Eric Altenburg **CS-306** **September 29, 2020**

Lecture 4.1 - Introduction to modern cryptography

1. Precise assumptions
    - describe all relevant problem components
    - adversary/attacker
        - Type of attacks: threat models
        - capabilities
        - limitations
    - computational assumptions
    - computing settings
2. Why precise assumptions are important?
    - basis for proofs of security
    - comparison among possible solutions
    - flexibility (in design & analysis)
3. Provable security
    - Security
        - Subject to certain assumptions, a scheme s proved to be secure according to a specific definition, against a specific adversary
    - Insecurity
        - A scheme is proved to be insecure with respect to a specific definitions
            * Finding a counterattack example
4. Why is provable security important?
    - In CS: formal proofs may not be necessary
        - Typical/Average case happens often
    - In cryptography: formal proofs are essential
        - Worst case will typically happen

Lecture 4.2 - Computational Security

1. OTP is perfect but impractical
    - 2 drawbacks: very large key and key can only be used once
        - Only once because if you take the two messages, then you can get the key
        - unavoidable so OTP is not practical
            * Because they can communicate the key secretly, why not put the message across that channel too?
2. Relax "perfectness"
    - Perfect secrecy requires that
        - leaks absolutely no extra info
        - unlimited computational power
    - Refined model
        - relaxed notion of security called computational security requires that
            * only leaks tiny amount of extra info
            * bounded with computational power
        - very small probability that we will be broken
3. Definition - Asymptotic
    - A scheme is secure if any efficient attacker A succeeds in breaking the scheme with at most negligible probability
4. Example
    - Almost optimal security guarantees
        - if key length n, the number of possible keys is 2^n
        - attacker running for time t succeeds with probability at most $\sim \frac{t}{2^n}$ (brute-force attack)
    - If $n = 60$, security is enough for attackers running a desktop computer
        - 4 GHz ($4 * 10^9$ cycles/sec), checking all 2^{60} keys require about 9 years

- if $n = 80$, a supercomputer would still need ~2 years
- today's recommended security parameter is at least $n = 128$
  - large difference between 2^{80} and 2^{128}
  - if within 1 year of computation attack is successful w/ prob $\frac{1}{2^{60}}$ then it is more likely that Alice and Bob are hit by lightning

5. Security Relaxation
   - no extra info is leaked out but a tiny amount
   - to computationally bounded attackers
   - attackers best strategy is still ineffective (random guess)

Lecture 4.3

1. Alice and Bob and Eve
   - if eve doesn't know whether it's probabilistic or deterministic, she can just send two of the same messages and see if she gets the same ciphertexts back
   - if she does, then she can say with confidence that the encryption being used is deterministic

Lecture 4.4 - Symmetric encryption, revisted: OTP with psuedorandomness

1. Perfect Secrecy & Randomness
   - Replace randomness with pseudorandomness
   - Ciphertext cannot be efficiently distinguished from pseudorandom
2. Stream Cipher vs Block Cipher
   - Stream:
     - Uses short key for long symbol streams into pseudorandom ciphertext
     - based on PRG
   - Block:
     - Uses short key for block of symbols into pseudorandom ciphertext blocks
     - based on PRF
3. Pseudorandom generators (PRGs)
   - Deterministic algorithm G that on input a seed $s \in \{0,1\}^t$, outputs $G(s) \in \{0,1\}^{l(t)}$
   - G is a PRG if:
     - expansion
       * for polynomial l, it holds that for any n, l(n) > n
       * models the process of extracting randomness from a short random string
     - pseudoranomness
       * no efficient statistical test can tell apart G(s) from a truly random string
4. Realizing ideal block ciphers in practice
   - We want a random mapping of n-bit inputs to n-bit outputs
     - There are $\approx 2^{n2^n}$
     - Can't do it in practice
   - Instead, we use a keyed function $F_k : \{0,1\}^n \to \{0,1\}^n$
     - indexed by a t-bit key k
     - there are only 2^t such keyed functions
     - a random key selects a "random-enough" mapping or a pseudorandom function
5. Generic PRF-based symmetric encryption
   - fixed length message encryption