Outline

- Confidentiality and privacy, cont’d
- Even more web risks
- Crypto basics
- Stream ciphers
- Block ciphers and modes of operation

Adjusting client behavior

- HTTPS and password fields are basic hints
- Consider disabling autocomplete
  - Usability tradeoff, save users from themselves
  - Finally standardized in HTML5
- Consider disabling caching
  - Performance tradeoff
  - Better not to have this on user’s disk
  - Or proxy? You need SSL

User vs. site perspective

- User privacy goals can be opposed to site goals
- Such as in tracking for advertisements
- Browser makers can find themselves in the middle
  - Of course, differ in institutional pressures

Third party content / web bugs

- Much tracking involves sites other than the one in the URL bar
  - For fun, check where your cookies are coming from
- Various levels of cooperation
- Web bugs are typically 1x1 images used only for tracking

Cookies arms race

- Privacy-sensitive users like to block and/or delete cookies
- Sites have various reasons to retain identification
- Various workarounds:
  - Similar features in Flash and HTML5
  - Various channels related to the cache
  - Evercookie: store in n places, regenerate if subset are deleted

Browser fingerprinting

- Combine various server or JS-visible attributes passively
  - User agent string (10 bits)
  - Window/screen size (4.83 bits)
  - Available fonts (13.9 bits)
  - Plugin versions (15.4 bits)
- (Data from panopticlick.eff.org, far from exhaustive)

History stealing

- History of what sites you’ve visited is not supposed to be JS-visible
- But, many side-channel attacks have been possible
  - Query link color
  - CSS style with external image for visited links
  - Slow-rendering timing channel
  - Harvesting bitmaps
  - User perception (e.g. fake CAPTCHA)
Browser and extension choices

- More aggressive privacy behavior lives in extensions
  - Disabling most JavaScript (NoScript)
  - HTTPS Everywhere (centralized list)
  - Tor Browser Bundle
- Default behavior is much more controversial
  - Concern not to kill advertising support as an economic model

Openness tradeoffs

- Error reporting
  - Few benign users want to see a stack backtrace
- Directory listings
  - Hallmark of the old days
- Readable source code of scripts
  - Doesn’t have your DB password in it, does it?

Clickjacking

- Fool users about what they’re clicking on
  - Circumvent security confirmations
  - Fabricate ad interest
- Example techniques:
  - Frame embedding
  - Transparency
  - Spoof cursor
  - Temporal “bait and switch”

Crawling and scraping

- A lot of web content is free-of-charge, but proprietary
  - Yours in a certain context, if you view ads, etc.
- Sites don’t want it downloaded automatically (web crawling)
- Or parsed and user for another purpose (screen scraping)
- High-rate or honest access detectable
-ography, -ology, -analysis

Cryptography (narrow sense): designing encryption
Cryptanalysis: breaking encryption
Cryptology: both of the above
Code (narrow sense): word-for-concept substitution
Cipher: the "codes" we actually care about

Caesar cipher
- Advance three letters in alphabet: A → D, B → E, ...
- Decrypt by going back three letters
- Internet-era variant: rot-13
- Easy to break if you know the principle

Keys and Kerckhoffs's principle
- The only secret part of the cipher is a key
- Security does not depend on anything else being secret
- Modern (esp. civilian, academic) crypto embraces openness quite strongly

Symmetric vs. public key
- Symmetric key (today's lecture): one key used by all participants
- Public key: one key kept secret, another published
  - Techniques invented in 1970s
  - Makes key distribution easier
  - Depends on fancier math

Goal: secure channel
- Leaks no content information
  - Not protected: size, timing
- Messages delivered intact and in order
  - Or not at all
- Even if an adversary can read, insert, and delete traffic

One-time pad
- Secret key is truly random data as long as message
- Encrypt by XOR (more generally addition mod alphabet size)
- Provides perfect, "information-theoretic" secrecy
- No way to get around key size requirement

Computational security
- More realistic: assume adversary has a limit on computing power
- Secure if breaking encryption is computationally infeasible
  - E.g., exponential-time brute-force search
- Ties cryptography to complexity theory

Key sizes and security levels
- Difficulty measured in powers of two, ignore small constant factors
- Power of attack measured by number of steps, aim for better than brute force
  - $2^{32}$ definitely too easy, probably $2^{64}$ too
- Modern symmetric key size: at least $2^{128}$
Crypto primitives

- Base complicated systems on a minimal number of simple operations
- Designed to be fast, secure in wide variety of uses
- Study those primitives very intensely

Attacks on encryption

- Known ciphertext
  - Weakest attack
- Known plaintext (and corresponding ciphertext)
- Chosen plaintext
- Chosen ciphertext (and plaintext)
  - Strongest version: adaptive

Certificational attacks

- Good primitive claims no attack more effective than brute force
- Any break is news, even if it’s not yet practical
  - Canary in the coal mine
- E.g., $2^{126.1}$ attack against AES-128
- Also watched: attacks against simplified variants

Fundamental ignorance

- We don’t really know that any computational cryptosystem is secure
- Security proof would be tantamount to proving $P \neq \text{NP}$
- Crypto is fundamentally more uncertain than other parts of security

Relative proofs

- Prove security under an unproved assumption
- In symmetric crypto, prove a construction is secure if the primitive is
  - Often the proof looks like: if the construction is insecure, so is the primitive
- Can also prove immunity against a particular kind of attack

Random oracle paradigm

- Assume ideal model of primitives: functions selected uniformly from a large space
  - Anderson: elves in boxes
- Not theoretically sound; assumption cannot be satisfied
- But seems to be safe in practice

Pseudorandomness and distinguishers

- Claim: primitive cannot be distinguished from a truly random counterpart
  - In polynomial time with non-negligible probability
- We can build a distinguisher algorithm to exploit any weakness
- Slightly too strong for most practical primitives, but a good goal

Open standards

- How can we get good primitives?
- Open-world best practice: run competition, invite experts to propose then attack
- Run by neutral experts, e.g. US NIST
- Recent good examples: AES, SHA-3
A certain three-letter agency

National Security Agency (NSA): has primary responsibility for “signals intelligence”

Dual-mission tension:
- Break the encryption of everyone in the world
- Help US encryption not be broken by foreign powers

Outline

Confidentiality and privacy, cont’d
Even more web risks
Crypto basics
Stream ciphers
Block ciphers and modes of operation

Stream ciphers

Closest computational version of one-time pad
Key (or seed) used to generate a long pseudorandom bitstream
Closely related: cryptographic RNG

Shift register stream ciphers

Linear-feedback shift register (LFSR): easy way to generate long pseudorandom sequence
- But linearity allows for attack
- Several ways to add non-linearity
- Common in constrained hardware, poor security record

RC4

Fast, simple, widely used software stream cipher
- Previously a trade secret, also “ARCFOUR”
- Many attacks, none yet fatal to careful users (e.g. TLS)
- Famous non-careful user: WEP
- Now deprecated, not recommended for new uses

Encryption ≠ integrity

Encryption protects secrecy, not message integrity
- For constant-size encryption, changing the ciphertext just creates a different plaintext
- How will your system handle that?
- Always need to take care of integrity separately

Stream cipher mutability

Strong example of encryption vs. integrity
- In stream cipher, flipping a ciphertext bit flips the corresponding plaintext bit, only
- Very convenient for targeted changes

Salsa and ChaCha

Published by Daniel Bernstein 2007-2008
- Stream cipher with random access to stream
  - Related to counter mode discussed later
- Fast on general-purpose CPUs without specialized hardware
- Adopted as option for TLS and SSH
  - Prominent early adopter: Chrome on Android
Stream cipher assessment

- Currently less fashionable as a primitive in software
- Not inherently insecure
- Other common pitfall: must not reuse key(stream)

Outline

Confidentiality and privacy, cont’d
Even more web risks
Crypto basics
Stream ciphers
Block ciphers and modes of operation

Basic idea

- Encryption/decryption for a fixed sized block
- Insecure if block size is too small
  - Barely enough: 64 bits; current standard: 128
- Reversible, so must be one-to-one and onto function

Pseudorandom permutation

- Ideal model: key selects a random invertible function
- I.e., permutation (PRP) on block space
- Note: not permutation on bits
- "Strong" PRP: distinguisher can decrypt as well as encrypt

Confusion and diffusion

- Basic design principles articulated by Shannon
- Confusion: combine elements so none can be analyzed individually
- Diffusion: spread the effect of one symbol around to others
- Iterate multiple rounds of transformation

Substitution/permutation network

- Parallel structure combining reversible elements:
- Substitution: invertible lookup table ("S-box")
- Permutation: shuffle bits

AES

- Advanced Encryption Standard: NIST contest 2001
  - Developed under the name Rijndael
- 128-bit block, 128/192/256-bit key
- Fast software implementation with lookup tables (or dedicated insns)
- Allowed by US government up to Top Secret

Feistel cipher

- Split block in half, operate in turn:
  \((L_{i+1}, R_{i+1}) = (R_i, L_i \oplus F(R_i, K_i))\)
- Key advantage: F need not be invertible
  - Also saves space in hardware
- Luby-Rackoff: if F is pseudo-random, 4 or more rounds gives a strong PRP
**DES**
- Data Encryption Standard: AES predecessor 1977-2005
- 64-bit block, 56-bit key
- Implementable in 70s hardware, not terribly fast in software
- Triple DES variant still used in places

**Some DES history**
- Developed primarily at IBM, based on an earlier cipher named "Lucifer"
- Final spec helped and "helped" by the NSA
  - Argued for smaller key size
  - S-boxes tweaked to avoid a then-secret attack
- Eventually victim to brute-force attack

**DES brute force history**
- 1977 est. $20m cost custom hardware
- 1993 est. $1m cost custom hardware
- 1997 distributed software break
- 1998 $250k built ASIC hardware
- 2006 $10k FPGAs
- 2012 as-a-service against MS-CHAPv2

**Double encryption?**
- Combine two different block ciphers?
  - Belt and suspenders
  - Anderson: don't do it
  - FS&K: could do it, not a recommendation
  - Maurer and Massey (J.Crypt'93): might only be as strong as first cipher

**Modes of operation**
- How to build a cipher for arbitrary-length data from a block cipher
- Many approaches considered
  - For some reason, most have three-letter acronyms
  - More recently: properties susceptible to relative proof

**ECB**
- Electronic CodeBook
  - Split into blocks, apply cipher to each one individually
  -Leaks equalities between plaintext blocks
  - Almost never suitable for general use

**Do not use ECB**

**CBC**
- Cipher Block Chaining
  - $C_i = E_K(P_i \oplus C_{i-1})$
  - Long-time most popular approach, starting to decline
  - Plaintext changes propagate forever, ciphertext changes only one block
**CBC: getting an IV**

- $C_0$ is called the initialization vector (IV)
  - Must be known for decryption
- IV should be random-looking
  - To prevent first-block equalities from leaking (lesser version of ECB problem)
- Common approaches
  - Generate at random
  - Encrypt a nonce

**Stream modes: OFB, CTR**

- Output FeedBack: produce keystream by repeatedly encrypting the IV
  - Danger: collisions lead to repeated keystream
- Counter: produce from encryptions of an incrementing value
  - Recently becoming more popular: allows parallelization and random access