ISE Cryptography – Lecture 08

HPKE

Final Exam

- Decent performance overall, moderately bimodal distribution
- Longer tails than the midterm and a larger IQR (29.5%)
- Mean 62.1%, standard deviation 19.2%

Housekeeping

- Overall grades are now in!
  - Mean 66.8%, median 70.0%, standard deviation 14.4%, IQR 20.9%
  - Moderate bimodality
  - Continuous assessment was generally the strongest component
- Focus is on the EPIC
  - How much talking I do today is up to you!
  - Some “ideas” slides for high-level elements
  - Beyond that, tell me what you’d like me to cover!
- Some content on HPKE (RFC 9180) which may be useful for the project
- Planning to spend some time with each group here today
  - Can review and comment on cryptographic design, can’t design it for you
  - “Eoin said” isn’t a justification that you can use in the interview
- I may make some notes on each group for my own reference

EPIC Advice

- Cryptography grading for the EPIC follows a very simple schema:
  - Unsatisfactory ⇒ D/F
  - Satisfactory ⇒ C
  - Good ⇒ B
  - Excellent ⇒ A2
  - Outstanding ⇒ A1
- Pay attention to the spec and make sure you do what I've asked you to do!
  - Get the core features working first before you try anything else
  - Adding bells and whistles doesn't help if the rest of it is garbage
- Remember, when you walk into the interview, you've got 0%
  - Aim to convince me that I can should you a high grade
  - Waffling isn't worth anything, and just eats up valuable time
  - Be prepared to defend your choices!

HPKE

A crash course in RFC 9180

HPKE

- Hybrid Public Key Encryption (HPKE) is defined in RFC 9180
  - Plug-and-play public-key encryption framework
- Why bother defining it? Lots of homebrew hybrid cryptosystems out there...
  - Lots of bad hybrid cryptosystems!
  - Outdated primitives
  - Lack of proof of IND-CCA2 security
  - No test vectors
- HPKE eliminates one-off “ECIES-ish” constructions
  - Easy to audit and reuse
  - Supports any KEM + KDF + AEAD trio (sort of)
  - Already powering TLS-ECH, OHTTP, MLS and more!

HPKE Ciphersuites

- Each HPKE ciphersuite needs three principal components:
  - A key encapsulation mechanism (KEM)
  - A key derivation function (KDF)
  - An AEAD encryption algorithm
- The usual assumptions about security apply to each of those components!
  - Nothing particularly new here
  - Familiar primitives used to make a larger construct
- A HPKE ciphersuite can be identified by the triple of algorithms:
  - E.g. DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, AES-128-GCM
  - The standard has a specific method to encode suite IDs
  - You probably don't need to worry about that for now!

HPKE Ciphersuites

- IANA maintains a public registry for each of the component types
  - IANA HPKE Registry
  - Maps a 16-bit identifier to algorithms in each category
- KDFs and AEADs are more limited
  - KDF restricted to HKDF-SHA256 and so on
  - AEAD restricted to AES-GCM and ChaCha20-Poly1305
- Lots of possible KEMs, though!
  - Various curves for elliptic curve Diffie-Hellman
  - Newer options like the X25519/Kyber-768 PQ/T hybrid
  - And the ML-KEM family
- Up to you to negotiate what algorithms will be used
  - Be wary of downgrade attacks!

HPKE Modes

Mode	Extra key material	Sender authentication	Typical use case
Base (0x00)	–	✖	Simple message encryption
PSK (0x01)	PSK + PSK_ID	PSK only	IoT bootstrapping
Auth (0x02)	Sender static key	✔	Secure logging
Auth-PSK (0x03)	Both	✔	Hardened channels

Security

- HPKE has decent security guarantees
  - IND-CCA2 for all modes
  - Relies on AEAD integrity & non-repeating nonces
- Forward secret-ish:
  - Has forward secrecy with respect to sender compromise
  - No forward secrecy with respect to recipient compromise
- No attempt made to hide message length
  - Pre-pad plaintexts if you need this property!
- Replay attacks fall into the “it depends” category

HPKE for the EPIC

- If you're planning to use HPKE for the EPIC...
  - Read the RFC
  - Get hands-on and try a tutorial
- There are lots of dedicated HPKE libraries out there
  - Constructing it from OpenSSL or Web Crypto API primitives, for example
- Even if you're not using it directly, it's a good primer on hybrid cryptography
  - Fundamentally no different from our generic PKE example
  - Mixed in with KEK/DEK ideas and key derivation

Key Commitment

AEAD with an extra guarantee

Key Commitment

- Key commitment is a property that crops up in AEAD discussions
  - A ciphertext/tag pair binds to exactly one secret key
  - There is negligible probability that the same (c, t, d, n) decrypts successfully under two distinct keys
- Why is this useful?
  - Prevents key-substitution attacks
  - Useful in protocols that re-encrypt or “frank” messages
  - Or where adversaries can try many keys

GCM

- Galois/counter mode (GCM) isn't key-committing by default!
- An adversary who knows two keys k_1, k_2 can craft (c, t) that validates under both
  - This is called a multi-key forgery
  - Practical exploits on message-franking systems were demonstrated in 2022
- Luckily, there are some simple workarounds
  - It's easy to wrap GCM as KC-GCM to provide key commitment
- Adding an all-zero plaintext block to the start of the message also works
  - If the first block doesn't decrypt to zero $\rightarrow$ reject
  - The catch is making this constant time across both pathways!
- In the context of the EPIC, risks are relatively low
  - The delightfully-named invisible salamander attack is tricky to pull off
  - Having a ciphertext decrypt to different plaintexts under different keys isn't that useful for a 1:1 file sharing situation

Questions?

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