, truly private!

Project C: Protocol Modeling with ProVerif

Prove It Secure!

Design and mathematically verify your own TLS-like protocol!

- Formal verification with ProVerif
- Specify security properties rigorously
- Find attacks before hackers do

Become a protocol designer:

- Design protocols that are *provably* secure
- Learn how TLS was formally verified
- Discover subtle attack patterns
- Master formal methods used by pros

If you can’t prove it, don’t ship it!

Project D: Zero-Knowledge Battleship Game

Cryptographic Gaming!

Build Battleship where cheating is mathematically impossible!

- Zero-knowledge proofs with RISC Zero
- Prove hits without revealing ship positions
- Verifiable gameplay, zero trust needed

The magic of ZK proofs:

- Build a zkVM from scratch
- Create proofs that feel impossible
- Learn by building an actual game
- Impress your friends with crypto magic!

You sunk my battleship... provably!

Project E: Post-Quantum Migration

Quantum-Proof the Future!

Prepare systems for the quantum computing era!

- Implement NIST's ML-KEM (Kyber)
- Deploy ML-DSA (Dilithium) signatures
- Design hybrid classical-quantum systems

Tomorrow's cryptography today:

- Migrate real protocols to PQC
- Analyze performance impacts
- Handle message size explosions
- Future-proof existing systems

Beat quantum computers at their own game!

Project F: Private Contact Discovery

Find Friends with Privacy!

Build WhatsApp-style contact discovery that's actually private!

- Private Set Intersection (PSI)
- Oblivious PRFs in action
- Zero leakage to servers

Real-world privacy tech:

- Protect address books from servers
- Prevent enumeration attacks
- Handle thousands of contacts efficiently
- Used by Signal and WhatsApp!

Your contacts, your business!

Project G: Privacy-Preserving Age Verification

Prove Your Age Privately!

Build ID checks that don't track you!

- Anonymous credentials
- Zero-knowledge age proofs
- Unlinkable verifications

Digital ID done right:

- Prove "over 18" without revealing birthdate
- Prevent tracking across services
- Mobile-friendly verification
- Pedersen commitments & Schnorr proofs

Old enough to enter, young enough for privacy!

Project H: Time-Locked Message Capsule

Send Messages to the Future!

Create digital time capsules that can't be opened early!

- Timelock encryption with drand beacon
- Threshold BLS signatures
- Unstoppable future reveals

Time travel for data:

- Messages that decrypt themselves
- Perfect for auctions & commitments
- Build on League of Entropy infrastructure
- No key escrow, no early access!

Encrypt today, decrypt tomorrow!

Create Your Own Cryptographic Adventure!

Be the Innovator!

Design your own cryptographic project from scratch!

- Novel protocol implementation
- Security analysis of real systems
- Formal verification challenges
- Your creative crypto idea!

The sky's the limit:

- Follow your cryptographic passion
- Solve a real problem you care about
- Combine techniques in new ways
- Potentially publishable research!

Your idea could change cryptography!

Section 3

Course Goals

Subsection 3.2

Problem Sets

Problem sets: theory meets practice

- **Already available**: All problem sets are posted on the course website.
- **Purpose**: Reinforce and deepen understanding of lecture material.
- **What to expect**:
  - Theoretical proofs and problem-solving.
  - Short coding tasks and computational experiments.
  - Bridge abstract concepts with concrete applications.
- **When are they due?** I'll tell you weeks in advance.

Check the website for all problem sets!

`https://appliedcryptography.page`

Section 3

Course Goals

Subsection 3.3

Graduate Students: Special Instructions

Graduate students: thesis integration

Special Requirements for Graduate Students

Graduate students are expected to discuss their thesis topic with me to develop an individual project that integrates with their research.

- **Individual project**: Tailored to your thesis research.
- **Integration**: Apply course concepts to your specific research area.
- **Consultation**: Schedule a meeting early in the semester to discuss your thesis and project ideas.
- **Outcome**: Deepen your understanding of cryptography within your research context.

Section 3

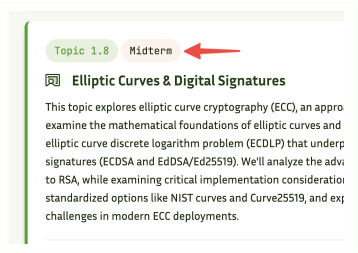
Course Goals

Subsection 3.4

Exams

Exam transparency: no surprises!

- **What's covered**: Specific topics, chapters, and concepts clearly outlined.
- **Format details**: Question types, duration, and grading scheme all disclosed.
- **Practice materials**: Sample questions are available through the problem sets.
- **No stress approach**: Focus on understanding concepts, not guessing what might appear.
- **If it wasn't mentioned in the slides or problem sets, then it's not on the exam.**



Check the website for complete exam information!

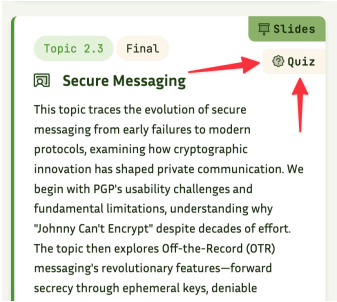
https://appliedcryptography.page

Self-assessment quizzes: test your understanding

Optional Learning Quizzes

Each class topic comes with an optional quiz to help you gauge your understanding.

- **Not graded**: purely for self-assessment
- **Immediate feedback**: check your answers right away
- **Fun and engaging**: designed to reinforce key concepts



Each topic has its own quiz on the website (except this introduction): https://appliedcryptography.page

Section 3

Course Goals

Subsection 3.5

The Key Exchange

The Key Exchange: optional weekly gathering

What is The Key Exchange?

An optional weekly gathering for students intending to become researchers or professionals in cryptography.

- Discuss cutting-edge research papers
- Practice presentation skills
- Develop scientific writing
- Explore career paths

Why join?

- Learn to read research papers efficiently
- Present technical concepts clearly
- Write like a cryptographer
- Navigate career decisions with confidence
- Build lasting connections with peers

The only prerequisite is your curiosity!

The Key Exchange: four-week rotating schedule

- **Week 1: Paper Deep Dive**
  - Read and analyze research papers
  - Learn efficient reading strategies
  - Discuss recent CRYPTO/Eurocrypt papers
- **Week 2: Student Presentations**
  - 15-minute talks on crypto topics
  - Practice with visual aids and demos
  - Receive constructive peer feedback
- **Week 3: Writing Workshop**
  - Technical writing exercises
  - Blog posts and security reports
  - LaTeX tutorials for papers
- **Week 4: Career Café**
  - Industry guest speakers
  - Graduate school applications
  - Interview preparation

When should we meet?

Let's find a time that works for everyone!

Section 3

Course Goals

Subsection 3.6

About Me

About me

- **Academic Background**
  - Ph.D. Computer Science, Inria Paris (2018)
  - Thesis: *"Formal Verification for Real-World Cryptographic Protocols"*
  - Advisors: Karthikeyan Bhargavan, Bruno Blanchet
- **Current Roles**
  - Visiting Professor, AUB (2025-2026)
  - Senior Applied Cryptography Auditor, Cure53
  - Director, Symbolic Software
- **Research & Practice**
  - 250+ security audits for major clients
  - Created Verifpal, Noise Explorer
  - Published at IEEE S&P, USENIX Security, RWC
  - Program Committee: CCS, NDSS, PETS
- **Fun Facts**
  - Active in cryptography for ≈15 years now
  - Published indie games on Steam & Nintendo

My goals for this course

My Commitment

"To provide you with an excellent applied cryptography education that prepares you for success in this field."

- High standards and quality teaching
- Your success is my priority
- We'll work together to achieve this

What I aim to deliver

- Strong foundation in cryptographic science and engineering
- Practical skills for research and industry
- Clear technical communication abilities
- A learning environment driven by **curiosity** and **intellectual excitement**
- Deep understanding over memorization

My expectations: be intellectually present

Minimum Requirements

- Attend all classes and exams, **on time**.
- Complete assignments, **on time**.

- **But I want you to do more:**
  - **Be curious**: Think deeply about what we're learning.
  - **Engage actively**: This is a conversation, not a monologue.
  - **Work hard**: Not because this is hard, but because you stand to learn so much!

- **Please, SPEAK UP!**
  - Don't understand something? **Ask!**
  - Have a question? **Voice it!**
  - Afraid of looking stupid? **Don't be!**
- **Why?**
  - I sometimes wasted weeks because my ego didn't allow me to ask "stupid" questions.
  - Don't worry: I've managed to make myself look dumber in front of the international cryptography community than you ever will.

Section 3

Course Goals

Subsection 3.7

About You

This is YOUR learning journey

What I need from you

- Think about what YOU want from this course.
- Express your interests and goals.
- Participate in shaping our discussions.
- I didn't come here just to watch you stare at me from a corner!

We're learning together

- This isn't just another course to check off.
- Applied cryptography is an exciting, specialized field.
- Your questions make the class better for everyone.
- Your engagement shapes the experience.

Let's make this journey extraordinary together!

Let’s get to know each other!

Please introduce yourself!

I’d love to learn more about each of you. Please share:

- Your name
- Your major and level (undergrad/grad)
- Your career aspirations
- What you hope to gain from this course

Why this matters

- Helps me tailor examples to your interests
- Enables better project matching
- Builds our classroom community
- Allows me to connect course material to your goals

AMERICAN UNIVERSITY OF BEIRUT

Applied Cryptography

CMPS 297AD/396AI

Fall 2025

Part 1: Provable Security

1.1: Introduction

Nadim Kobeissi

https://appliedcryptography.page