CS5321 Network Security
Week2: Crypto Basics (2)

Daisuke MASHIMA

http://www.mashima.us/daisuke/index.html

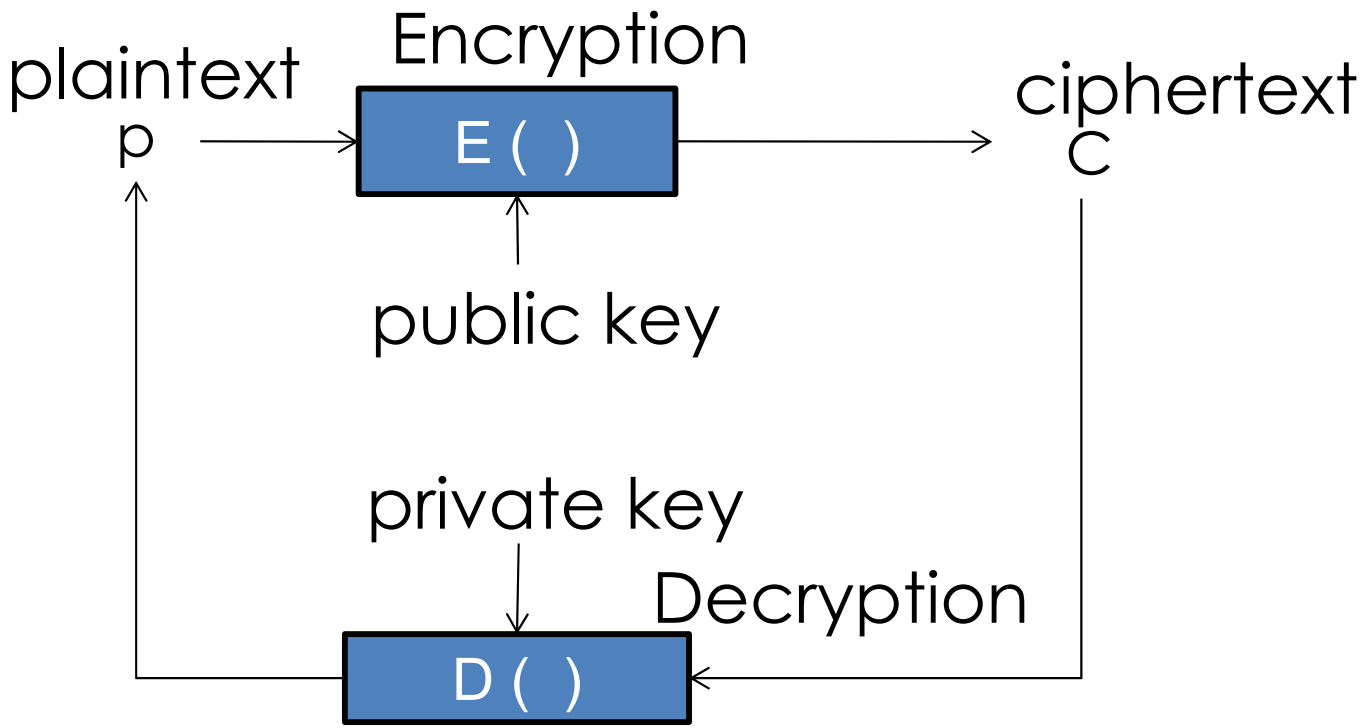
2022/23 Sem 2

Basic Cryptographic Primitives

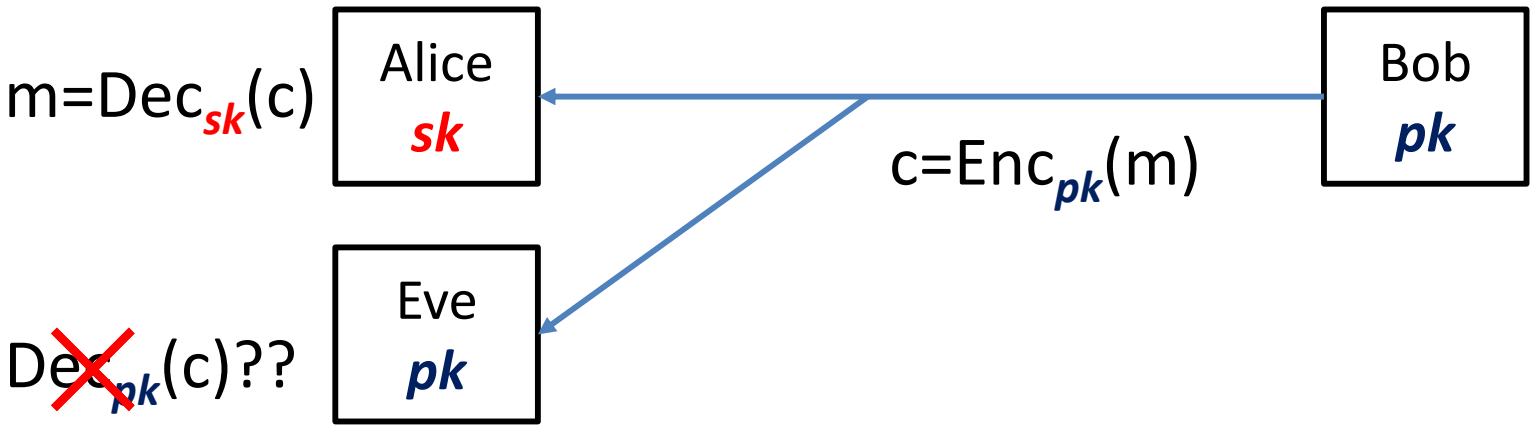
- Symmetric (shared-key, same-key)
  - Stream cipher (one-time pad)
  - Block cipher (pseudo-random permutation PRP)
  - Modes of encryption
  - Message authentication code (MAC)
  - Authenticated Encryption (Encryption + MAC)
- **Asymmetric (public-private key)**
  - **Diffie-Hellman key agreement**
  - **Public-key encryption**
  - **Digital signature**
- **Others (unkeyed symmetric)**
  - **Cryptographic hash function and more**

Asymmetric Encryption Primitives

- Also known as public-key cryptography
- The encryption key is different from the decryption key
- A pair of public key and private key
  - Only private key must be secret.

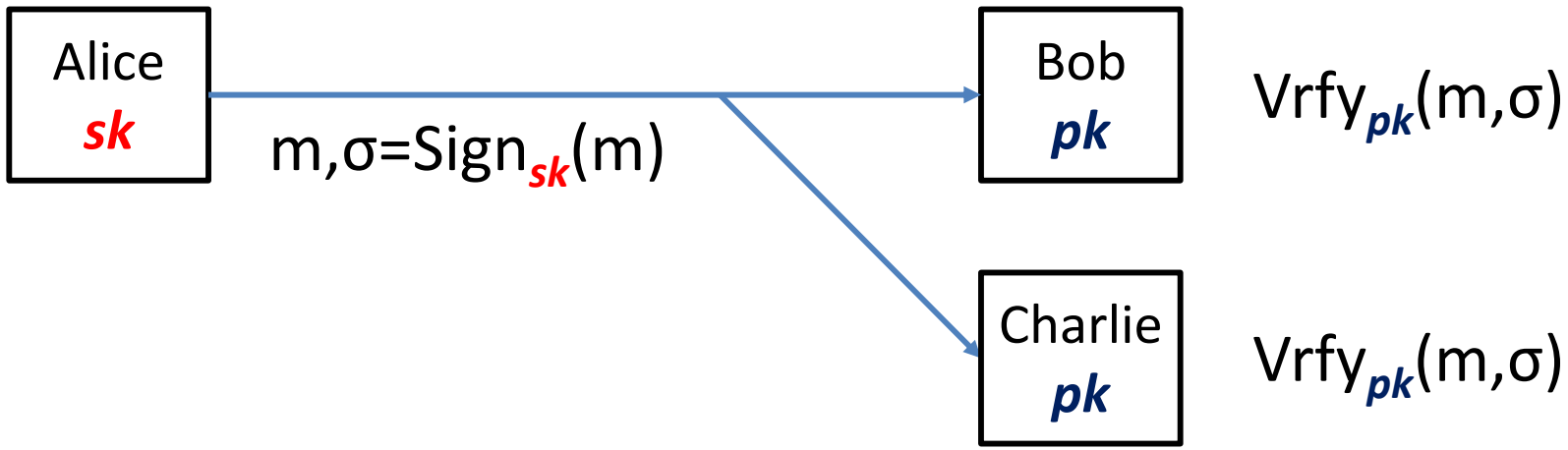


Public-key encryption



- Receiver (Alice) generates (***pk***,***sk***) and sends ***pk*** to Bob
  - or publicize her ***pk*** (e.g., her webpage, central database)
  - assume that intended parties receive ***pk*** via *authenticated channels* (what if there's no such channel? We'll learn **PKI**)
- Anyone with ***pk*** can encrypt message
- Only the one with ***sk*** can decrypt message
  - cannot decrypt with ***pk*** (Eve may encrypt but can't decrypt)

Digital signature

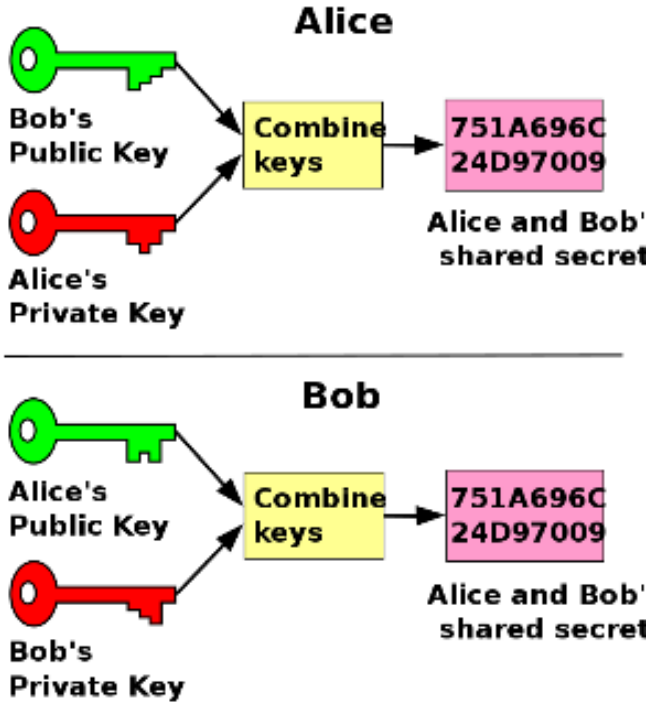


- Only the one with ***sk*** can generates signature σ for message m
- Anyone with ***pk*** can verify the signature
  - assume that intended parties receive ***pk*** via *authenticated channels*
- **Non-repudiation**: signed document becomes proof that Alice indeed signed the document

Asymmetric Encryption Primitive: DH

- **Diffie-Hellman (DH) key agreement**
  - 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)
  - Can Eve compute g^{ab} (mod p) ?

    => Secure against *eavesdropper*
  - Why? **Discrete Logarithm Problem** is conjectured to be hard!



(Wikipedia)

Discrete Logarithm Problem

- Public values: **large prime p**, **generator g** of cyclic group G
  - A group G consisting of numbers relatively prime to p is a cyclic group. Generator g is an element of G that satisfies $g^{p-1} = 1 \bmod p$ (only p-1 satisfies this)
    - https://eli.thegreenplace.net/2019/diffie-hellman-key-exchange/
- $g^a \bmod p = x$
- 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	11
x	2	4	8	5	10	9	7	3	6	1	2

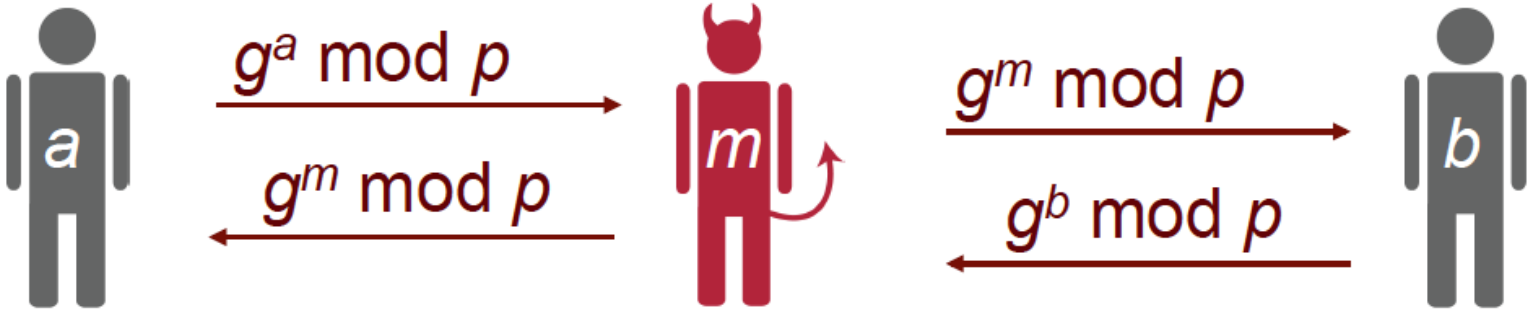
- **Number field sieve** is fastest algorithm known today to solve discrete logarithm problem
  - Running time: $O(e^{(1.923+o(1))(\ln(p))^{1/3}(\ln(\ln(p)))^{2/3}})$ **<- Subexponential**
- Going back to DH Protocol:
  - Eve cannot find a (or b), and thus cannot compute $g^{ab} \pmod p$

Example

- a=3, b=6, g=2, p=11
  - A→B: $g^a \pmod p = 2^3 \pmod{11} = \mathbf{8}$
  - B→A: $g^b \pmod p = 2^6 \pmod{11} = 64 \pmod{11} = \mathbf{9}$
  - Bob computes $(g^a)^b \pmod p = \mathbf{8}^6 \pmod{11}$
    $= 262144 \pmod{11} = 3$
  - Alice computes $(g^b)^a \pmod p = \mathbf{9}^3 \pmod{11}$
    $= 729 \pmod{11} = 3$

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



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

Asymmetric Primitive: RSA

- **RSA algorithm**
  - Invented by Rivest, Shamir, Adleman in 1978
  - Let p, q be large secret primes
  - Pick e, compute d so ed ≡ 1 mod ϕ(pq)

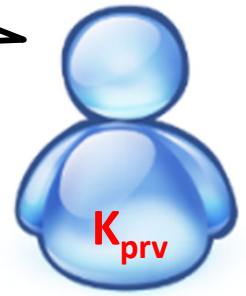
    Relatively prime to ϕ(pq)

    = 1 + v ϕ(pq)

  Φ(n) counts the number of positive integers up to n that are relatively prime to n. when p, q are prime numbers, Φ(pq) = (p-1)(q-1)

  - Public key **N**=pq, **e**
  - Private key **p**, **q**, **d**

  - **Only** I can **decrypt**
  - **Only** I can **sign**



  - Encryption/decryption of message M
    - Encryption: $C = M^{e} \bmod N$
    - Decryption: $M = C^{d} = M^{ed} \bmod N$ $(= M^{1 + v\phi(N)} \bmod N = M^{v\phi(N)} M \bmod N)$

      $M^{\Phi(N)} \bmod N = 1$
      (Fermat-Euler genralization)
  - Signing message M
    - Signature generation: $\Sigma = M^{d} \bmod N$
    - Signature verification: $\Sigma^{e} = M^{ed} = M^{1 + v\phi(N)} = M \pmod N$
  - Secure? **Large prime factorization** is conjectured to be hard!

Example

- **p**=3, **q**=7, **N**=21, ϕ(**pq**)=12, **e**=5, **d**=5
  - ϕ(**pq**)=(3-1)*(7-1)=2*6=12
  - **e**=5
    - Want d such that **ed** = 1 mod ϕ(**pq**) = 1 mod 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 …

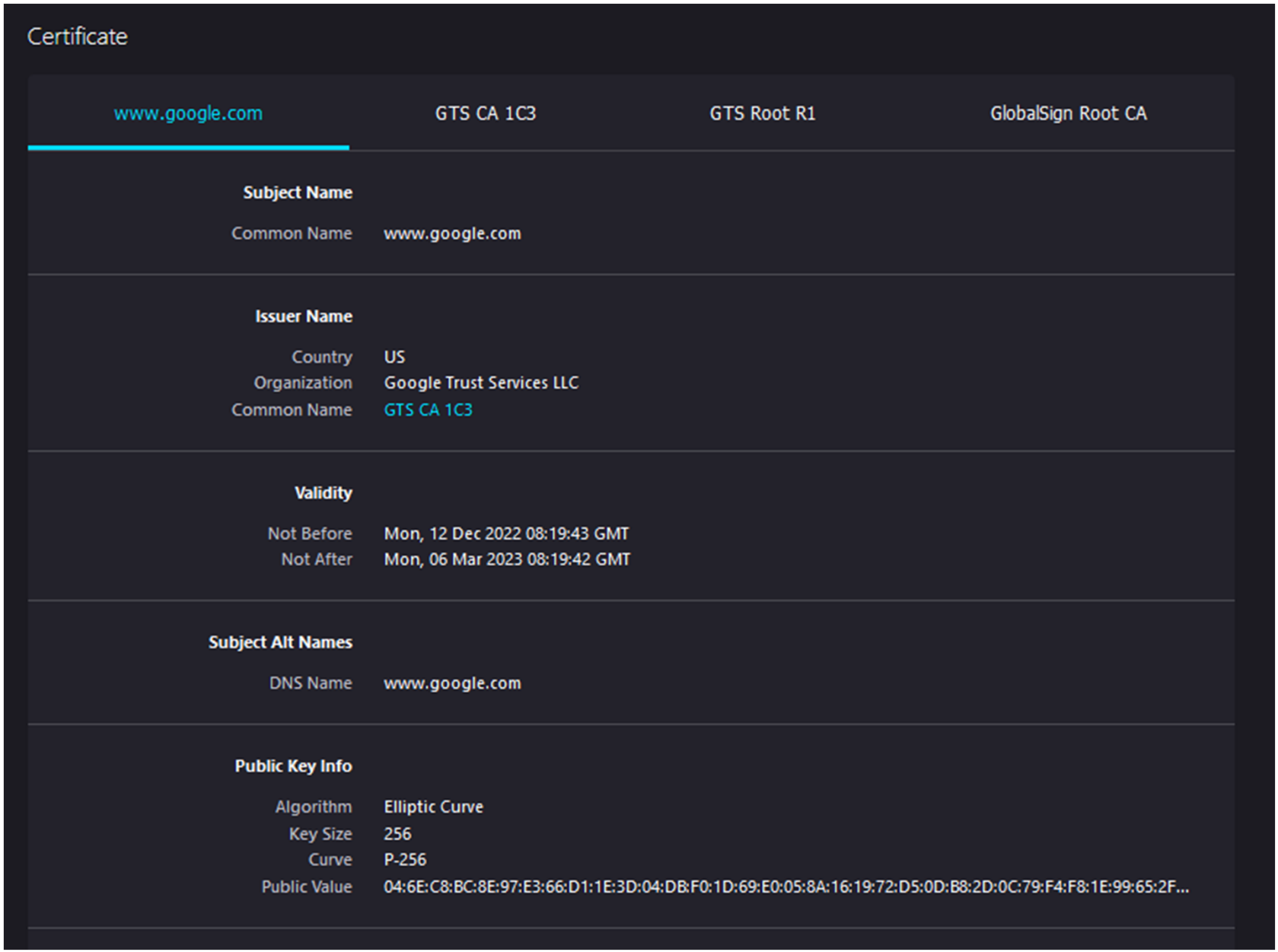
PKI (Public-key Infrastructure) Overview

- We need an **infrastructure to distribute keys**, in particular, the public key. This is generally known as key distribution or key management.
  - "RFC 4949 defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to **create, manage, store, distribute, and revoke digital certificates** based on asymmetric cryptography"
- Three ways of key distribution:
  - Public announcement (e.g., Email, web site)
  - Publicly-available directory (e.g., PGP public key server)
  - **Certificate**

Certificate and Certification Authority

- A **(digital) certificate** is a digital document that contains at least the following 4 main items
  1) The identity of an entity
  2) The associated public key
  3) The time window that this certificate is valid.
  4) The **signature of the CA**
  - Typically, it has other meta-information. Including (serial number, purpose, crypto algorithms)
- The CA keeps a **directory of public keys**.
- The CA has its own public-private key. The CA's public key has been securely distributed to all entities involved.
- CA is also responsible for **revocation** of certificates by maintaining CRL (certificate revocation list)

Certificate Example



Comparison **Sym** vs **Asym** Crypto

Symmetric crypto

- Need shared secret key
- 86 bit key for high security (year 2020)
- ~1,000,000 ops/s on 1GHz processor
- 10x speedup in HW

Asymmetric crypto

- Need authentic public key
- 2048 bit key (RSA) for high security
- ~100 signatures/s, ~1000 verify/s (RSA) on 1GHz processor
- Limited speedup in HW

Recommendation of Key size [Lenstra and Verheul]

	1982	1995	2002	2010	2020	2030	2040
Sym	56	66	72	78	86	93	101
RSA	417	777	1028	1369	1881	2493	3214

Basic Cryptographic Primitives

- Symmetric (shared-key, same-key)
  - Stream cipher (one-time pad)
  - Block cipher (pseudo-random permutation PRP)
  - Modes of encryption
  - Message authentication code (MAC)
  - Authenticated Encryption (Follow-up for Week 1)
- Asymmetric (public-private key)
  - Diffie-Hellman key agreement
  - Public-key encryption
  - Digital signature
- **Others (unkeyed symmetric)**
  - **Cryptographic hash function and more**

Hash function vs. cryptographic hash func

- Hash functions
  - functions that takes inputs of arbitrary length and compress them into short, fixed-length outputs (or **digests**)
  - H: $\{0,1\}^* \rightarrow \{0,1\}^l$
  - evaluation of H is efficient and algorithm is public

- Cryptographic hash functions usually require three properties:
  - Collision resistance (strong collision resistance)
  - Second preimage resistance (weak collision resistance)
  - Preimage resistance (oneway)

Notions of security (informal analysis)

- [P1] *Collision resistance*: it is computationally infeasible to find x, x' such that x'≠x and H(x')=H(x)

- [P2] *Second-preimage resistance*: **given x**, it is computationally infeasible to find x'≠x such that H(x')=H(x)

- [P3] *Preimage resistance*: **given y**, it is computationally infeasible to find x such that H(x)=y. (i.e., H is one-way)

Notions of security (informal analysis) (cont'd)

- [P1] *Collision resistance* implies [P2] *Second-preimage resistance*
  - if adversary can find x'≠x for given x such that H(x')=H(x), then it can clearly find a colliding pair x and x'.

- [P2] *Second-preimage resistance* implies [P3] *Preimage resistance*:
  - Assume H is second preimage resistance but not preimage resistance, then we show contradiction.
  - For given x (and y=H(x)), one can find x' that satisfies H(x')=y because H does not have preimage resistance.
  - When domain is very large, one can find x'≠x with high probability. Thus, contradiction.

Why are these properties important?

- What if an attacker can easily find a collision?

- Digital signature would not be reliable.
  - Digital signature is often made on the hash value of message

$$m,\sigma=\text{Sign}_{sk}(\mathbf{H(m)})$$

  - Finding collision m', an attacker can send the following!

$$m',\sigma$$

Attack Complexity: *Preimage resistance*

- Assume cryptographic hash function with n-bit output

- *G*iven 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 $\mathbf{2^{n-1}}$ trials on average

Attack Complexity: *Second-preimage 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 $\mathbf{2^{n-1}}$ trials on average

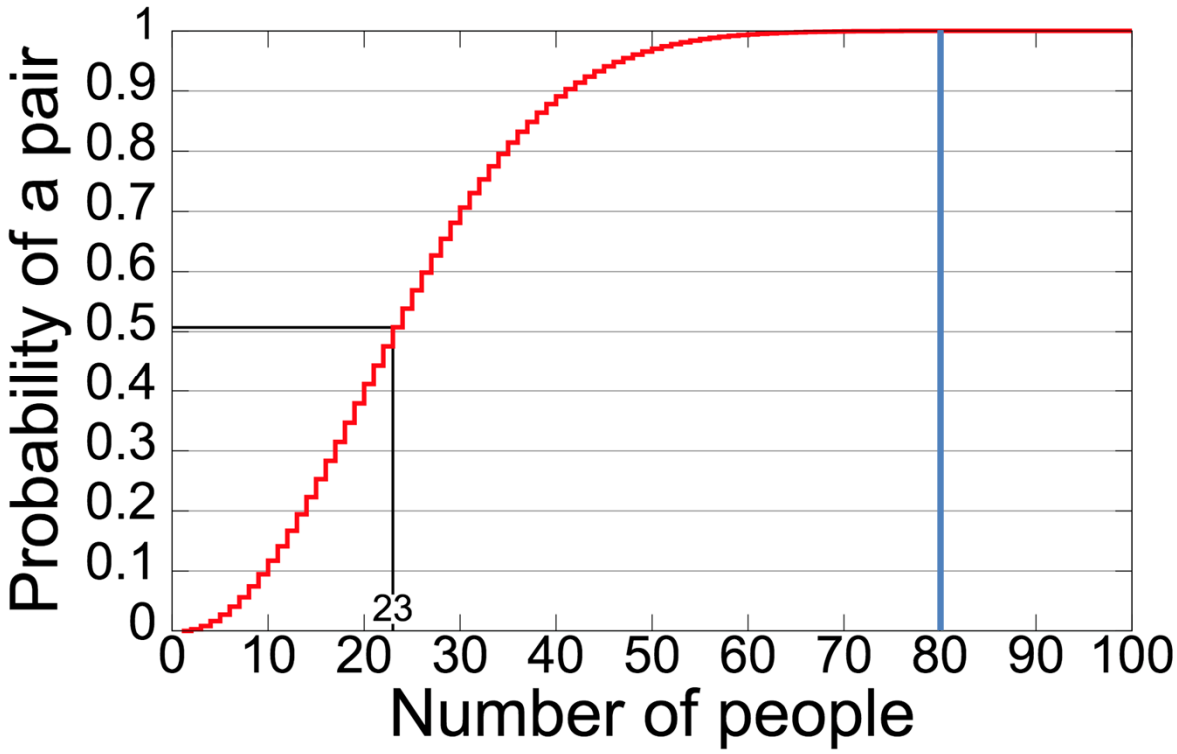
Attack Complexity: Collision Resistance

- How many operations does it take to find x and x', s. t. x' =/= 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 collision 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 (with 50% probability)
    - $2.146 * 2^{n/2}$ (with 90% probability)

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 $1.17*365^{1/2} \sim 22.4$



Implication of Birthday Paradox

- Let us consider a hash function that generates 40-bit digest

- How many hash calculations are needed to find collision with probability > 50%?
  - 1.17 * (2^40)^(1/2) = 1.17 * 2^20 = 1,226,833
- How many hash calculations are needed to find collision with probability > 90%?
  - 2.146* (2^40)^(1/2) = 2.146 * 2^20 =2,250,244

- Recent GPU (e.g., Geforce RTX 3080) can calculate 100million hash values per second.

- The size of hash value matters!

SHA-1: Broken! or Broken?

- Retires by the end of 2030

DARKReading Technology | Tech News and Analysis

DR Tech | 2 MIN READ | DR TECHNOLOGY

NIST Finally Retires SHA-1, Kind Of

SHA-1 was deprecated in 2011. NIST has set the hashing algorithm's final retirement date to Dec. 31, 2030.

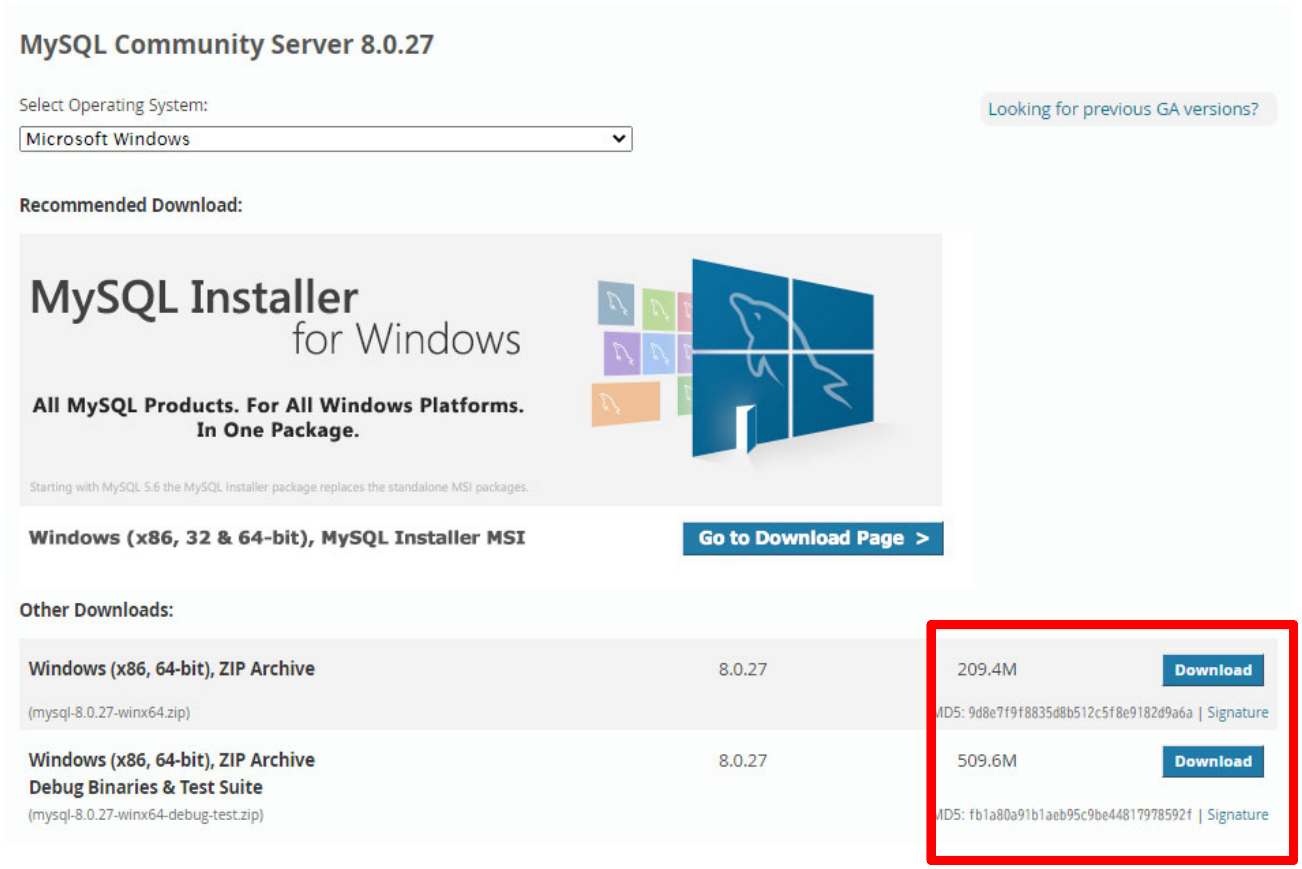
Dark Reading Staff
Dark Reading

December 16, 2022

- SHA-1 has a 160-bit message digest and thus has 80-bit strength
- In 2005 an attack shows it requires only 2^{69} operations to find a hash collision
  - Down to 2^{61} in 2010
- According to 2017 report, finding two different PDF files with same SHA-1 hash value required 9,223,372,036,854,775,808 (=2^{63}) operations
  - 6,500 years of single-CPU, 110 years of single-GPU

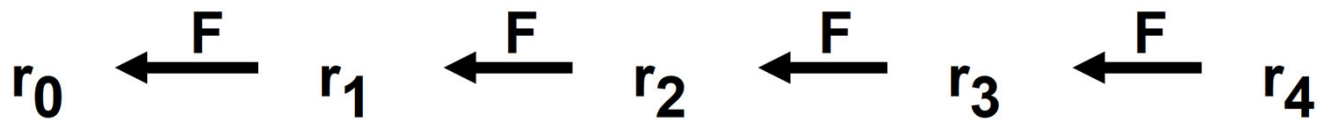
Fingerprinting

- Instead of storing the original data, one can simply stores a short hash digests
- Examples:
  - Virus fingerprinting
  - Deduplication
  - P2P file sharing



One-Way Hash Chains

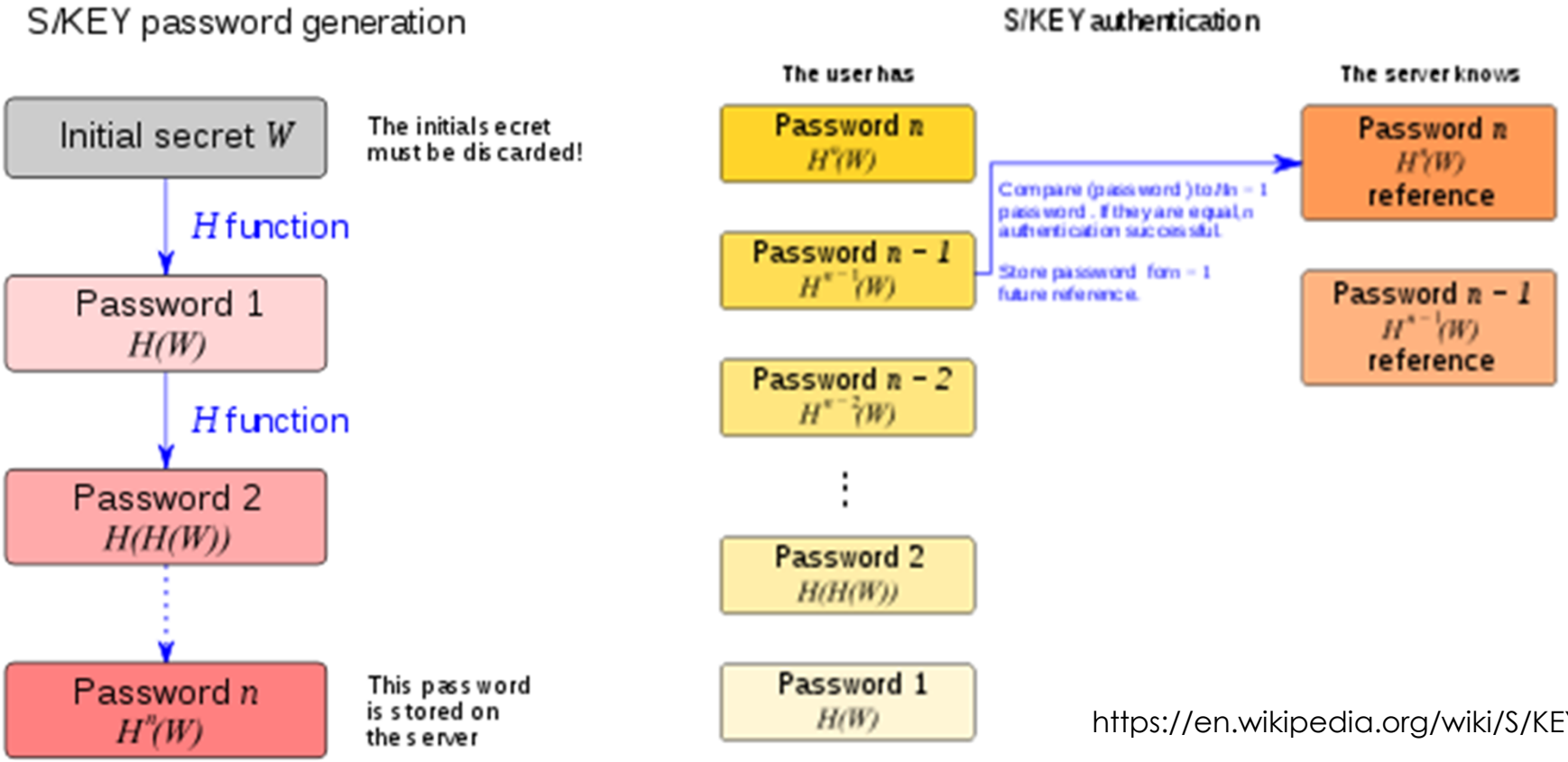
- Versatile cryptographic primitive
- Construction
  - Pick random r_N and public one-way function F
  - $r_i = F(r_{i+1})$
  - Secret value: r_N, public value: r_0



- Properties
  - Use in reverse order of construction: $r_0, r_1, \ldots, r_N$
  - Infeasible to derive r_i from r_j (j<i)

Application: S/KEY

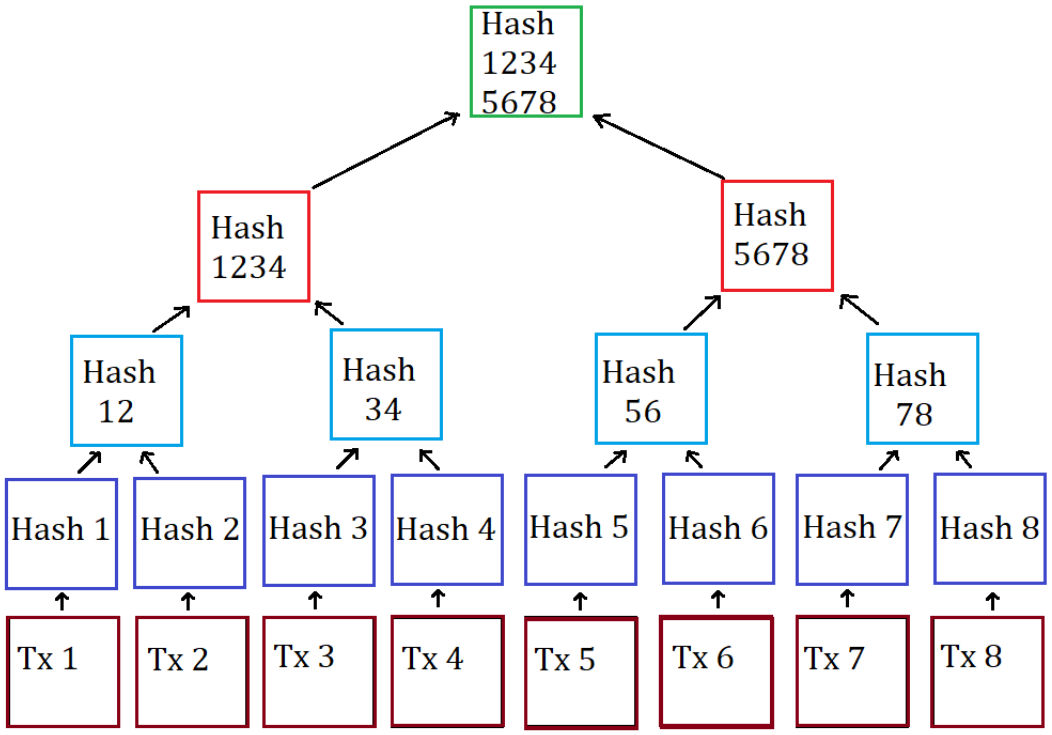
- One-time password authentication
- Consider a setting where user authenticates herself with password W over insecure communication channel
  - Password registration must be done in secure way.

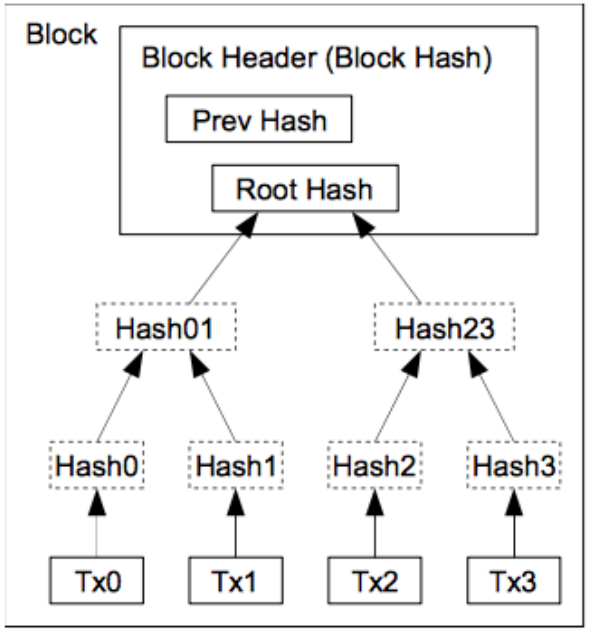


https://en.wikipedia.org/wiki/S/KEY

Merkle Hash Trees

- Authenticate a sequence of data values Tx_1, Tx_2, … Tx_N
- Construct binary tree over data values
- Widely used data structure, e.g., Blockchain





Transactions Hashed in a Merkle Tree

https://hackernoon.com/achieving-blockchain-scalability-with-sparse-merkle-trees-and-bloom-filters-3b9945f003f

Password hashing

- Passwords are not stored in plaintext but hashed and stored (e.g., *passwd* file of UNIX system)
- Preimage resistance of H makes inverting it difficult
- Attacks
  - password space is quite limited (e.g., dictionary of English words) => domain of H is small
  - pre-computation makes it even easier (rainbow table)
- Mitigations
  - slow hash functions (e.g., bcrypt)
  - salt: store (h,s) together when h=H(password||s), s is random
    - pre-computation is hard

Commitment Schemes

- C(m) = commitment to a value m; e.g., a bid in an auction
  - "hides" the content of m and "binds" the sender to value m
  - value of m revealed later (when auction is over)
  - e.g., r <-$- $\{0,1\}^l$ per commitment; C(m) = h(r‖m); later reveal r,m.
- Needs secure hash function that has preimage resistance, and collision resistance.

What's next?

- Basics for authentication and secure communication
  - Key Establishment / Distribution
  - Authentication using Symmetric Key
  - Authentication using PKI
  - Authentication using KDC
  - TLS

QUESTIONS?

Reading for Week 4: PKI Security

- **SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements** [IEEE S&P 2013]
  - A guided tour of history of PKI research and development!

- ***Certificate Transparency*** [COMMUNICATIONS OF THE ACM 2014]
  - Most widely used security mechanisms for certificate misissuance problems