Cryptographic Hashes

Peter Chapin

Vermont State University

CIS-3240: Computer Security

What is a File?

- For our purposes, a file is just a “bag of bytes.”
  - A collection of values from 0 to 255 (or 0x00 to 0xFF in hex).
  - Each value is 8 bits
  - *No interpretation!*
    - We are not concerned with the meaning of the file’s contents
- A file can be small
  - … even just one byte
- A file can be large
  - … many gigabytes or more!
- A file contains an integer number of bytes; there are no partial bytes.

Simple Checksum

- Add all the bytes of a file, ignoring carry
  - File: 0x00 0x05 0x2C 0x3A 0xFF (just five bytes)
  - Checksum: 0x6A
- Send the file and the checksum
  - The receiver computes the checksum and compares it with the value received
  - If they disagree, there was a transmission error
- Provides error detection

Error Detection Incomplete

- Consider this case
  - Original file: 0x00 0x05 0x2C 0x3A 0xF
  - Modified file: 0x01 0x04 0x2C 0x3A 0xF
  - Both have the same checksum
- The receiver won't notice that the file has been modified!
- However...
  - Errors due to "natural causes" (noise, truncated files, etc) will *usually* be detected.

Error Detection Always Incomplete

- No error detection scheme can detect all errors!
  - 8-bit checksum => only 256 possible checksum values
  - 1000-byte message => 2^{8000} messages!
  - Thus, many (*many*!) messages must have the same checksum
  - *Note that I'm using "file" and "message" interchangeably.*

CRC Checksums

- **Cyclic Redundancy Check** checksums…
  - … are typically larger (16-bit or 32-bit)
  - … are good against naturally occurring errors (burst noise)
  - … are easy to calculate in hardware
- *They are commonly used*
  - NICs compute CRCs on incoming/outgoing network packets
  - Disk controllers compute CRCs on disk blocks read/written
  - *This is how our machines know they've read or received bad data!*

Enter: Mallory

- Unfortunately, CRCs are not good against a malicious attacker
  - Mallory can “easily” modify a message such that it has the same CRC.
  - Message: “I will pay Mallory $100.00”
  - Modified Message: “I will pay Mallory $1000.00”
  - Mallory then adjusts the message in some trivial way so the CRC is the same.
    - e.g., adds extra spaces, blank lines, rewords sentences, etc.
- *Violation of Data Integrity!*
  - The second of the “Big Two” security services after confidentiality

Cryptographic Hash Functions

- A *cryptographic hash function* (or just “hash function”)…
  - … is like a super checksum
  - Mallory can’t feasibly modify the message so that it produces the same *hash value*.
- Let
  - *M* be the message; *F* be the hash function; *H* be the hash value
  - Then we can write: *H = F(M)*
- Typical hash values are large (160 bits or more)
  - … but still small compared to the message being hashed (usually)

Hash Function Behavior

- Pre-Image Resistance
  - Given *H*, it should be infeasible to find an *M* such that *H = F(M)*
- Collision Resistance
  - Let $H = F(M_1)$ for a given M_1, it should be infeasible to find a second message M_2 such that $H = F(M_2)$
  - Note that many such messages exist, but it should be difficult to find one
  - Mallory should be unable to modify M_1 to some M_2 with the same H
- Strong Collision Resistance
  - It should be infeasible to find two messages M_1 and M_2 such that $H = F(M_1)$ and $H = F(M_2)$. Note that neither message is given here.

Happy Birthday!

- All serious hash functions have strong collision resistance because of the Birthday Attack
  - Go around the room and have everyone announce their birthday
  - About how many people do you need before two people have the same birthday?
  - *Much fewer than you might think!*
  - Each birthday announcement adds another target for new people to match
  - When the number of people involved is 23 (or more), the probability is greater than 50% that two will have the same birthday.
  - Sometimes called the *Birthday Surprise*

Alice Cheats Bob

- Alice prepares two contracts:
  - C1: “Alice will pay Bob a fortune for no significant work.”
    C2: “Alice will pay Bob nothing, and he will become her slave.”
  - Alice identifies 32 (for example) independent, inconsequential changes to each contract
    - e.g., extra spaces, blank lines, adjusted wordings
  - Alice computes *H = F(C1)* for every possible combination of changes and makes a table of results. The table has 2^{32} entries
  - Alice computes *H = F(C2)* for every possible combination of changes and compares each result with the previous table. Finds match!

Alice Cheats Bob

- After 2^{33} hash computations, Alice has now found two messages, *C1* and *C2*, with the same hash!
- Alice does this:
  - Alice presents *C1* to Bob, which he signs (we will talk about digital signatures later).
  - Alice cuts his signature from *C1* and appends it to *C2*
  - The signature verifies because *C2*'s hash is the same as *C1*'s!
  - Bob is now Alice's slave
- Surprisingly, this method works fine for 64-bit hashes!
  - … because of the Birthday Surprise

64 Bit Hashes Too Small

- Any reasonable hash function generates 128-bit hashes or more
- Today, most people use 160-bit or 256-bit hashes (or even larger)
- Consider what a 160-bit hash means for Alice
  - She has to do 2^{80} hash calculations to make a table with 2^{80} hash values, each 20 bytes in size (so 20 * 2^{80} bytes)
  - She has to do another 2^{80} hash calculations and look up each hash value in the previously made table
  - *Very difficult!*
- Note: Birthday Attack is independent of hash function
  - Amounts to “brute force” for the hash function world

Caution!

- Although one can't (feasibly) change a message to have the same hash…
  - … if the attacker is in a position to change the message, they can **also** change the hash to match.
  - … Thus, *just computing a hash by itself doesn't ensure data integrity*. It must be coupled with something else (digital signature, MAC, etc.). We will get to those things.