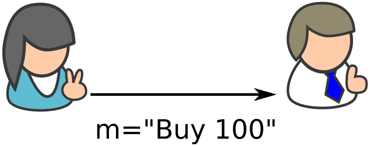
50.020 Security
Lecture 3: Hash Functions

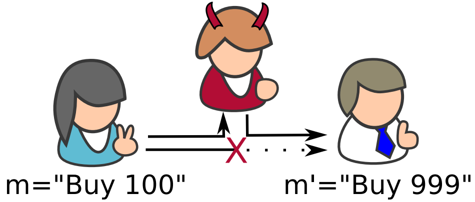
Data Integrity

Data Manipulation attacks



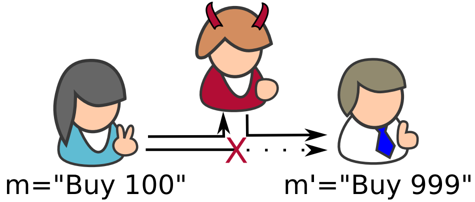
- Alice sends Bob a message:
  - "Hi Bob, I'm Alice, please buy 100 stocks of Company A"
- Alice sends the message in plaintext
- Attacker Eve wants to manipulate Alice's stock trade.
  - Eve can jam, eavesdrop and insert
- What kind of attacks are possible here?

Data Manipulation attacks



- Attack example: Attacker eavesdrops, jams, spoofs similar message:
  - "Hi Bob, I'm Alice, please buy 999 stocks of Company B"
- Bob assumes the message is from Alice, buys stocks for her
- What is the problem here?

Data Manipulation attacks



- Attack example: Attacker eavesdrops, jams, spoofs similar message:
  - "Hi Bob, I'm Alice, please buy 999 stocks of Company B"
- Bob assumes the message is from Alice, buys stocks for her
- What is the problem here?

- Secure authentication and integrity of the message

How to protect the message?

- Obvious idea: encrypt the message (e.g., using OTP)

Example (Using OTP to encrypt "buy100")

```
- "buy100"= 0x62757931300a
- Key      = 0xA29C7B1E0E3AEE
- Result   = 0xC0E9022F3E0AE4
```

How to protect the message?

- Obvious idea: encrypt the message (e.g., using OTP)

Example (Using OTP to encrypt "buy100")

```
- "buy100"= 0x6275793130300a
- Key      = 0xA29C7B1E0E3AEE
- Result   = 0xC0E9022F3E0AE4
```

- Can an eavesdropper break the confidentiality of the message?
- Can an eavesdropping and injecting attacker change the content?

Does symmetric encryption protect data integrity?

Your answer (Yes/No)? Why?
Solution to be provided in class.

Other measures to protect integrity

- Block ciphers are not always enough
- We need a dedicated tool to validate message integrity

Cryptographic Hash Functions

Cryptographic properties for functions

- In cryptography, *preimage* resistance means that given $y = f(x)$
  - it is *hard* to find the input x for f to produce y
- *Second pre-image* resistance means that given x and f
  - it is *hard* to find an input x' for f such that $f(x) = f(x')$
- *Collision* resistance means that given f
  - it is *hard* to find any two inputs x, x' for f such that $f(x) = f(x')$
- *Random oracle* property: A random oracle maps each unique input to random output with uniform distribution
  - Informally: for two correlated inputs m_1 and m_2, the output of f is completely uncorrelated
- CRCs have only preimage resistance

Design goals for hash functions

- Cryptographic hash functions are designed to have all four properties
  - Preimage resistance
  - Second preimage resistance
  - Collision resistance
  - Random oracle property
- Using cryptographic hash functions, *message authentication codes* can be constructed
- We now discuss special algorithms, similar goals can be achieved with block ciphers
- Standard hash functions are not designed to have all of these properties

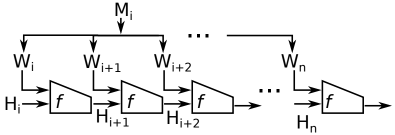
SHA-1

We will explain hash functions based on SHA-1. It has the following characteristics:

- Processes input blocks of 512 bit
- Pre-defined initial state of 160 bit
- Hash output is a 160 bit block
- Uses Merkle-Dåmgard construction
- 80 internal rounds in total

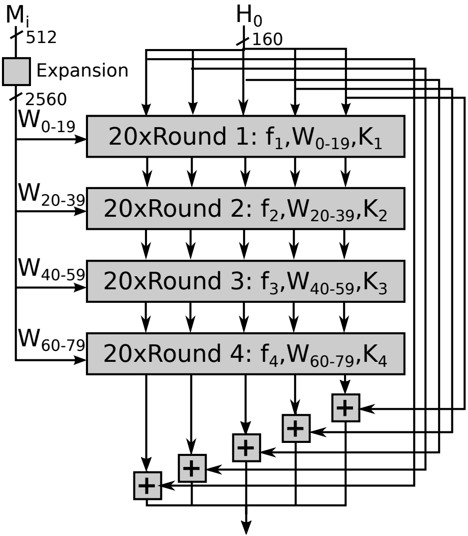
Merkle-Dåmgard construction

50.020 Security Lecture 3: Hash Functions

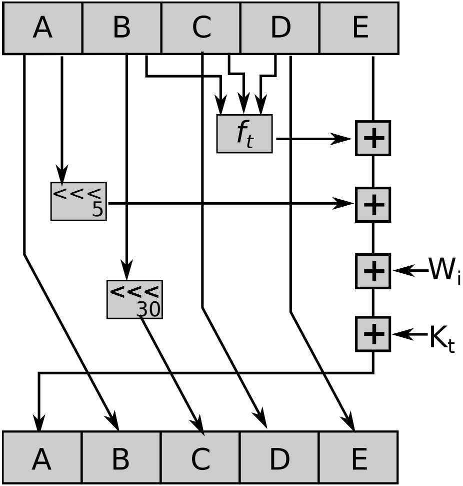


- Merkle-Dåmgard is a construction for cryptographic hashes:
  - Repeated application of a collision resistant compressing function
  - Each stage uses previous output and new chunk of input
- In SHA-1
  - SHA-1 has a constant (public) initial values in the MD chain
  - 512 bit input blocks are expanded into 2560 bit=80$\cdot$ 32 bit words
  - 4 stages, each stage has 20 rounds of compression
  - Each stage has different constants K_t and a non-linear function f_t

Overall SHA-1 operation



One round in SHA-1



Why 80 rounds?

- Increasing the number of rounds has several benefits:
  - It makes brute force attacks more expensive (each hashing takes longer)
  - It makes attacks relying on *differential cryptanalysis* harder
- The exact value for SHA-1 was most likely chosen as compromise between effort and security
- For SHA-2, 64 rounds are default. Attacks have been found for 52 round versions

Cryptanalysis of hash functions

- Two potential goals for attacker: find preimages or collisions
- Collisions are much easier to find, but less useful
- It has been shown that for MD, if f is collision resistant, then H is collision resistant
- Attacking the collision resistance of f is a first part of attack
  - Find two plaintexts that hash to the same value
  - What is the estimated effort for an n bit hash? 2^n?
- Actually, it is only $2^{\mathrm{n}/2}$. Why?

Birthday paradox:

- What is the probability, that in a group of n people, two have the same birthday?
- Variant: for which group size, the probability approaches 0.5?

- for 23 people, the probability is 50%
- for 70 people, the probability is 99.9%

- For SHA1, a minimum hash length of 160 bits is usually suggested
- A 160 bit has relates to 2^{80} effort to find collision (considered infeasible today)

Birthday paradox math

- What is the probability to have NO people with shared birthday?
  - Lets denote d=365, and number of people=n
  - Per pair, 1/d chance to have same birthday
  - For n people, $\binom{n}{2}$ distinct pairings

People (n)	# pairs	P(No shared birthday)	Numerical result
2	1	1-1/d	0.99
3	3	$(1\text{-}1/d)^{3}$	0.99
4	6	$(1\text{-}1/d)^{6}$	0.98
5	10	$(1\text{-}1/d)^{10}$	0.97
23	253	$(1\text{-}1/d)^{253}$	0.49

- Or as closed form expression: P(No shared birthday)=$(1-\frac{1}{d})^{\binom{n}{2}}$

How to use this for an attack

- Collisions can be directly be used to attack
  - Commitment schemes
  - Digital signature schemes
  - TLS certificates (more on them later, breaks TLS)
- Anything where the plaintext is under direct control of attacker
- Attacks have been demonstrated for MD5 (precursor of SHA-1) and SHA-1
  - Keywords: "MD5 Collisions Inc" and SHAttered
- Birthday paradoxon does not help for second preimage finding
  - Our message authentication system can use SHA-1 safely

Cryptanalysis of SHA-1

- In Feb 2005, researchers found the following:
  - Collisions can be found with effort 2^{69} steps (instead of 2^{80}, factor 2048)
  - In 2009, that result was claimed to be improved to 2^{52} steps (but found to be incorrect)
  - If assuming 2^{60} tries required, and 2^{14} ops per SHA-1 [1]
  - Nowadays, breaking SHA-1 would probably cost
    - In 2015, $700k
    - In 2018, $173k
    - In 2021, $43k...
- Google computed first collision in 2017[2], claim took 9,223,372,036,854,775,808$\approx 2^{63}$ tries
  - 6,500 years of single-CPU computations and 110 years of single-GPU computations.
  - Assuming 100$ per year per CPU, cost=650,000$

[1] schneier.com/blog/archives/2012/10/when_will_we_se.html

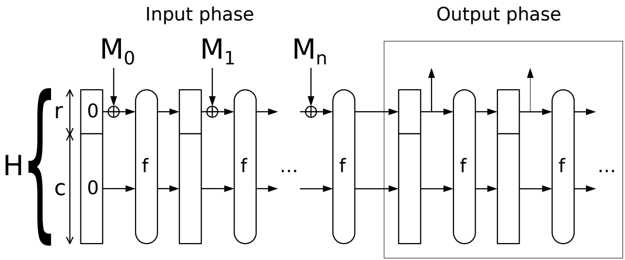
[2] https://shattered.io/

SHA-1, SHA-2, SHA-3

- SHA-2 was designed by NSA (like SHA-1), and published in 2001
- US National Institute of Standards and Technology (NIST) "promotes" security standards
- Successor of SHA-2 was chosen in a semi-public process
- In Oct 2012, Keccak was selected as SHA-3 algorithm
  - Focus on security and implementation speed
- SHA-1 appears to have weaknesses as discussed
  - SHA-2 shares a lot of the structure
- SHA-3 should be considered for high-security projects

SHA-3

- SHA-3 (Keccak) is fundamentally different to SHA-1/SHA-2
- It uses a "sponge" construction instead of MD
- r bits of message are "fed" into S per round
- r bits of output per round can be taken out afterwards



Conclusion

- Message integrity is not preserved by stream ciphers
- Many other ciphers also do not guarantee integrity
- Secure Hash functions are designed to allow integrity validation
  - In particular, second preimage resistance helps here
- MD5 is practically broken
- SHA-1's collision resistance is questionable
- Long term, SHA-3 is best choice