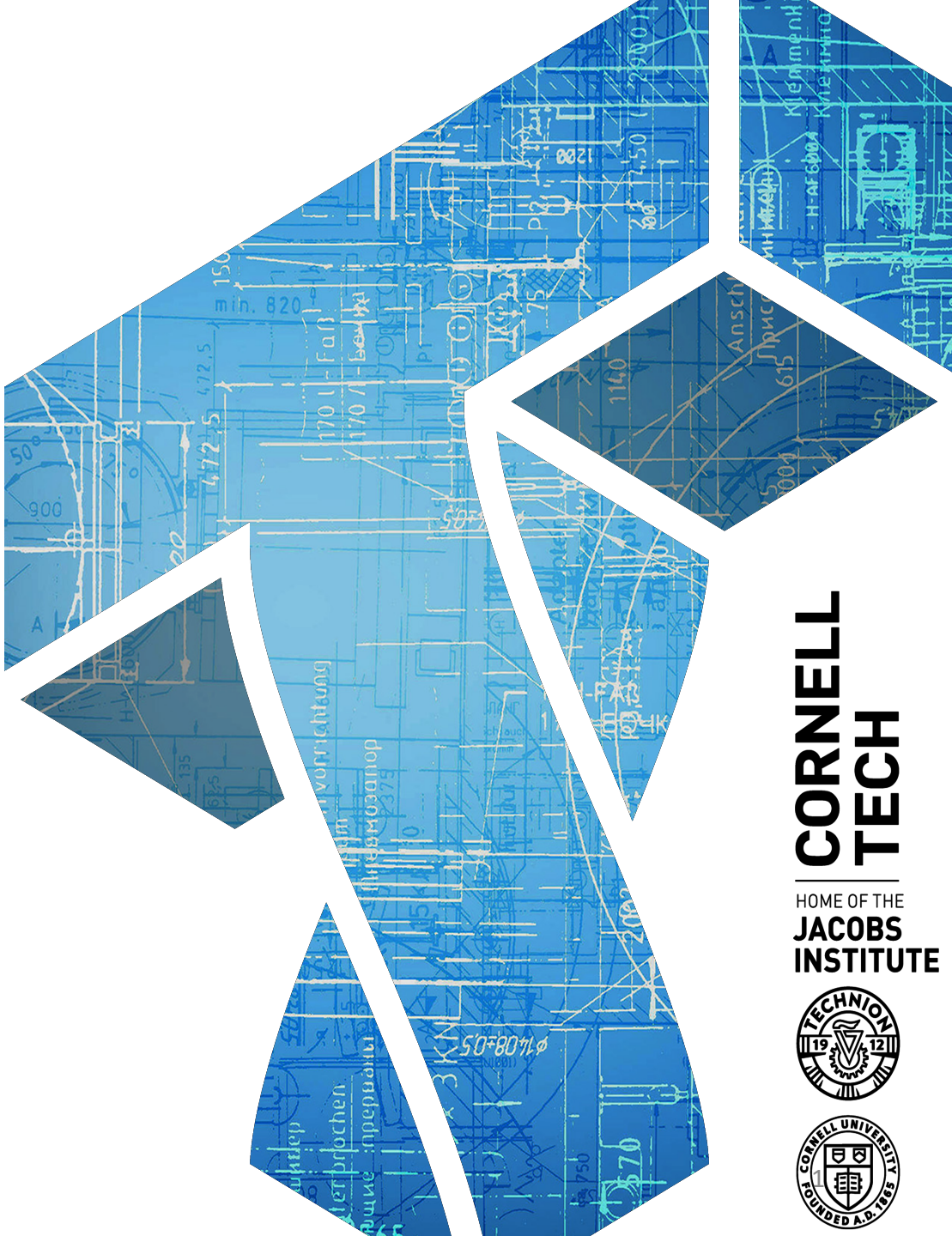
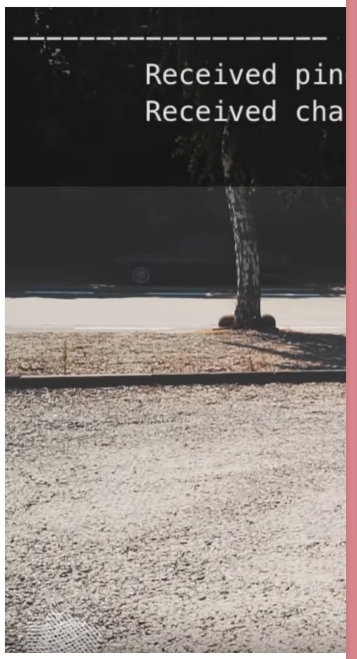
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda

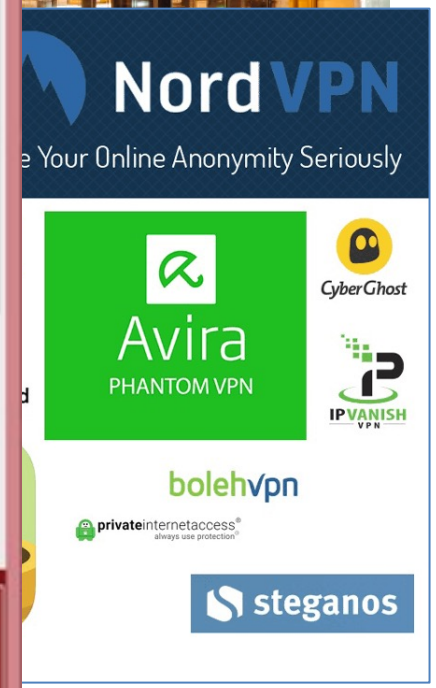


CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Cryptography use cases



https://www



.com/blog/vpn-beginners-guide/

Cryptography in this course

Symmetric primitives

One-time pads

Stream ciphers / Pseudorandom generators
(RC4, ChaCha20, Salsa20)

Block ciphers
(AES, DES, Feistel networks)

Block cipher encryption modes
(CTR, CBC)

Cryptographic hash functions, password hashing, key derivation
(SHA256, SHA3, PBKDF, scrypt, HKDF)

Message authentication
(CBC-MAC, HMAC, Carter-Wegman MACs)

Authenticated encryption with associated data (AEAD)
(Encrypt-then-MAC, AES-GCM)

Asymmetric primitives

Public-key encryption
(Raw RSA, OAEP, ElGamal, Hybrid encryption)

Key exchange
(RSA key transport, Diffie-Hellman over ECC, X3DH)

Digital signatures
(RSA PKCS#1v2, PSS, Schnorr, DSA)

Threat models & security goals

- Algorithms always assumed known by the adversary
  - Kerchoff's principle
- Public keys assumed known by the adversary
- Per-message randomness generally assumed to be private, unless disclosed explicitly (e.g., CTR mode IV) or modeling bad RNG

Threat models & security goals: symmetric encryption

- Confidentiality (of plaintexts & secret keys)
  - Passive adversaries
  - Active adversaries
    - Chosen-plaintext attacks
      - Ability to obtain encryptions of chosen plaintexts, to help decrypt unknown message
    - Chosen-ciphertext attacks
      - Ability to submit mauled ciphertexts to oracle leaking whether decryption succeeds
- Ciphertext integrity (for symmetric encryption):
  - Only holder of symmetric secret key can generate valid ciphertext

Threat models & security goals: Public-key encryption

- Confidentiality (of plaintexts & secret keys)
  - Passive adversaries
  - Active adversaries
    - Chosen-plaintext attacks
      - Ability to obtain encryptions of chosen plaintexts, to help decrypt unknown message
    - Chosen-ciphertext attacks
      - Ability to submit mauled ciphertexts to oracle leaking whether decryption succeeds
- Adversary always knows public key

Threat models have expanded over time

- Originally:
  - Key recovery
  - (Full) plaintext recovery
  - Passive attacks
- Now:
  - Rule out leaking any information about plaintext (semantic security)
  - Chosen-ciphertext attacks assumed

And so have our basic primitives

Symmetric encryption:

Encryption-only modes -> AEAD -> Misuse-resistant AEAD -> Misuse-resistant, committing AEAD

Public-key encryption:

Trapdoor permutations (RSA) -> Randomized PKE -> CCA-secure PKE

Want primitives to provide security in as many contexts as possible.
Improves usability, robustness

Threat models & security goals: message authentication / digital signatures

- Unforgeability (message authentication)
  - Chosen-message attack: can observe tags on adversarially chosen messages, wins by generating new message and valid tag for it
- Unforgeability (digital signatures)
  - Same as above, but adversary also gets public verification key

Threat models & security goals: Cryptographic hash functions

- One-wayness:
  - Can’t recover input given hash, assuming input high entropy
- Collision-resistance:
  - Can’t find two distinct messages that hash to same value
- Random oracle
  - Hash function “behaves like” public, random function

- Slow hashing for passwords (argon2, scrypt)
  - Must use time and memory to compute hash, no “speed up” attacks

Threat models & security goals: Encrypted messaging

Systems like Signal, iMessage

- Sender authenticity
- Message confidentiality and integrity
- Replay resistance
- Forward secrecy

How do cryptographers *evaluate* crypto?

- Cryptanalysis
  - Try to break it. Bonus points if it's an implementable attack
- Formal analyses by hand
  - Give rigorous definition of security, prove manually that scheme meets it (usually via reduction to underlying hard problem)
- Automated protocol analysis tools
  - Build software tools to analyze protocols for bugs
- Implementation analysis tools
  - Build software tools to analyze implementations

How do cryptographers *design* crypto?

- Avoid security through obscurity:
  - Public designs and evaluation
- Public competitions
  - Set out requirements document, solicit submissions and then have several years of people trying to break stuff
- Standardization processes (IETF / ISO)
  - Write down protocols as RFCs (Request for Comments)
  - Sometimes this is design-by-committee
- No single person can design good crypto. Community effort

What should to do if you need some crypto?

- Use existing, well-reviewed libraries
- If not a standard use case: Ask for help!
  - May get pointed to existing solution(s)
  - Even experts don't deploy crypto without others reviewing
  - Experts often happy to help answer questions
- Beware of Stack overflow; vet answers with experts

Primitive	Use cases / examples	Security goals	Good schemes	Bad schemes
Block ciphers	Building block for symmetric encryption	Indistinguishable from random permutation	AES, 3DES	DES, Skipjack
Pseudorandom Functions (PRFs) / Message authentication codes (MACs)	Authenticating data with shared secret key, key derivation	Indistinguishable from random function	HMAC w/ good hash function OMAC, CMAC, PMAC	CBC-MAC without prefix-free encoding
Symmetric encryption (AEAD)	Main mechanism for encrypting data; TLS record layer, encrypting data at rest	Message confidentiality and associated data + ciphertext authenticity	Encrypt-then-MAC GCM OCB	Encryption only modes: CTR mode, CBC mode, ECB mode, RC4
Hash functions	Key derivation, PW hashing, digital signatures, HMAC	Behave like a public random function (implies coll resist, one-wayness, etc.)	SHA-256 SHA-3	MD4, MD5, SHA-1
Password-based key derivation	Password hashing, PW-based encryption	No shortcut attacks	PBKDF2, scrypt, bcrypt, argon2	Plain hash function

Primitive	Use cases / examples	Security goals	Good schemes	Bad schemes
RSA PKE	Encrypt symmetric key	No partial info on messages leaked to active attacker	RSA-OAEP w/ 2048 bit moduli	RSA-PKCS#1 v1.5, “raw” RSA < 2048 bit N
ECC PKE	Encrypt symmetric key	No partial info on messages leaked to active attacker		ElGamal by itself
Hybrid encryption	Encrypt data efficiently using recipient public key	No partial information on messages leaked; attacker can’t maul ciphertext	ECIES RSA-OAEP w/ one-time Encrypt-then-MAC scheme	Raw RSA kem, bad sym encryption (e.g., CBC mode)
Digital signatures	Authenticated key exchange, code signing	Unforgeability under chosen message attacks	ECDSA, RSA PSS	RSA-PKCS#1 v1.5
Diffie-Hellman key exchange	Establishing secure channel	Attacker can’t recover derived session key	ECC DH, Finite field DH	<< 256 bit ECC groups, << 2048 bit FF groups

Some topics we did not get to

- Lightweight cryptography
- Lattice-based and other post-quantum cryptography
- Authenticated data structures
  - Merkle trees, append-only logs
- Blockchain
  - Authenticated data structures + consensus
- More advanced security analyses

Moral character of cryptography

- *"Cryptography rearranges power."*
- Phil Rogaway's paper on moral aspects of cryptography
  - https://web.cs.ucdavis.edu/~rogaway/papers/moral-fn.pdf
- Some high level points:
  - Cryptography is inherently political
  - Modern history of field: "Chaumian" research versus Goldwasser-Micali research, cypherpunks, (lack of) academic political activisim
  - Call to action for cryptographers to embrace social responsibility
- Fascinating read, and would highly recommend!

Summary of the summary

- Cryptography is fascinating subject…
- and important:
  - Plays pivotal role in many security contexts
  - At center of power relationships

- Thanks for participating!