Remote questions: https://course.netsec.inf.ethz.ch/questions

Crypto Refresher

Symmetric Cryptography, Asymmetric Cryptography, Hash Functions

Network Security AS 2020

15 September 2020

Adrian Perrig

ETH zürich

Other Lectures about Cryptography

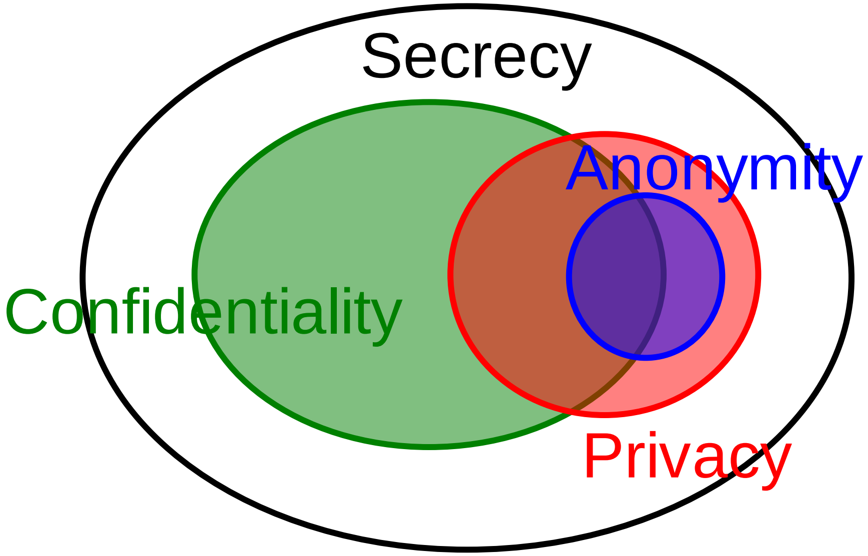
- Information Security (Prof. Dr. Srdjan Capkun, Prof. Dr. David Basin, Dr. Ralf Sasse)
  https://infsec.ethz.ch/education/as15_ss19/ss2020/infsec.html
- Applied Cryptography (Prof. Dr. Kenny Paterson)
  https://appliedcrypto.ethz.ch/education/lectures/appcry1.html

Secrecy, Confidentiality, Privacy, Anonymity

- Often considered synonymous, but are slightly different
- Secrecy
  - Keep data hidden from unintended receivers
  - “Alice and Bob use encrypted communication links to achieve secrecy”
- Confidentiality
  - Keep *someone else’s* data secret
  - “Trent encrypts all user information to keep their client’s information confidential in case of a file server compromise”
- Privacy
  - Keep *data about a person* secret
  - “To protect Alice’s privacy, company XYZ did not disclose any personal information”

Secrecy, Confidentiality, Privacy, Anonymity

- Anonymity
  - Keep *identity* of a protocol participant secret
  - “To hide her identity to the web server, Alice uses The Onion Router (TOR) to communicate”



Integrity, Authentication

- Sometimes used interchangeably, but they have different connotations
- Data integrity
  - Ensure data is “correct” (i.e., correct syntax & unchanged)
  - Prevents unauthorized or improper changes
  - “Trent always verifies the integrity of his database after restoring a backup, to ensure that no incorrect records exist”
- Entity authentication or identification
  - Verify the identity of another protocol participant
  - “Alice authenticates Bob each time they establish a secure connection”
- Data authentication
  - Ensure that data originates from claimed sender
  - “For every message Bob sends, Alice authenticates it to ensure that it originates from Bob”

Difference between Integrity and Authentication

- Integrity is often a property of *local or stored data*
  - For example, we want to ensure integrity for a database stored on disk, which emphasizes that we want to prevent unauthorized changes
  - Integrity emphasizes that data has not been changed
- Authentication used in network context, where entities communicate across a network
  - Two communicating hosts want to achieve data authentication to ensure data was not changed by network
  - Authentication emphasizes that data *was created by a specific sender*
  - Implies integrity, data unchanged in transit
  - Implies that identity of sender is verified

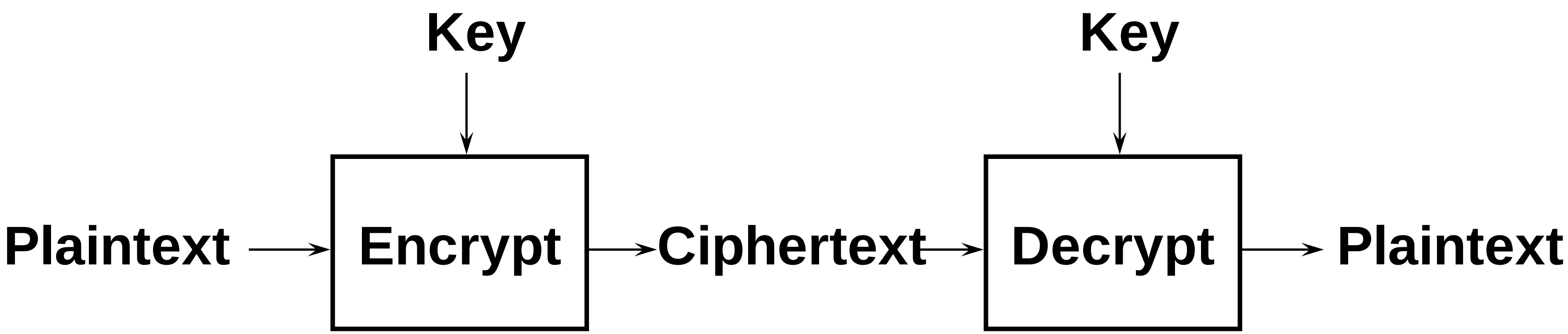
Basic Cryptographic Primitives

- Symmetric (shared-key, same-key)
  - Block cipher (pseudo-random permutation PRP)
  - Stream cipher (pseudo-random generators PRG)
  - Message authentication code (MAC)
- Asymmetric (public-private key)
  - Diffie–Hellman key agreement
  - Public-key encryption
  - Digital signature
- Others (unkeyed symmetric)
  - One-way function
  - Cryptographic hash function

Symmetric Cryptography

Symmetric Encryption Primitives

- Encryption key = decryption key
- Encryption: E_K(plaintext) = ciphertext
- Decryption: D_K(ciphertext) = plaintext
- We write {plaintext}$_K$ for E_K(plaintext)



Stream Ciphers

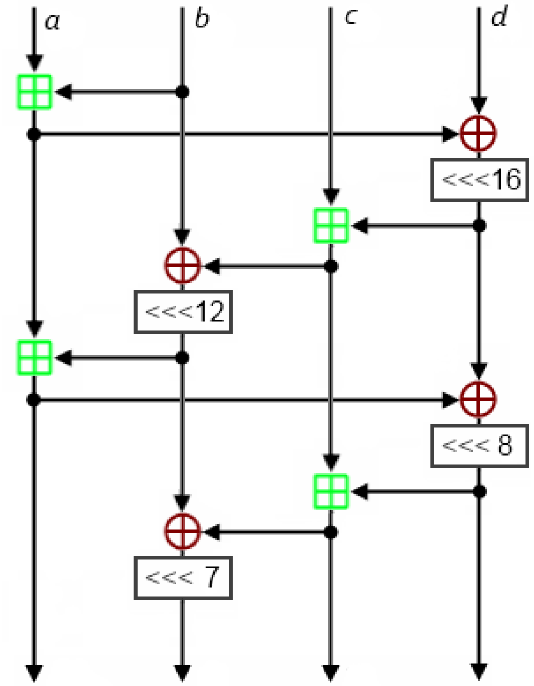
- One-time pad
  - Use unique random keystream for each message
  - Is this secure? **Yes.**
  - Is this secure if we re-use keystream? **No.**
- Stream ciphers use pseudo-random generator (PRG) to generate keystream from seed
  - Encryption: use shared key k and initialization vector IV for the seed
    ciphertext = plaintext $\oplus$ PRG(k, IV)
  - Send IV, ciphertext
- Example: AES in CTR mode

ChaCha Stream Cipher

- Dan Bernstein designed ChaCha in 2008 (adaptation of earlier Salsa stream cipher)
- Used in TLS 1.3 and Wireguard VPN
- Structure: 128-bit constant, a 256-bit key, a 64-bit counter (Position), and a 64-bit nonce, arranged as a 4 × 4 matrix of 32-bit words
- Quarter round function illustrates cipher structure using addition, xor, and rotate
- High speed: around 4 cycles per byte

Initial state of ChaCha

Cons	Cons	Cons	Cons
Key	Key	Key	Key
Key	Key	Key	Key
Pos	Pos	Nonce	Nonce



Stream Cipher Vulnerabilities

- Keystream reuse attack
  - Enormous security vulnerability if same keystream used to encrypt two different messages
  - $c1 = p1 \oplus k$, $c2 = p2 \oplus k$
  - $c1 \oplus c2 = p1 \oplus p2$ (which is easy to analyze, because the unknown key is removed!)
  - $c1 = p1 \oplus \mathrm{PRG}(K, IV)$, where IV = initialization vector, make sure IV is never used twice!
- Ciphertext modification attack
  - Alteration of ciphertext will alter corresponding values in plaintext after decryption
  - Example, encrypt a single bit: $c = p \oplus k$, for $p=1$, $k=0$, thus $c=1$
  - If attacker changes c to 0 during transmission, decrypted value is changed to 0! $p = c \oplus k$, if $c=0$, $k=0$, then $p=0$
  - To defend, need to ensure authenticity of ciphertext

Block Ciphers

- Block cipher is a pseudo-random permutation (PRP), each key defines a one-to-one mapping of input block to output block
  - Substitution cipher with large block size
- Encrypt each block separately
- Examples: DES, Rijndael (=AES)

Advanced Encryption Standard: AES

- Officially adopted for US government work, but voluntarily adopted by private sector
- Winning cipher was Rijndael (pronounced Rhine-doll)
  - Belgian designers: Joan Daemen & Vincent Rijmen
- Adopted by NIST in November 2001
- {128, 192, 256}-bit key size
- High-speed cipher
  - Using native AESni instructions on Intel and AMD CPUs, 128-bit AES encryption requires only 30 clock cycles! 7 times faster than loading a byte from DRAM!

Block-Cipher Modes of Operation

- Block cipher modes of operation
  - ECB: Electronic code book
  - CBC: Cipher block chaining
  - CFB: Cipher feedback
  - OFB: Output feedback
  - CTR: Counter mode
  - GCM: Galois Counter Mode:
    encryption and authentication in a single pass!

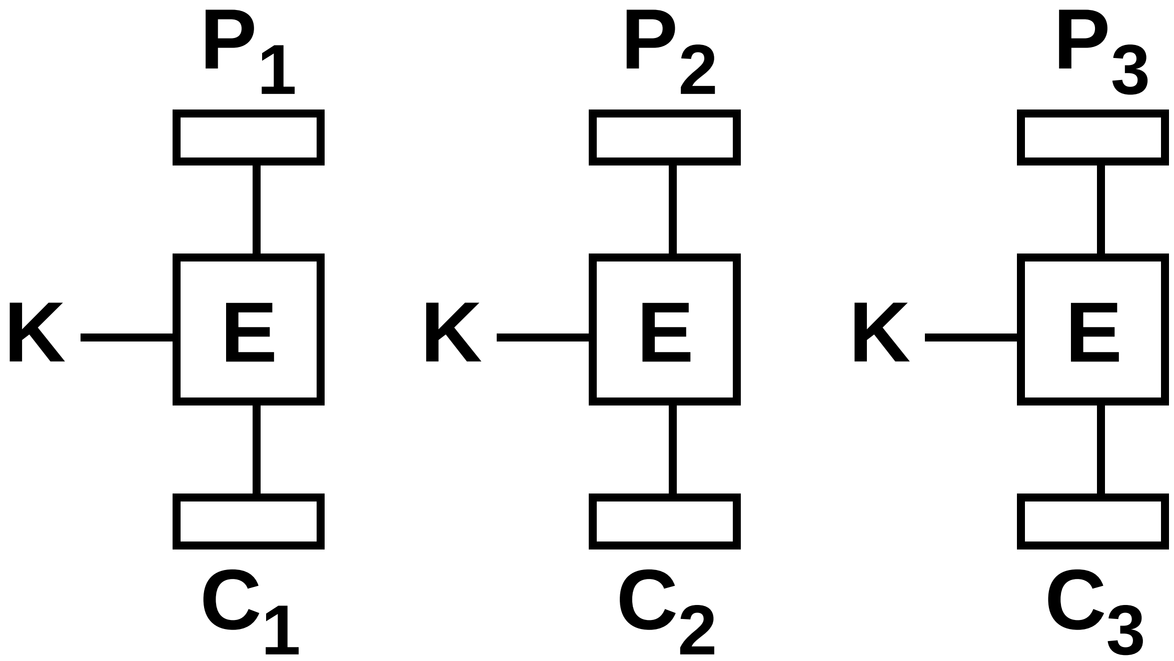
Block cipher in a stream cipher mode

- Stream cipher: plaintext is XORed with keystream generated from secret key and initialization vector (IV)

Electronic Code Book (ECB)

- Natural approach for encryption: given a message M, split M up into blocks of size b bits (where b = input size of block cipher)
- Ciphertext = $\{M_1\}_K \{M_2\}_K \ldots \{M_n\}_K$
- This approach is called Electronic Code Book mode (ECB mode)
- Advantages
  - Simple to compute
- Disadvantages
  - Same plaintext always corresponds to same ciphertext
  - Traffic analysis yields which ciphertext blocks are equal → know which plaintext blocks are equal
  - Adversary may be able to guess part of plaintext, can decrypt parts of a message if same ciphertext block occurs
  - Adversary can replace blocks with other blocks

ECB Mode

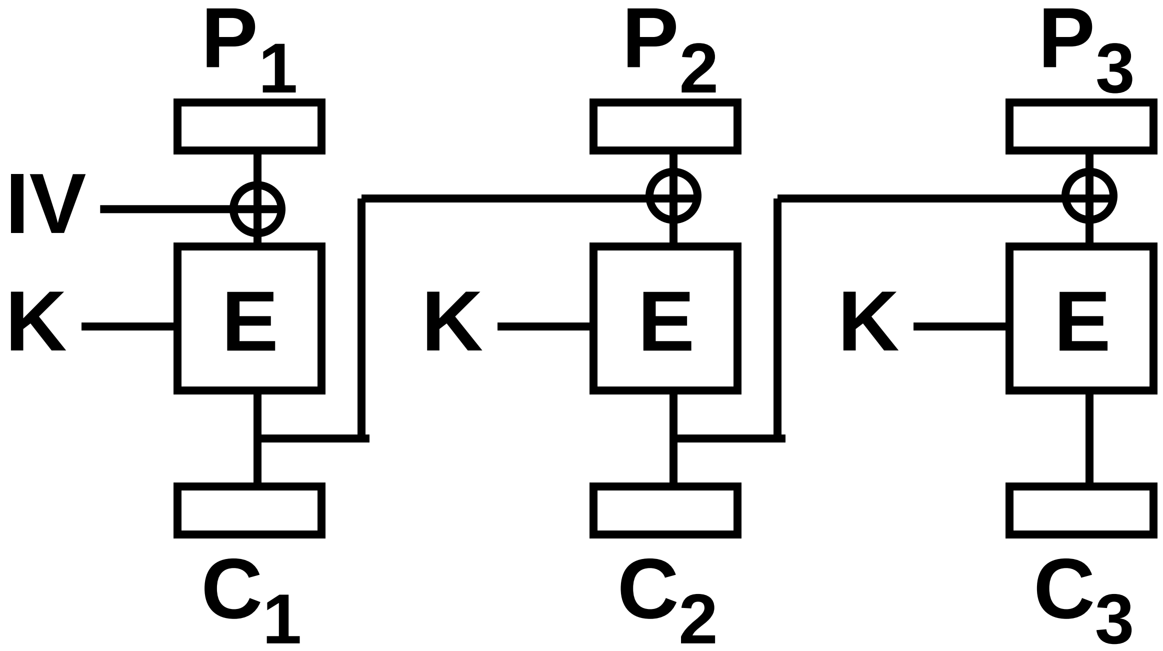


Desired Properties

- Semantic security: if adversary had to guess value of a single plaintext bit, can guess at best at random
  - Even if we keep encrypting a 1-bit message with skewed distribution (e.g., a fire alarm message, which almost always carries the same plaintext bit), attacker cannot guess value of plaintext given ciphertext
- Adversary cannot cut-and-paste blocks of ciphertext, otherwise complete message garbled

Cipher Block Chaining (CBC)

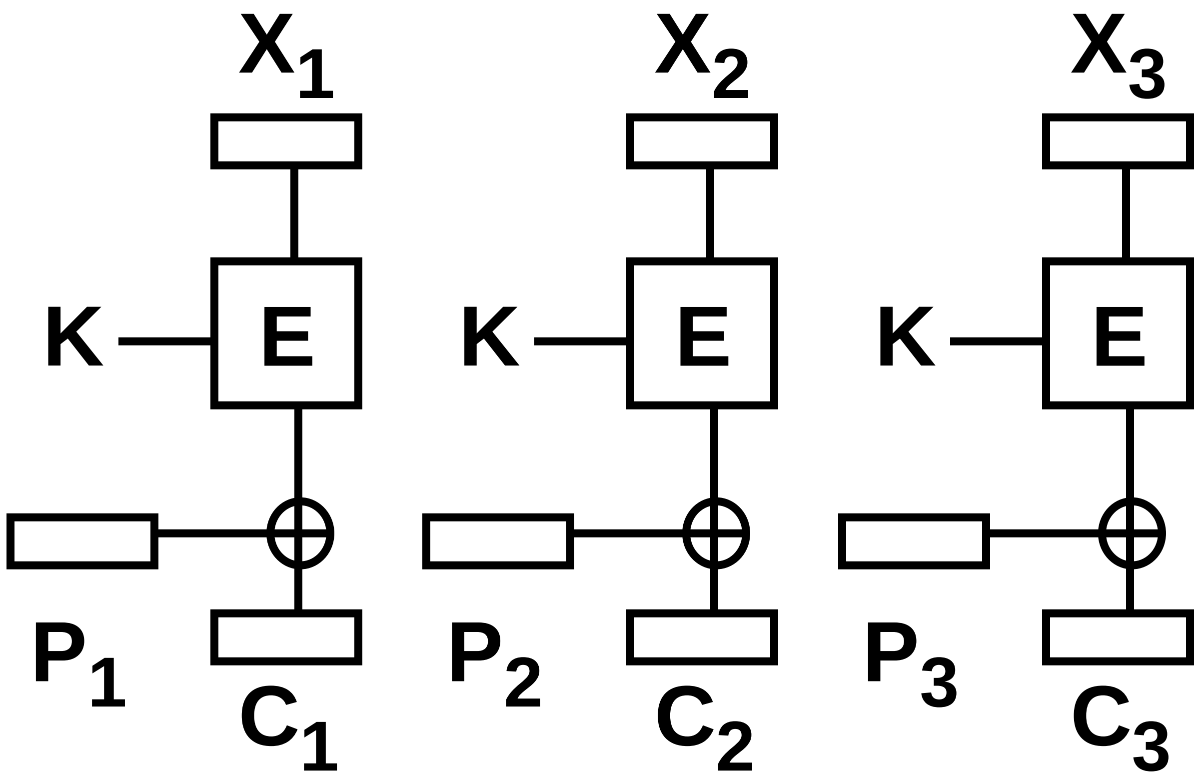
- $C_j = \{ P_j \oplus C_{j-1} \}_K$
- C_0 = IV called initialization vector



- Advantages
  - Semantic security
- Disadvantages
  - Altered ciphertext only influences two blocks
  - Not secure for variable-sized messages!
  - However, self-synchronizing can be an advantage too!

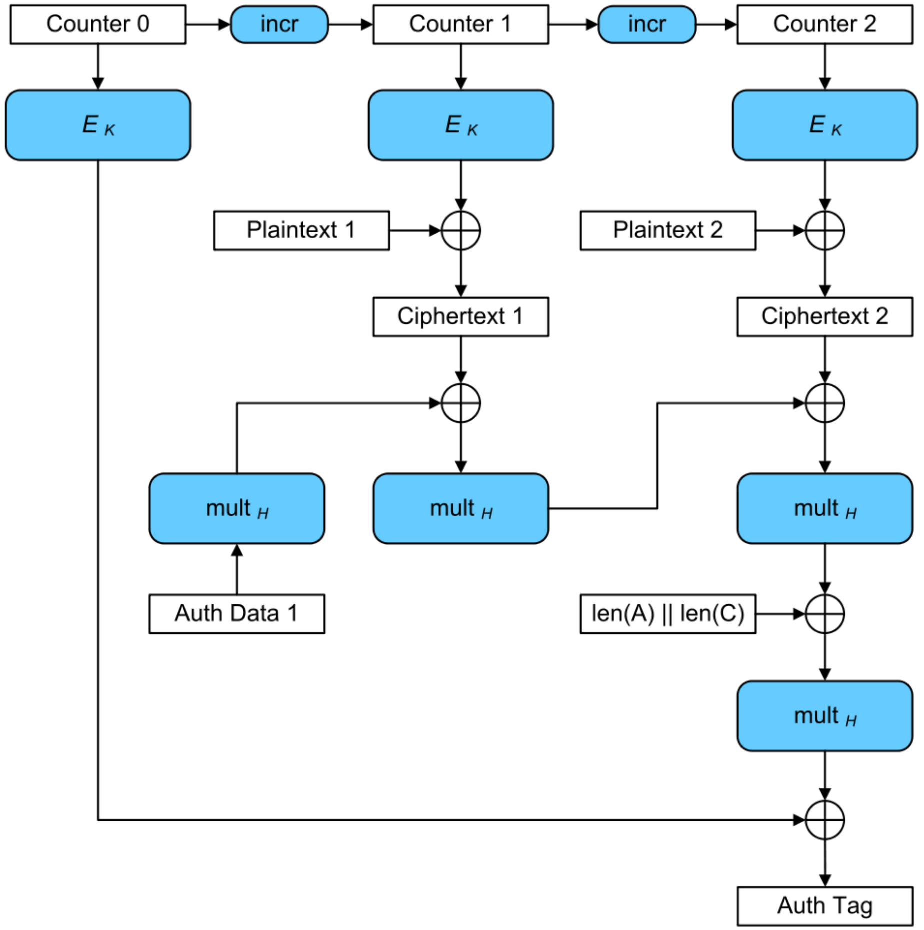
Counter Mode (CTR)

- X1 = IV called initialization vector
- Xj = X1 + i - 1
- Cj = { Xj }K ⊕ Pj



- Advantages
  - Semantic security
- Disadvantages
  - Altered ciphertext only influences single block
  - Same vulnerabilities as any stream cipher

Galois Counter Mode (GCM)

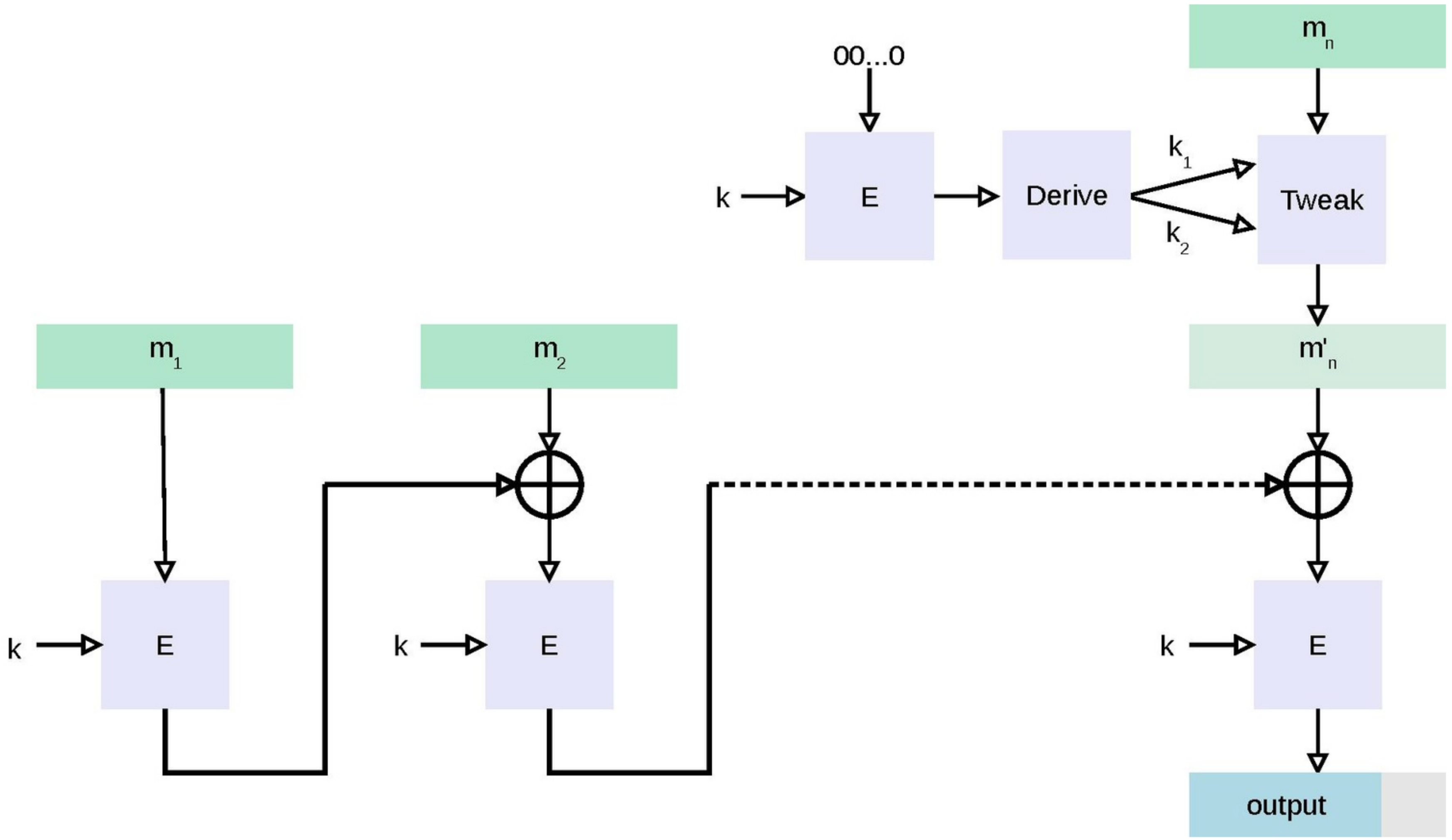


Graphic from https://en.wikipedia.org/wiki/Galois/Counter_Mode

Message Authentication Codes

- Message authentication codes (MAC) provide a “cryptographic checksum” for authentication, integrity
  - Write MAC(K, M), or MAC_K(M) (K is a shared symmetric key)
  - Intuition: MAC for a specific message can only be calculated when knowing the key K
- Use
  - A and B share symmetric key K_{AB}
  - A→B: M, MAC(K_{AB}, M)
- Hash-based MAC: HMAC
  - Example based on SHA256:
  - HMAC-SHA256(K, M)=SHA256(K ⊕ opad || SHA256(K ⊕ ipad || M))
  - ipad = 3636..36, opad = 5C5C..5C
- Block-cipher based MAC: CMAC
  - Fixes length vulnerabilities of CBC-MAC

Cipher-based Message Authentication Code (CMAC)



Graphic from https://en.wikipedia.org/wiki/One-key_MAC

Asymmetric Cryptography

Asymmetric Primitive: Diffie–Hellman

- Public values: large prime p, generator g
- Alice has secret value a, Bob has secret b
- A → B: g^a (mod p)
- B → A: g^b (mod p)
- Bob computes $(g^a)^b = g^{ab}$ (mod p)
- Alice computes $(g^b)^a = g^{ab}$ (mod p)
- Eve *cannot* compute g^{ab} (mod p)

Example

- a=3, b=6, g=2, p=11
- A → B: $g^a \pmod{p} = 2^3 \pmod{11} = 8$
- B → A: $g^b \pmod{p} = 2^6 \pmod{11}$

  $= 64 \pmod{11} = 9$
- Bob computes $(g^a)^b \pmod{p} = 8^6 \pmod{11}$

  $= 262144 \pmod{11} = 3$
- Alice computes $(g^b)^a \pmod{p} = 9^3 \pmod{11}$

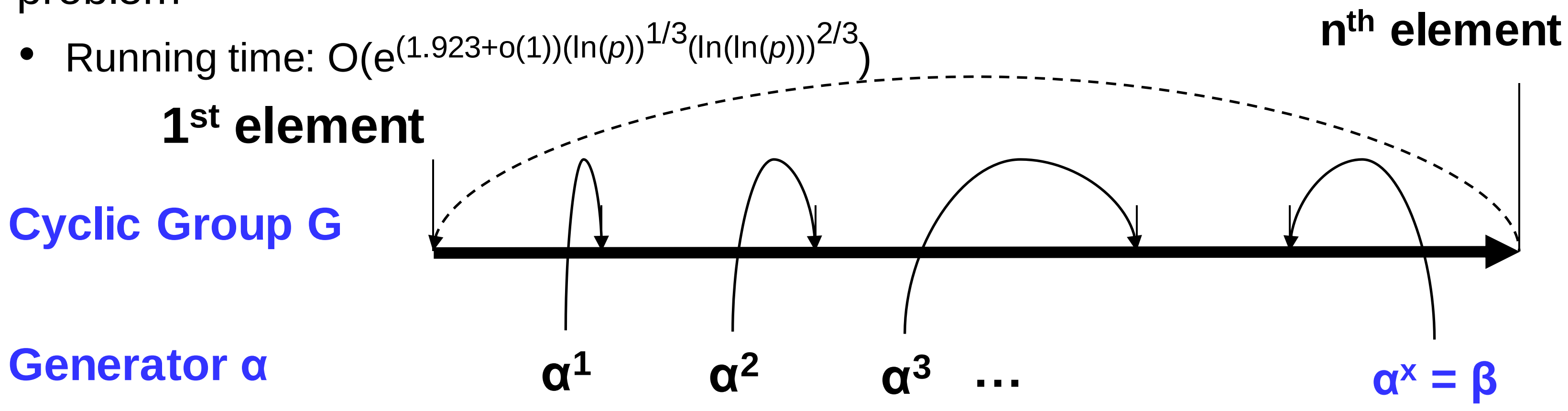
  $= 729 \pmod{11} = 3$

Discrete Logarithm Problem

- Public values: large prime *p*, generator *g*
- $g^a \bmod p = x$
- Discrete logarithm problem: given *x*, *g*, and *p*, find *a*
- Table *g*=2, *p*=11

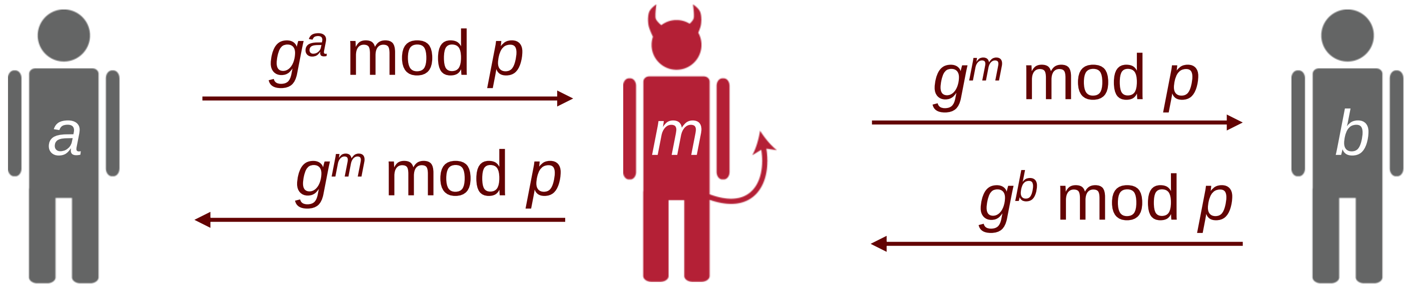
a	1	2	3	4	5	6	7	8	9	10
x	2	4	8	5	10	9	7	3	6	1

- Number field sieve is fastest algorithm known today to solve discrete logarithm problem
  - Running time: $O(e^{(1.923+o(1))(\ln(p))^{1/3}(\ln(\ln(p)))^{2/3}})$



Problem: Man-in-the-Middle Attack

- Public values: large prime p, generator g
- Problem: in Man-in-the-Middle attack, Mallory impersonates Alice to Bob and Bob to Alice



$$g^{am} \bmod p \neq g^{bm} \bmod p$$

Asymmetric Primitive: RSA

- Invented by Rivest, Shamir, Adleman 1978
- Let p, q be large secret primes
- Pick e, compute d so $ed \equiv 1 \bmod \phi(pq)$
  - Public key: $N=pq$, e
  - Private key: p, q, d
- Signature generation of message M
  $\Sigma = M^d \bmod N$
- Signature verification:
  $\Sigma^{\mathrm{e}} = M^{ed} = M^{1 + K\phi(pq)} = M \pmod{N}$

Example

- p=3, q=7, N=21, $\phi(pq)$=12, e=5, d=5
- $\phi(pq)$= (3-1)*(7-1) = 2*6 = 12
- e=5
  - Want d such that $ed \equiv 1 \bmod \phi(pq) = 1 \bmod 12$
  - Pick d=5
- M=2
  - Signature $\Sigma = M^d \bmod N = 2^5 \bmod 21 = 11$
  - Signature verification:
    $\Sigma^e = (M^d)^e \bmod 21 = 11^5 \bmod 21$
    $= 161051 \bmod 21 = 2$
- Could also pick e=7, d=7…

Encrypted Key Exchange (EKE)
DH Protocol

- A, B share password P, want to authenticate each other and establish a shared secret key
- $K = \mathrm{H}(P)$, A picks random a, B picks random b
- 1: A → B: $\{ g^a \}_K$
- $K' = \mathrm{H}(g^{ab})$
- 2: B → A: $\{ g^b \}_K$, $\{ N_{\mathrm{B}} \}_{K'}$
- 3: A → B: $\{ N_{\mathrm{A}} , N_{\mathrm{B}} \}_{K'}$
- 4: B → A: $\{ N_{\mathrm{A}} \}_{K'}$
- Dictionary attacks? (Enables verification by attacker if a given password was correct or not.)

Difference between Authentication and Signature

- Authentication enables the receiver to verify origin, but receiver cannot convince a third party of origin
- Signature enables the receiver to verify origin, and receiver can convince third party of origin as well
- Signature also provides authentication

Comparison Symmetric vs Asymmetric Crypto

Symmetric crypto

- Need **shared** secret key
- 128 bit key for high security (year 2020)
- ~100,000,000 ops/s on 5GHz processor
- 10x speedup in HW

Asymmetric crypto

- Need **authentic** public key
  → public-key infrastructures (PKIs)
- 3072 bit key (RSA), 384 bit key (EC) for high security (year 2020)
- ~1000 signatures/s
  ~10000 verify/s (RSA) on 5GHz processor
- Limited speedup in HW

Hash Functions

Cryptographic Hash Functions

- Maps arbitrary-length input into finite length output
- Properties of a *secure* (cryptographic) hash function
  - One-way: Given y = H(x), cannot find x’ s.t. H(x’) = y
  - Weak collision resistance: Given x, cannot find x’ ≠ x s.t. H(x) = H(x’)
  - Strong collision resistance: Cannot find x ≠ x’ s.t. H(x) = H(x’)
- Example: MD5, SHA-1
  - Are they secure?

Attack Complexity: One-Wayness

- Assume secure hash function with n-bit output
- One-wayness: given output y, how many operations does it take to find any x, such that H(x) = y?
  - Assumption: best attack is random search
  - For each trial x, probability that output is y is 2^{-n}
  - P[find x after m trials]=$1-(1-2^{-n})^m$
  - Rule of thumb: find x after 2^{n-1} trials on average

Attack Complexity: *Weak* Collision Resistence

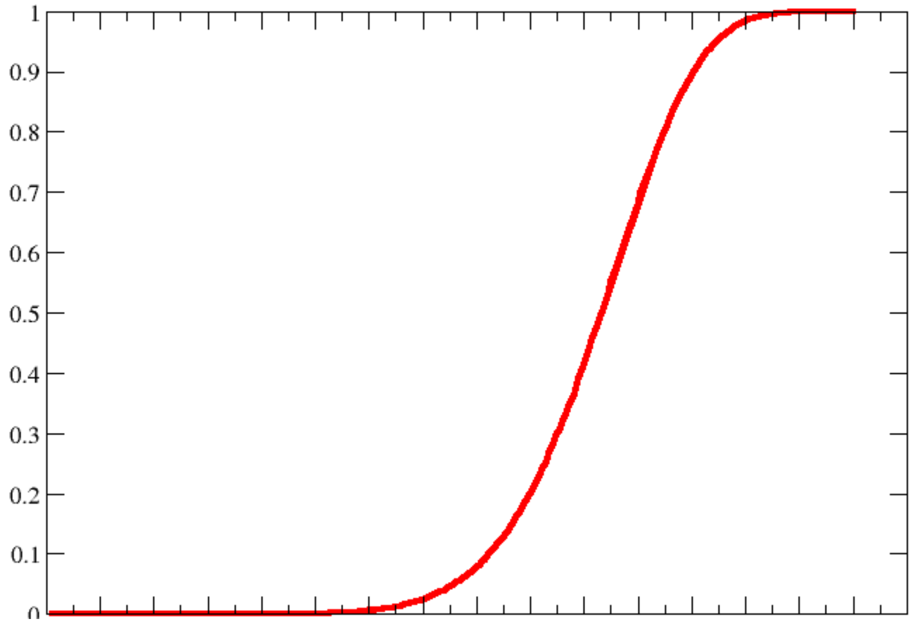
- Weak collision resistance (or second pre-image collision resistance): given input x, how many operations does it take to find another x' ≠ x, s.t. H(x) = H(x')?
  - Assumption: best attack is random search
  - For each trial x', probability that output is equal is 2^{-n}
  - P[find x after m trials]=$1-(1-2^{-n})^m$
  - Rule of thumb: find x' after 2^{n-1} trials on average

Attack Complexity: *Strong* Colllision Resistence

- Strong collision resistance: how many operations does it take to find x and x', s.t. $x' \neq x$ and $H(x) = H(x')$?
  - Assumption: best attack is random search
  - Algorithm picks random x', checks whether H(x') matches any other output value previously seen
  - P[find col after m trials]=
    $1-(1-1/2^n)(1-2/2^n)(1-3/2^n)\ldots(1-(m+1)/2^n)$
  - Rule of thumb: find collision after $2^{n/2}$ trials on average
    - ($1.17*2^{n/2}$ to be a bit more precise)

Birthday Paradox

- How many people need to be in a room to have a probability > 50% that at least two people have the same birthday?
- Answer: approximately $1.17*365^{1/2} \sim 22.4$



Lack of Collision Resistance

- How good is life without collision insurance?
- No real effect on most protocols
  - SSL, IPsec, SSH, etc. use MD5 in three ways
    - Key expansion
    - HMAC
    - Signatures
  - In most use cases, not affected by collisions
- What about PKI certificates?
  - Register certificate for www.something.com and use certificate for www.bank.com if H(Cert www.something.com) = H(Cert www.bank.com)
  - Countermeasure?

One-Way Hash Chains

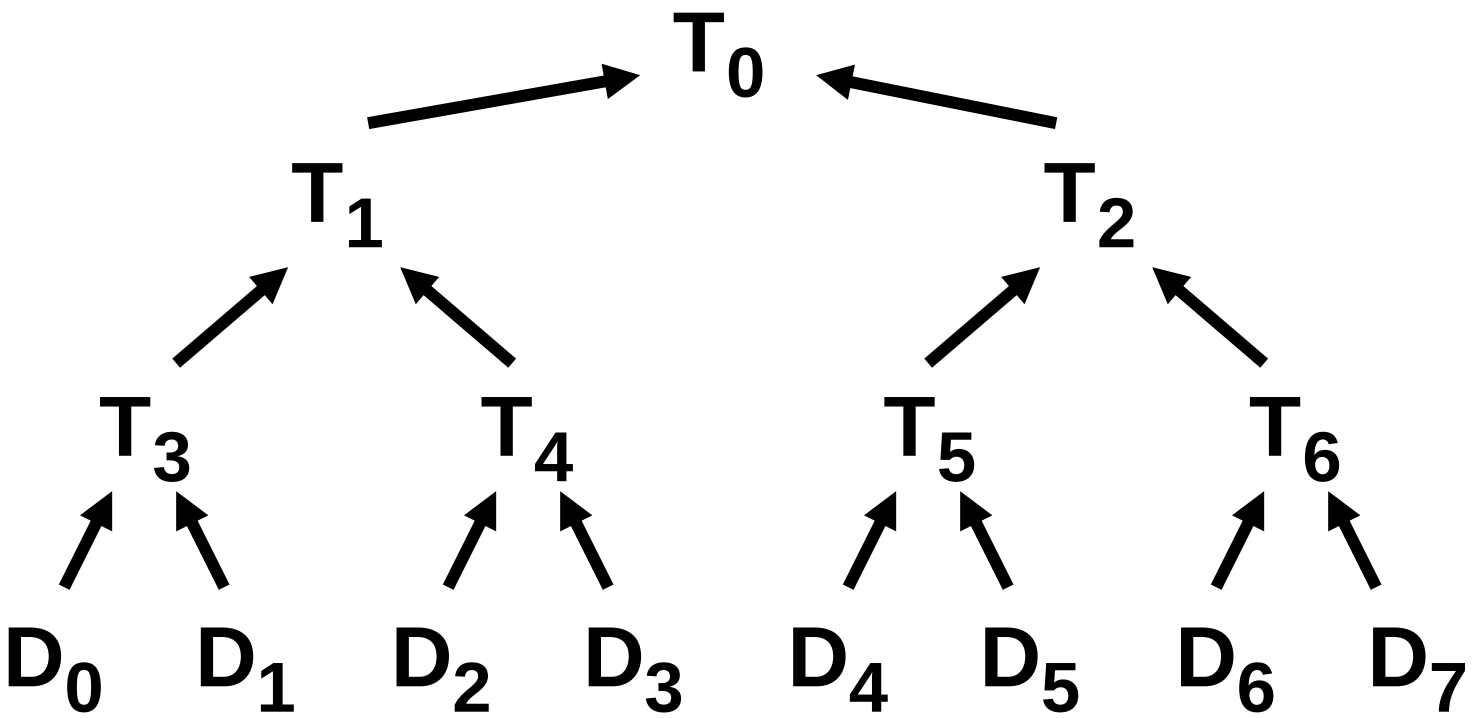
- Versatile cryptographic primitive
- Construction
  - Pick random r_N and public one-way function F
  - $r_i = F(r_{i+1})$
  - Secret value: r_N , public value r_0

$$r_0 \xleftarrow{F} r_1 \xleftarrow{F} r_2 \xleftarrow{F} r_3 \xleftarrow{F} r_4$$

- Properties
  - Use in reverse order of construction: r_0 , r_1 … r_N
  - Infeasible to derive r_i from r_j ($j<i$)
  - Efficiently authenticate r_i using r_j ($j<i$): $r_j = F^{i-j}(r_i)$
  - Robust to missing values

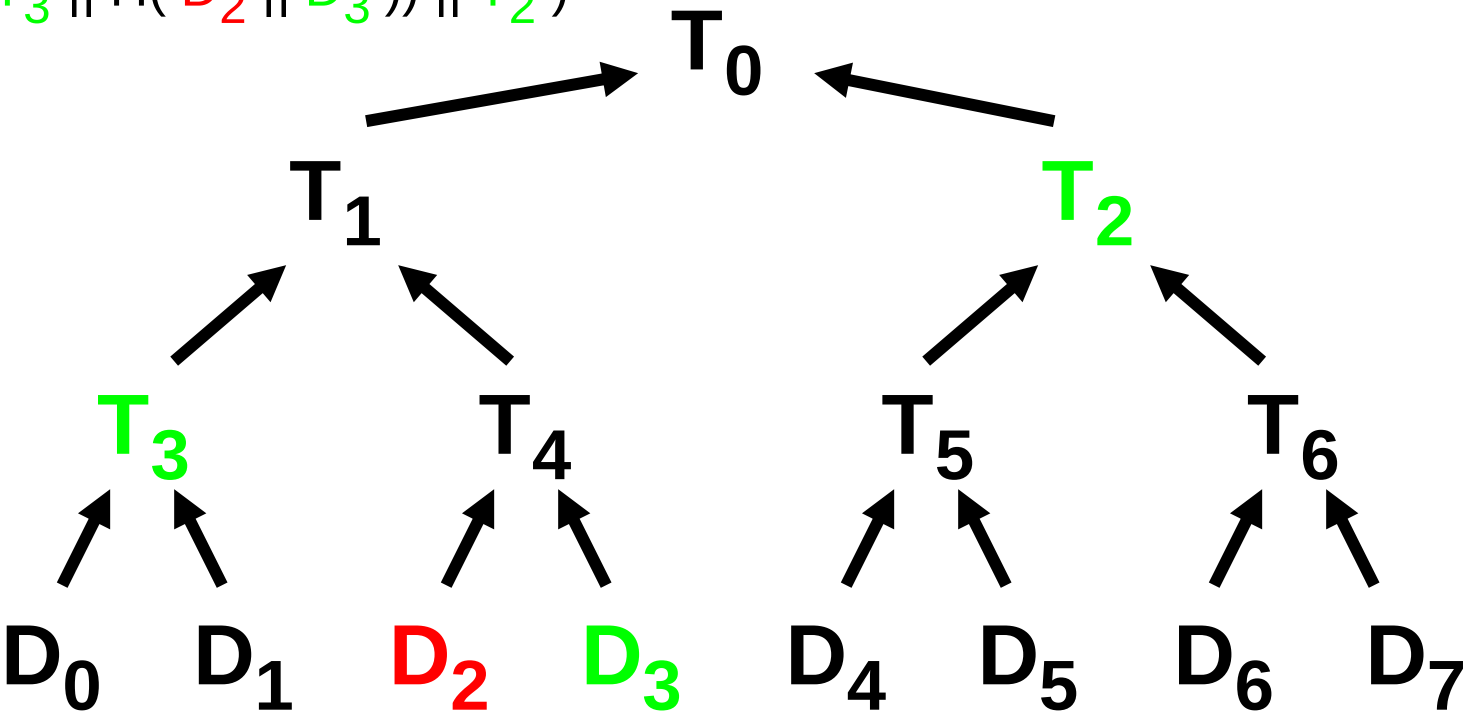
Merkle Hash Trees

- Authenticate a sequence of data values D_0 , D_1 , …, D_N
- Construct binary tree over data values



Merkle Hash Trees

- Verifier knows T_0
- How can verifier authenticate leaf D_i ?
- Solution: recompute T_0 using D_i
- Example authenticate D_2 , send D_3 T_3 T_2
- Verify $T_0 = H(\, H(\, T_3 \,||\, H(\, D_2 \,||\, D_3 \,)) \,||\, T_2 \,)$



Selecting Cryptographic Key Lengths

- Useful resources
  - https://www.keylength.com
  - https://en.wikipedia.org/wiki/Key_size

Method	Date	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul	2037	99	2986 2464	175	2986	186	197
[2] Lenstra Updated	2032	90	2278 3034	179	2278	179	179
[3] ECRYPT II	2031 - 2040	128	3248	256	3248	256	256
[4] NIST	2016 - 2030 & beyond	128	3072	256	3072	256	256
[5] ANSSI	> 2030	128	3072	200	3072	256	256
[6] IAD-NSA	-	256	3072	-	-	384	384
[7] RFC3766	-	123	2986	246	2986	231	-
[8] BSI	> 2022	128	3000	250	3000	250	256

Powers of Two Conversion & Useful Units

- $2^n = 10^m$
  $m \sim (n/10) * 3$
  $n \sim (m/3) * 10$
- Fast conversion trick:
  - $2^{10} \sim 10^3$, $2^{20} \sim 10^6$, $2^{30} \sim 10^9$
  - $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, 128, 256, 512
- Seconds per day $\sim 2^{16}$, seconds per year $\sim 2^{25}$
- Schneier's Applied Cryptography, p18
  - Probability to get hit by lightning per day (10^{-10} , 2^{-33})
  - Number of atoms on earth (10^{51} , 2^{170})
  - Number of atoms in the universe (10^{77} , 2^{265})
  - Time until next ice age (14,000, 2^{14} years)
  - Duration until sun goes nova (10^9 , 2^{30} years)
  - Age of the Universe (10^{10} , 2^{33} years)