ISE Cryptography – Lecture 00

Module Information

The Lecturer

You probably know me already!

Stuff I've done...

- Founded Tracworx: 300k+ assets tracked, €33m under management, Fortune 500 clients
- Currently founding a new startup (watch this space)
- Assoc. Prof. (Practice) of Software Engineering at UL
- I lecture on software testing, data structures and algorithms, and cryptography at ISE
- Helped establish the ISE programme; served on Industry and Young Advisory Boards
- Raised €5m+ in venture and research funding
- Open-source libraries downloaded over 2 million times
- Sunday Independent 30 under 30

Commitments

- This isn't secondary school! This isn't a company either.
- Discussion, participation, hands-on learning is a must!
- I'm going to have slides, resources etc online in advance of each lecture.
- I'm going to have problem sets / exercises for most weeks.
- I'm also going to assign (short) readings from the texts sometimes.
- These aren't optional; future lectures will rely on it!
- Primary mode of communication should be Brightspace.
- Email is secondary, I'll block in "office hours" to answer.

The Module

What are we going to cover, and why?

Course outline

- What will we cover and how will it be assessed?
  - Module information
- What does it mean for a cipher to be secure?
  - Introduction to cryptography
- How can we encrypt data using only a short secret key?
  - Stream ciphers
- How do we securely encrypt messages of any size?
  - Block ciphers
- How can we detect if a message has been tampered with?
  - Message authentication codes

Course outline (continued)

- How can we create a unique, unforgeable fingerprint for any piece of data?
  - Hash functions and HMAC
- How can two parties communicate securely without a shared secret?
  - Public-key cryptography
- How can we prove who sent a message and build tamper-proof encryption?
  - Digital signatures, CCA, AEAD and HPKE
- How do we safely turn a password into a cryptographic key?
  - Passwords, keys and KDFs
- What happens to modern cryptography when quantum computers arrive?
  - Post-quantum cryptography

Assumptions and guidelines

- I'm assuming that...
  - ...you've no prior security/cryptography experience
  - ...you've been out on residency a few times
  - ...you know enough Python, C and JS to get by
- Lectures are 2-3 hours long, so expect a 10 minute break after every 45-50 minutes of class
  - I generally aim to start the break at a natural stopping point in the material
  - Remind me if I forget!
- Lectures are split between theory and practice, so be ready to get hands-on with discussions, exercises and code throughout

Maths

- This isn't Leaving Cert maths! No differentiation, no trigonometry, no integration
- Cryptography uses a different kind of maths: discrete, logical, and algebraic
  - Modular arithmetic (clock arithmetic: $7 + 8 = 3 \pmod{12}$)
  - XOR and bitwise operations (you'll be very comfortable with these by Week 2)
  - Probability (simple counting arguments, not continuous distributions)
  - Logical reasoning: "if this is secure, then that must also be secure"
- The proofs in this course are about understanding *why* something is secure
  - Not about memorising derivations or solving equations
  - If you can follow a logical argument step by step, you have what you need
- The notation may look unfamiliar at first ($\mathcal{M}$, $\mathcal{K}$, $\mathrm{SS}_{\mathrm{adv}}[\mathcal{A}, \mathcal{E}]$, $\Pr[\ldots]$)
  - But the concepts behind the symbols are accessible
  - We'll introduce every symbol before we use it

How to get the most out of this module

- This module is designed around active learning, not passive note-taking
  - Lectures mix theory, discussion, and hands-on coding
  - The style is meant to suit you, so speak up if something isn't working!
- **Ask questions** during class, not just after
  - If you're confused, someone else probably is too
  - There are no stupid questions, only stupid ciphers
- **Discuss** the material with your classmates
  - Explaining a concept to someone else is the fastest way to find gaps in your understanding
  - Study groups are strongly encouraged

- **Review and reinforce** after each lecture
  - Re-read slides, re-do examples, try the exercises before looking at solutions
  - Spaced repetition beats cramming every time
- **Target understanding over memorisation**
  - If you can explain *why* a construction is secure, you understand it
  - If you can only recite the definition, you don't (yet!)
- **Spend your time on hard problems**, not easy ones
  - If an exercise feels comfortable, move on to the next one
  - The struggle is where the learning happens

Assessment

- Weekly quizzes and/or labs (20% overall, best $n - 2$ of n)
- Midterm in class on Wed Week 4 (TBD) at 09:00 (40% overall)
- Final in class on Wed Week 7 (TBD) at 09:00 (40% overall)
  - These three components make up your Block 2.4a grade
- Joint EPIC project during Weeks 8, 9 and 10
  - Cryptography is one of the topics assessed by the EPIC
  - This makes up your Block 2.4b grade
- Attendance isn't graded, but you should consider it mandatory
  - If you're not going to be here, notify me via email

Continuous assessment

- Goals:
  - Motivate attendance and learning
  - Build competence and confidence with the material
  - Make the midterm and final easier with practice
  - Build a bank of marks to fall back on: get the full 20 marks and you only need 60 out of 80 on the exams for an A1
- What if you miss a week, or mess up a quiz/lab?
  - You've basically got two "free" misses: the best $n - 2$ of n quizzes/labs count towards your grade
- Plan of action: quizzes in weeks 2, 3, 5 and 6.
  - That way, there's no quiz when we've got an exam anyway!
- Take-home labs most weeks, due before the next lecture.

EPIC project advice

- The EPIC project runs during Weeks 8, 9 and 10 and makes up your Block 2.4b grade
- Cryptography is one of several topics assessed
  - You won't need to code a full cryptosystem from scratch!
- Focus on demonstrating that you can **choose appropriate primitives** for a given problem (e.g. AEAD for confidentiality + integrity, HKDF for key derivation)
- Show that you understand **why** you chose them, not just **how** to call the API
- Avoid rolling your own crypto – use well-established libraries (e.g. `cryptography` in Python, Web Crypto API in JS)

AI and Learning

How to use it without fooling yourself

What the research says

- We're scientists... bargain-basement scientists, but scientists nonetheless!
  - So let's deal with evidence, not vibes...
- A carefully designed AI tutor **doubled learning gains** in a Harvard physics course vs. active classroom learning (randomised controlled trial, 194 students)
- But when Turkish high-school students got unrestricted ChatGPT access, they solved **48% more practice problems** yet scored **17% worse on the exam** without it
- The difference? Whether **you** do the thinking...
  - ...or the AI does!
- AI is a powerful learning tool.
  - It's also a powerful *learning-avoidance* tool!
  - The rest of this section is about telling the two apart.

Use AI to think harder

- **Explain a concept to the AI**, then ask it to find gaps in your explanation.
  - If you can teach it, then you understand it.
  - ...even if your student is a glorified rubber duck.
- **Do the problem yourself first**, then ask the AI to critique your work.
  - Producing an answer yourself creates stronger memory traces than reading one.
- **Ask AI to generate practice questions** and quiz you.
  - Retrieval practice is pretty much the single most effective study technique.
  - Spaced repetition makes it even more effective, so go back to the same questions after a few days.
- **Ask “why”, “why not” and “what if”**: use AI as a Socratic partner.
  - “Why does CBC need an IV?” beats “Explain CBC mode” every time.
  - Be curious about edge cases and limitations!

The traps to avoid

- **The practice illusion**: getting through more problems with AI feels productive...
  - but if AI supplied the reasoning, you haven't learned it.
  - That's the 48%-more-practice, 17%-worse-on-exam finding.
- **Copy-paste escalation**: an MIT study tracked users over 4 months and found they steadily shifted toward copying AI output verbatim...
  - ...with measurable declines in brain activity associated with original thinking.
- **Skipping the struggle**: the difficulty you feel when working through a problem is where learning actually happens.
  - A controlled trial found an 11-point retention gap at 45 days when students had unrestricted AI access.
- **Beginners are most at risk**: a meta-analysis of 17+ studies found that novices who relied on AI failed to build foundational problem-solving skills.
  - Experts benefited; beginners didn't.

The Texts

Read them or weep.

Course texts

- Obviously, no one here would ever pirate college textbooks...
  - But who wants to spend €140 on a fancy doorstop?
  - So I've picked open-source texts instead!
- Download these during/after today's lecture. I'll be assigning reading from both of them throughout the course.
  - **Spoiler alert:** assigned reading is examinable!
- *A Graduate Course in Applied Cryptography*, v0.6 (2023), Dan Boneh and Victor Shoup
  - **/dæn boʊˈneɪ/** and **/ˈvɪktər ʃuːp/**
  - The core text for the course, covering all the material in depth with rigorous proofs and exercises
- *Crypto 101*, Laurens Van Houtven (lvh)
  - **/ˈlɔːrəns vɑn ˈhɑʊtvən/**
  - A more accessible introduction to the basics of cryptography for software engineers
  - Used as a supplementary text

Questions?

Ask now, catch me after class, or email eoin@eoin.ai