Miscellany!

Lecture 20

The Last Lecture!

Public-Key Crypto Maths

- Initially public-key crypto was based on hardness of problems in modular arithmetic and number theory (RSA/factoring, modular discrete log)
- Problems from several other areas, since then
  - Elliptic curve cryptography (mainstream, currently)
  - Code-based crypto
  - Lattice-based crypto
  - Multivariate Polynomial crypto

"Post-Quantum Crypto" candidates

Elliptic Curve Crypto

- Starting 1985 (by Miller, Koblitz)
- Groups where Discrete log (and DDH) is considered much harder than in modular arithmetic, and hence much smaller groups can be used.
- Given a finite field F, one can define a commutative group $G \subseteq F^2$, as points (x,y) which lie on an "elliptic curve," with an appropriately defined group operation
    - Different curves yield different groups
- Today, most popular PKE schemes use Diffie-Hellman over elliptic curves specified by various standards.
    - Pro: Significantly faster!
    - Con: Which elliptic curves are good?

Code-Based Crypto

- Coding theory based, since McEliece crypto system (1978)
  - A random linear code is specified by a matrix G s.t. a message x is encoded into a codeword Gx. Can easily check if c is a codeword, but seems hard to check if c is close in Hamming distance to a codeword.
  - Structured linear codes exist for which error correcting algorithms are known
  - Idea: Masquerade structured codes to look random. Secret key reveals the original structured code
- Not commonly used today, as large key sizes and slow computation

Lattice-Based Crypto

- Lattice: set of (real) vectors obtained by linear combination of basis vectors using only integer coefficients
  - Hard problems related to finding short vectors in the lattice
- Original use of lattices: to break a candidate for PKE (called the "Knapsack cryptosystem") by Merkle and Hellman
- Constructions: NTRU (1996), Ajtai/Ajtai-Dwork (1996/97), ...
- More recent constructions based on Learning With Errors (LWE) over $\mathbb{Z}_q$ which is hard if some lattice problems are
  - $(A, Ax + e)$ is pseudorandom when e is a "short" noise vector

Lattice-Based Crypto: PKE

- NTRU approach: Private key is a "good" basis, and the public key is a "bad basis"
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis
- To encrypt a message, encode it (randomized) as a short "noise vector" u. Output c = v+u for a lattice point v that is chosen using the public basis
  - To decrypt, use the good basis to find v as the closest lattice vector to c, and recover u=c-v
- NTRU Encryption: use lattices with succinct basis
- Conjectured to be CPA secure for appropriate lattices. No security reduction known to simple lattice problems

Lattice-Based Crypto: PKE

- An LWE based approach:
  - Public-key is (A,P) where P=AS+E, for random matrices (of appropriate dimensions) A and S, and a noise matrix E over $\mathbb{Z}_q$
  - To encrypt an n bit message, first map it to a vector $\underline{\mathbf{v}}$ in (a sparse sub-lattice of) $\mathbb{Z}_q^n$; pick a random vector $\underline{\mathbf{a}}$ with small coordinates; ciphertext is $(\underline{\mathbf{u}},\underline{\mathbf{c}})$ where $\underline{\mathbf{u}} = A^T\underline{\mathbf{a}}$ and $\underline{\mathbf{c}} = P^T\underline{\mathbf{a}} + \underline{\mathbf{v}}$
  - Decryption using S: recover message from $\underline{\mathbf{c}} - S^T\underline{\mathbf{u}} = \underline{\mathbf{v}} + E^T\underline{\mathbf{a}}$
    - Allows a small error probability; can be made negligible by first encoding the message using an error correcting code
  - CPA security: By LWE assumption, the public-key is indistinguishable from random; and, encryption under random (A,P) loses essentially all information about the message

Quantum Cryptography

- Quantum information: Using microscopic physical state of "quantum systems" (spin of atoms/sub-atomic particles, polarization of photons etc.) to encode information (and generate randomness)
  - Communicated over special "quantum channels" (optic fibers, free space...)
- Quantum Key-Distribution: Agreeing on a secret key over a public (quantum+classical) channel, **without computational restrictions on the adversary**
  - Needs authenticated communication between the two parties
  - Can use a short key for (information-theoretically secure) MAC to bootstrap the communication

QKD History

- Bennett and Brassard proposed BB84 in 1984
  - Eavesdropping on a quantum channel will change the qubits that the adversary observes
  - Similar ideas by Wiesner in early 1970s
- QKD scheme based on “entanglement” by Ekert in 1990
- Several other schemes by now

Quantum Key Distribution

- Originally restricted definitions/proofs
  - e.g., in BB84 Eve measured/transformed each transmitted qubit separately
  - Didn't consider composability (e.g., key may be used for other tasks later, and attack may not be separately on QKD and subsequent use)
- Universally Composable Security for QKD (2005)
- Originally several idealizations required for security: crucially depends on reliable quantum channels and devices
  - Many idealizations can be removed using quantum error-correction, quantum repeaters, self-testing devices
  - Commercial products available

End-to-End Encryption

- In typical web-based applications (e.g., email services), servers are privy to all the information used by the clients
  - TLS only protects against outside eavesdroppers
- End-to-end encryption/authentication: Don't trust the server
  - e.g., OpenPGP standard for email
    - Users need to distribute their public-keys (key-signing parties, web-of-trust)
  - Many chat applications using the "Signal protocol"
    - e.g., WhatsApp, Google Allo (incognito mode), Signal, Facebook Messenger (secret conversations)

End-to-End Encryption

- Security considerations
  - Forward secrecy: avoid using long term encryption keys
    - Keep updating the key after each message, deleting old ones, so that the current key doesn't reveal past keys ("Ratcheting")
  - Plausible Deniability: No one should be able prove to anyone else that the sender actually sent a message.
    - Messages are signed using MAC keys that are shared, so receiver can forge MACs. Further, MAC key revealed publicly in the next round, so anyone could have forged MACs.
- Several "usability" considerations in chat applications
  - e.g., Key-exchange requires a Receiver → Sender message. OK in a conversation, but a problem for an offline message.
    - In the Signal Protocol, users leave several public-keys (for one-time use) on the server for senders to use.

Anonymity

- Encryption does not mask the fact that a communication occurred
  - In IPsec tunnel mode (e.g., when using VPN), entire IP packets (including headers) are communicated under encryption
  - But routers themselves can observe source/destination of the packets
- Services that facilitate anonymous routing
  - TOR (based on Onion Routing): sender selects a sequence of routers, with each knowing only its two neighbours in the path.
  - If observing the source and the destination, remains possible to correlate incoming and outgoing channels via timing (no buffering for efficiency)

Leakage from Traffic Rate

- Traffic rate can leak a lot of information about the messages
  - Reading key-strokes from SSH via timing attacks
  - Identifying Netflix movies being played, from the sizes of video segments and the rate at which they are communicated (the compressed sizes depend on the movie)
  - Recovering spoken words from encrypted Voice-over-IP services
- IDEAL model for encryption allowed this explicitly!

Side-channel Attacks

- Various physical signals leak information about the state (including keys and internal randomness) of the algorithms
  - Timing: can be exploited even remotely
  - Power-monitoring: if connected to the same electrical circuitry. Reveals how code executed e.g., taking/not taking conditional branches.
  - Acoustic and/or Electromagnetic signals: with an antenna
  - Cold boot/Data remnance attacks...
- More attacks by actively tampering
- Engineering solution: mitigate the side-channels
- Leakage-Resilient crypto: can't predict/prevent <u>all</u> side-channels, so design assuming a low bandwidth unknown side-channel

Summary

- Many crypto concepts in this course
  - High-level primitives: SKE, PKE (perfect/CPA/CCA), MAC, Digital Signatures, ...
  - Low-level primitives: Secret-Sharing, OWF, PRG, PRF, Trapdoor OWF, 2UHF, CRHF, ...
  - Security Models: IND/SIM, Random Oracle model, ...
  - A little bit of the math: DDH, RSA, ...
- Crucial to practical network security
- Major (but lessening) gaps between theory and practice
- Several other components in network/information security: human behaviour (phishing), software engineering (bugs), formal methods (for security policies, high-level protocols), machine-learning (for intrusion detection), securing hardware, ...

That's All Folks!