Merkle Trees, Diffie-Hellman, Asymmetric Encryption

24 April 2025
Lecture 5

Slide Credits: Steve Zdancewic (UPenn)

Topics for Today

- Merkle Trees
- Key Establishment: Diffie-Hellman
- Asymmetric Encryption
  - RSA

- Source: HAC 8.2

Merkle Tree: Basic Idea

I have a lot of files, but I don't trust where they're stored

I want to let people (including me):

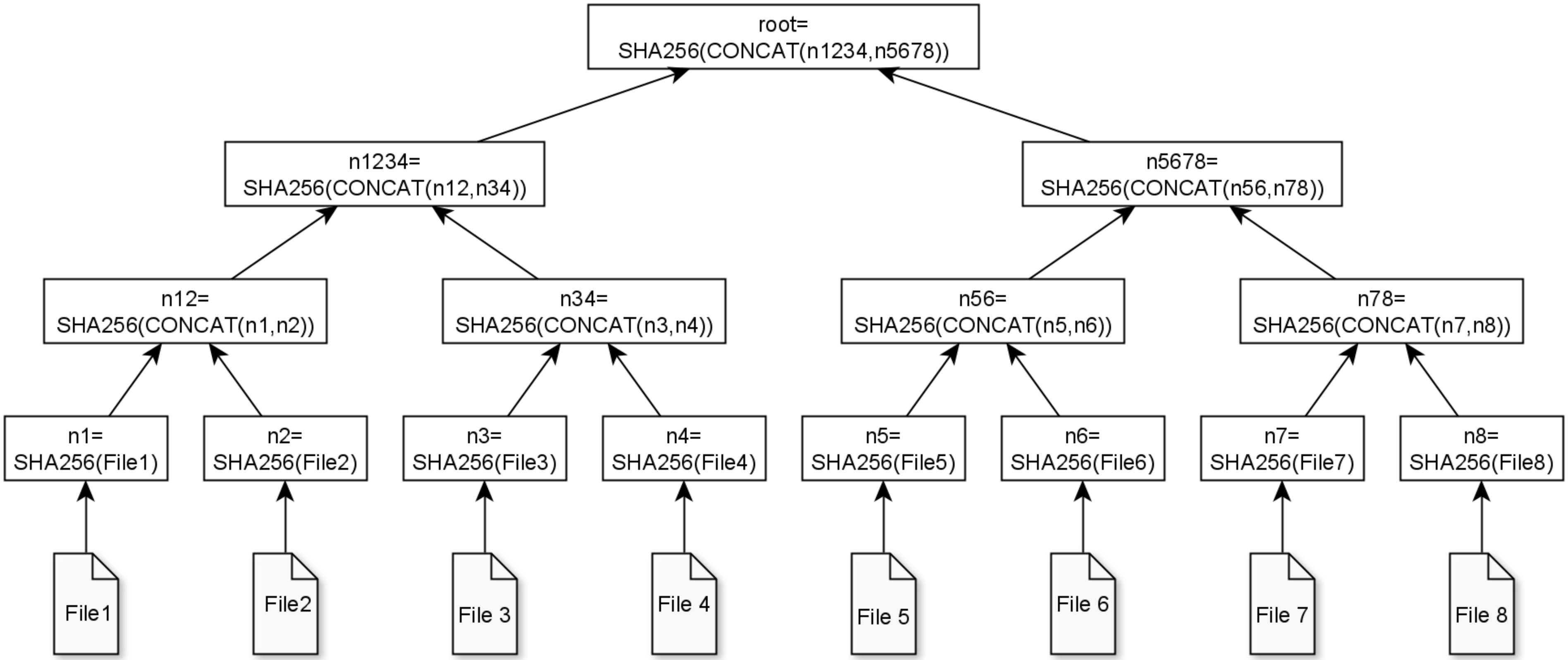
- Know if a particular file is in the collection
- Download a copy from an unsafe location and check it's ok

I want to add files and:

- Prove I only added a file (nothing removed)
- Prove I didn't change any files in the collection

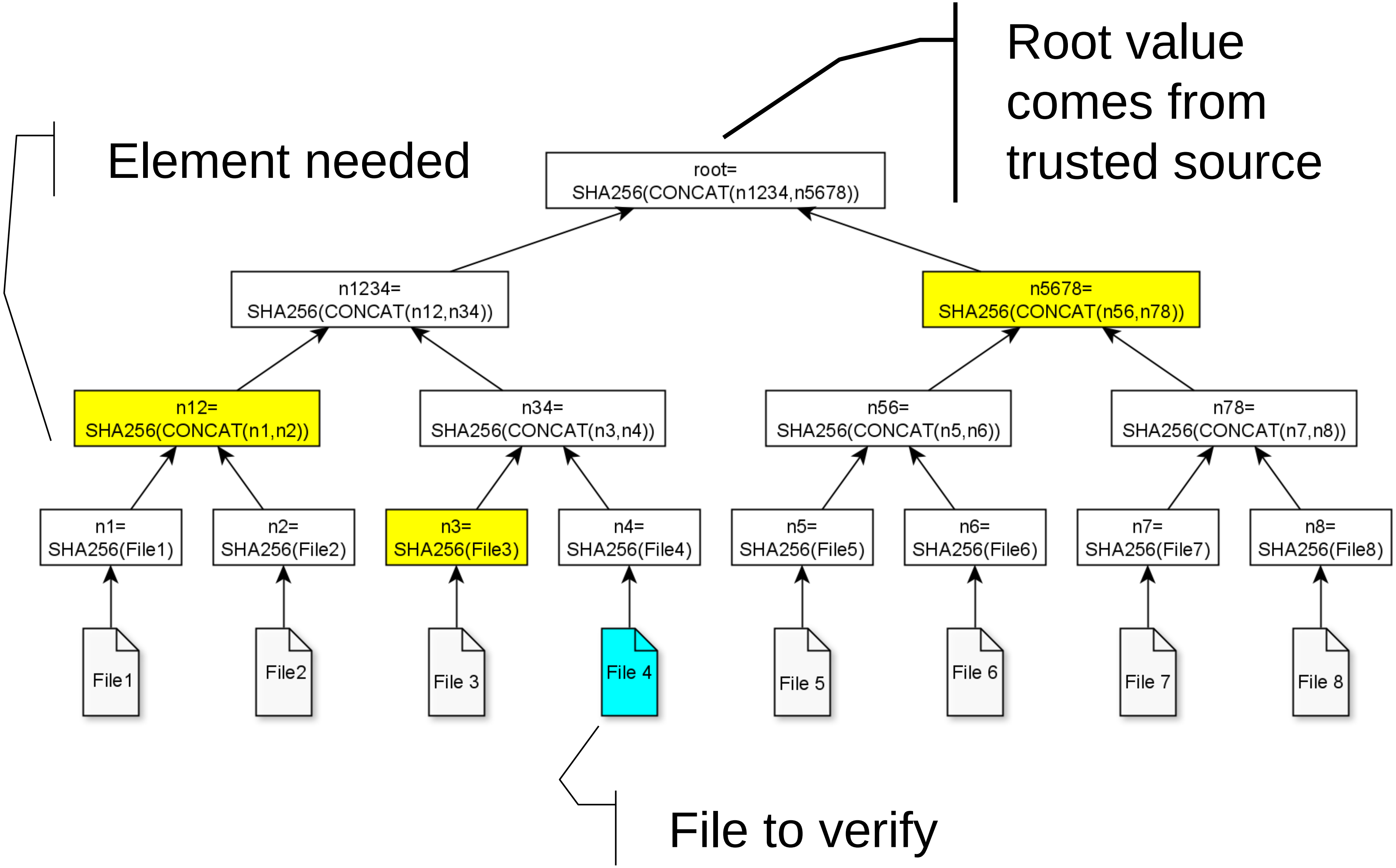
I want the operations to be fast

Building a Merkle Tree



This is a binary Merkle tree – $O(\log_2 n)$. Can also have more children or be unbalanced

Proof of Membership – File 4



Merkle Trees

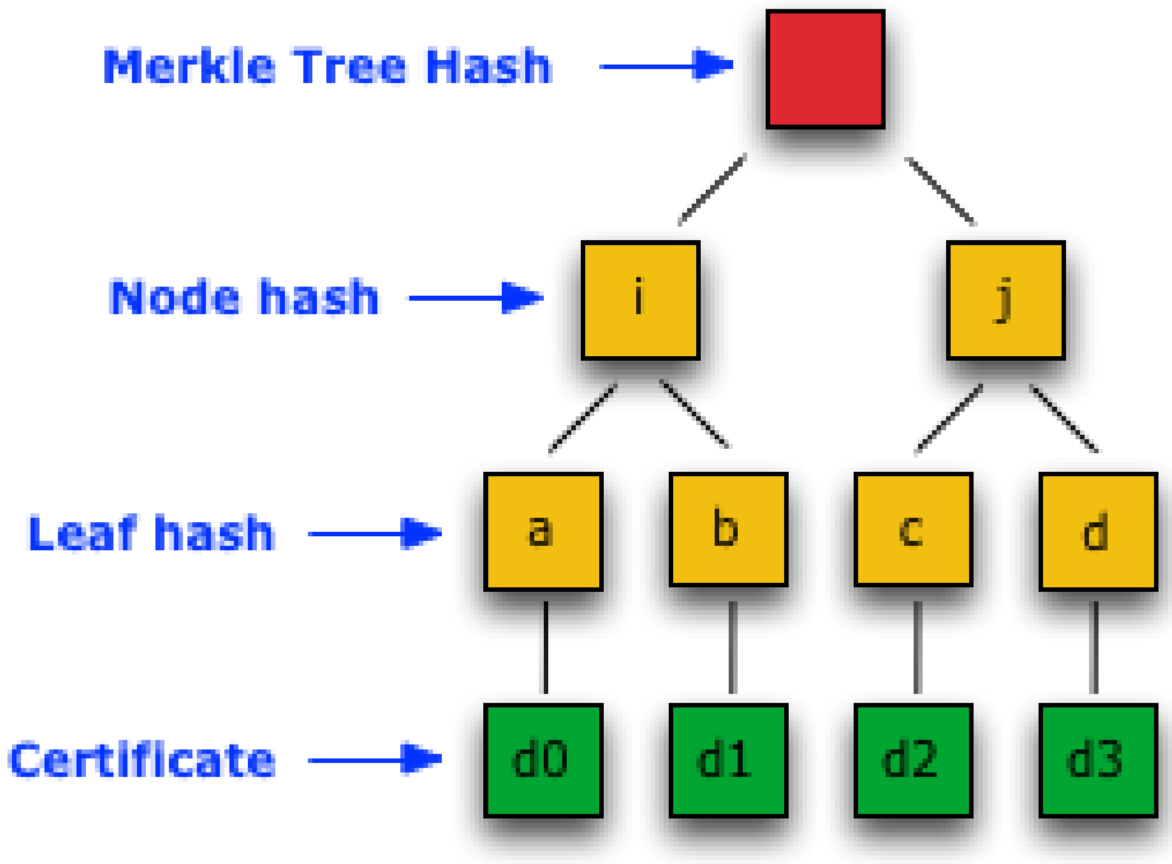


Figure 1

Append: Add d4, d5

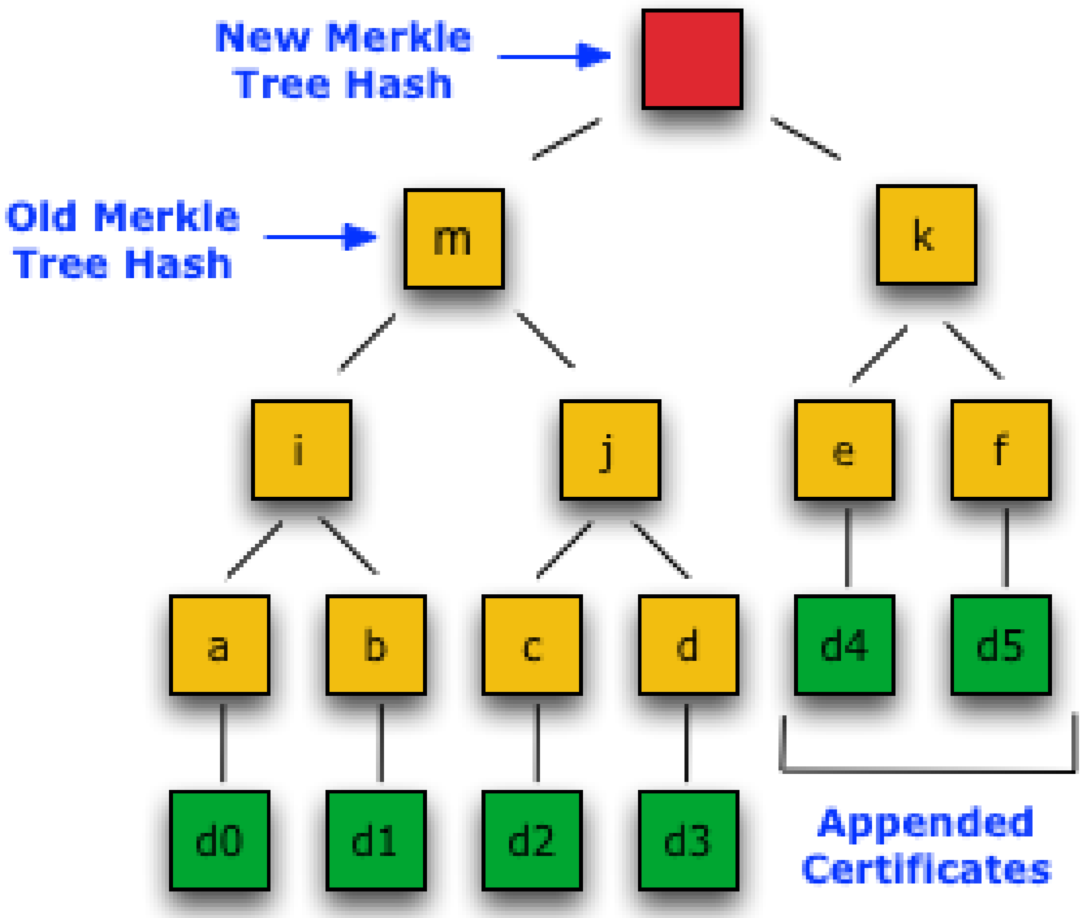


Figure 2

Merkle Tree Verification

Need these to verify member ship

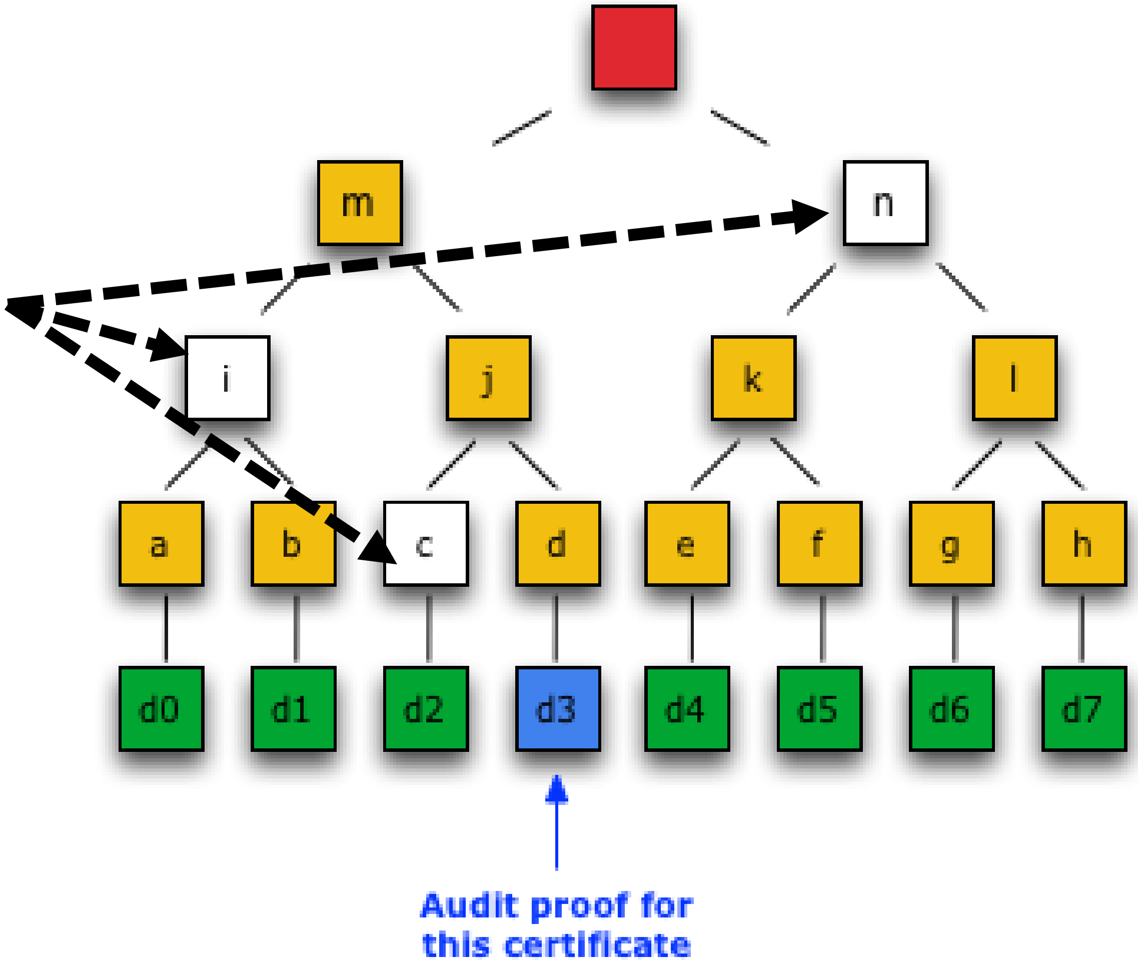
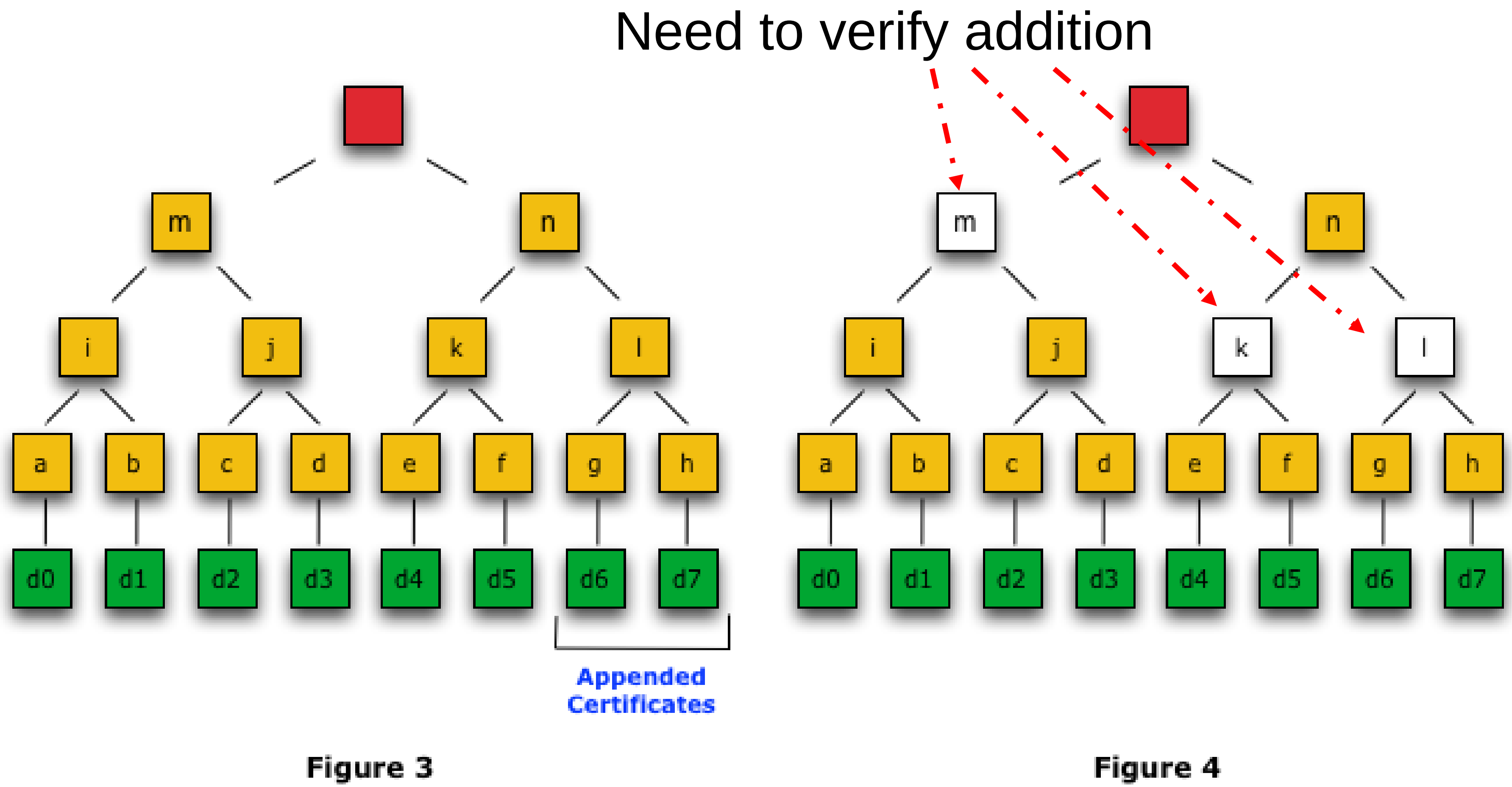


Figure 5

Merkle Tree Consistency



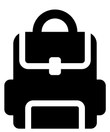
So Far

- Merkle Trees
- Key Establishment: Diffie-Hellman
- Asymmetric Encryption
  - RSA

One-way Functions

- A function is one-way if it's
  - Easy to compute
  - Hard to invert (in the average case)

- Examples
  - Exponentiation vs. Discrete Log
  - Multiplication vs. Factoring
  - Knapsack Packing



    - Given a set of numbers $\{1, 3, 6, 8, 12\}$ find the sum of a subset
    - Given a target sum, find a subset that adds to it

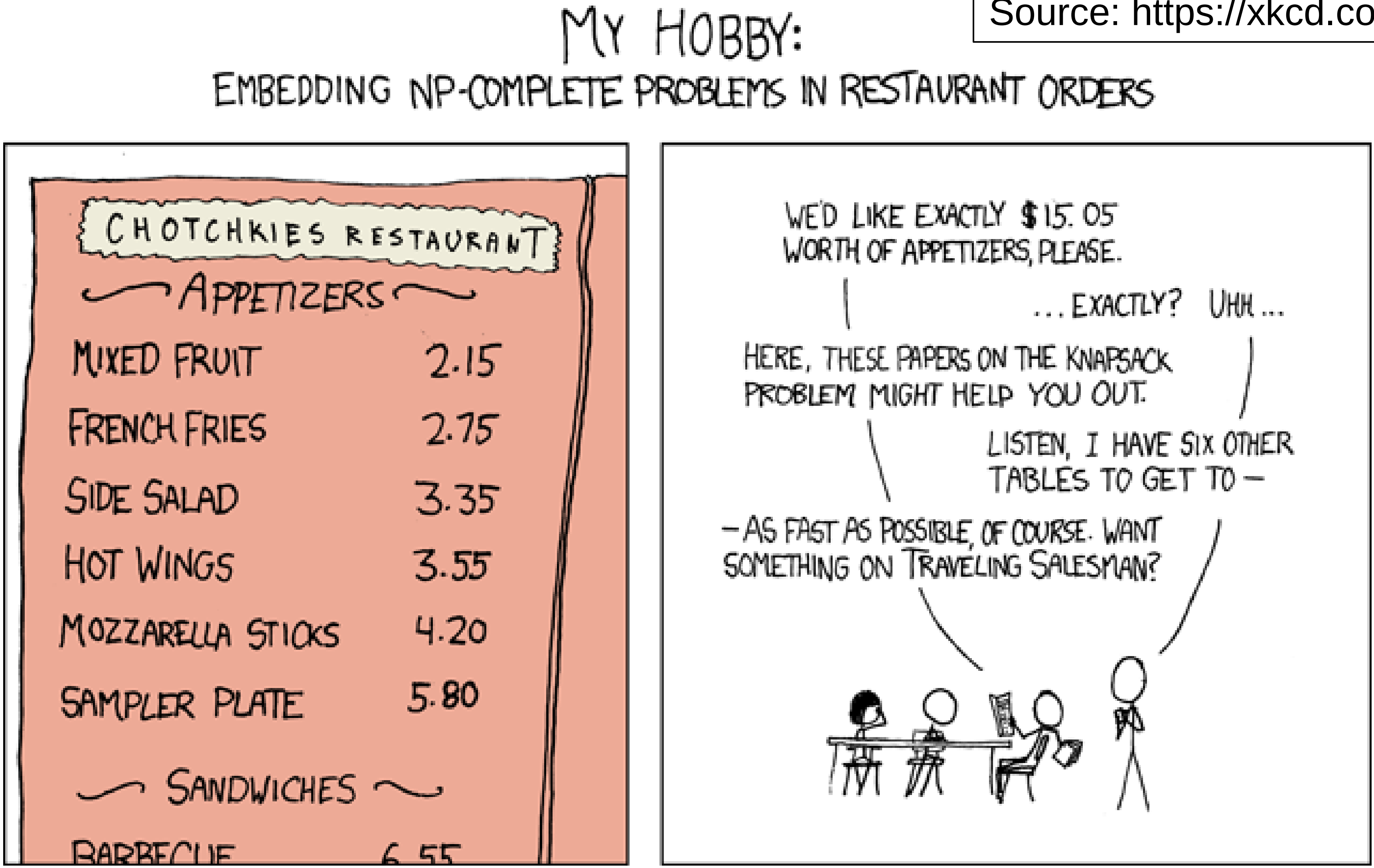
- Trapdoor functions
  - Easy to invert given some extra information
  - e.g. factoring $p \times q$ given q

Trap door

Image source: By Jno. B Jeffery Printing & Engraving, Chicago [Public domain], via Wikimedia Commons

Knapsack Problem

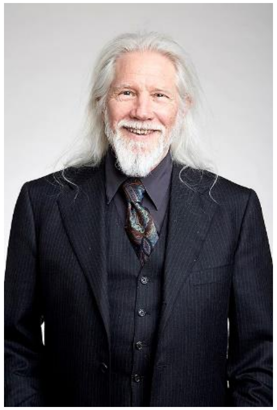
Source: https://xkcd.com/287/



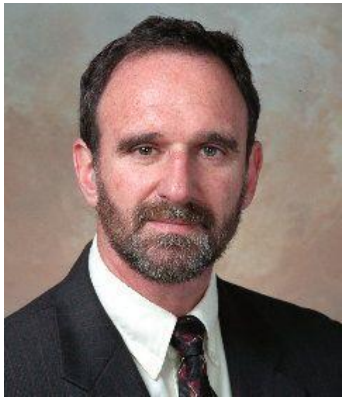
Diffie-Hellman Key Exchange

- Problem with shared-key systems:
  Distributing the shared key
- Suppose that Alice and Bart want to agree on a secret (i.e. a key)
  - Communication link is public
  - They don't already share any secrets

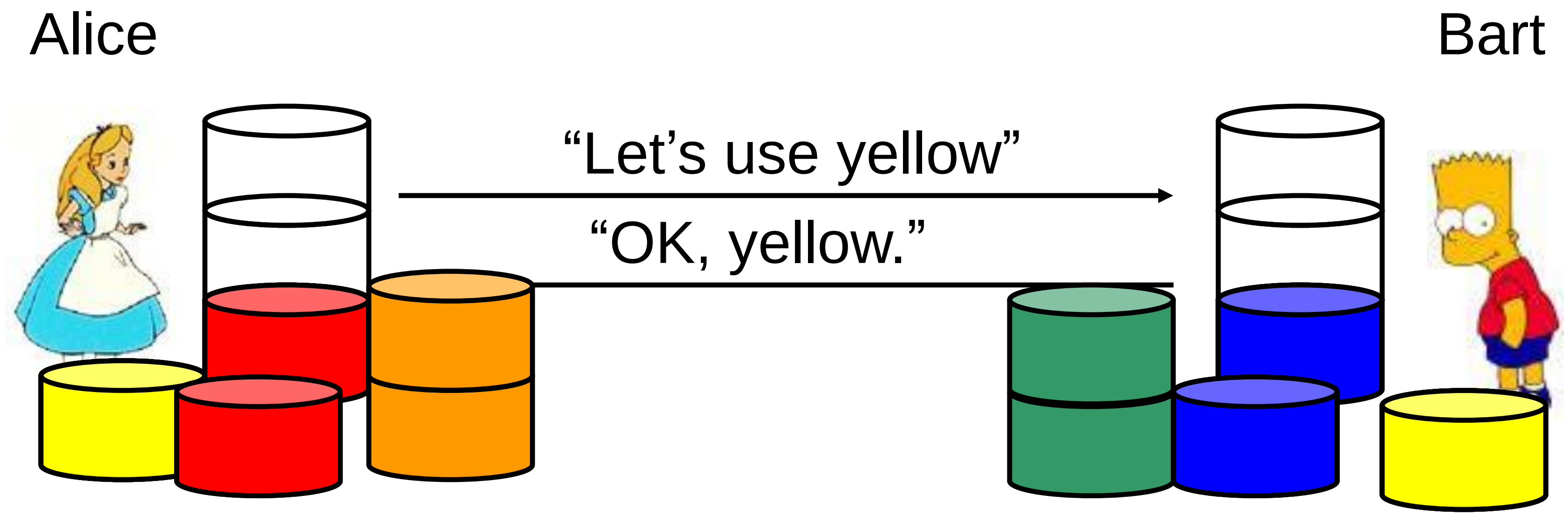
Whitfield Diffie



Martin Hellman

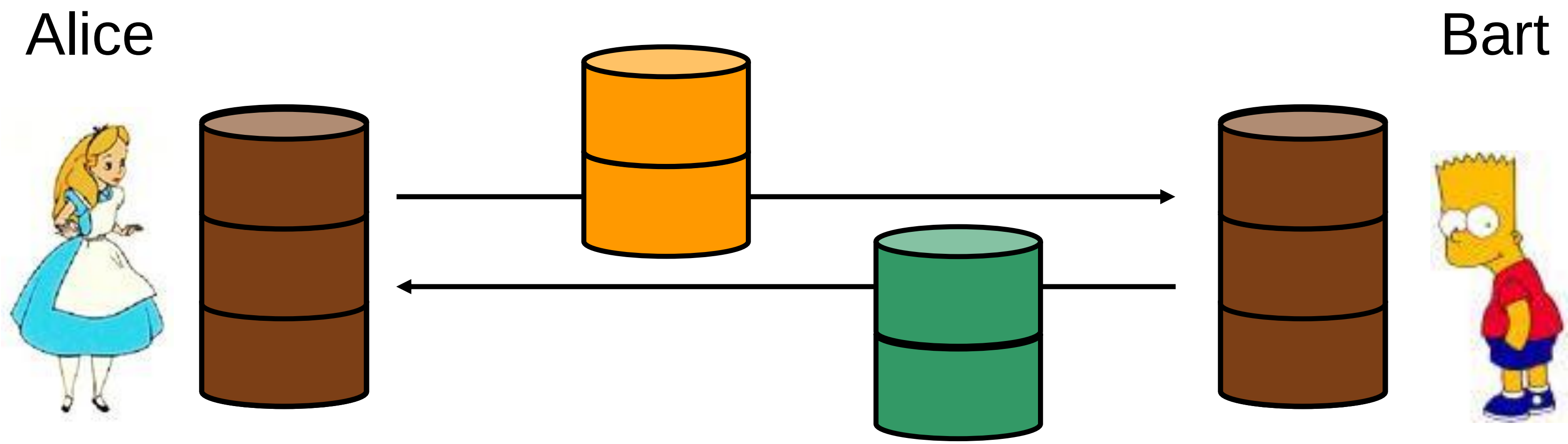


Diffie-Hellman by Analogy: Paint



1. Alice & Bart decide on a public color, and mix one liter of that color.
2. They each choose a random secret color and mix two liters of their secret color
3. They keep one liter of their secret color and mix the other with the public color

Diffie-Hellman by Analogy: Paint

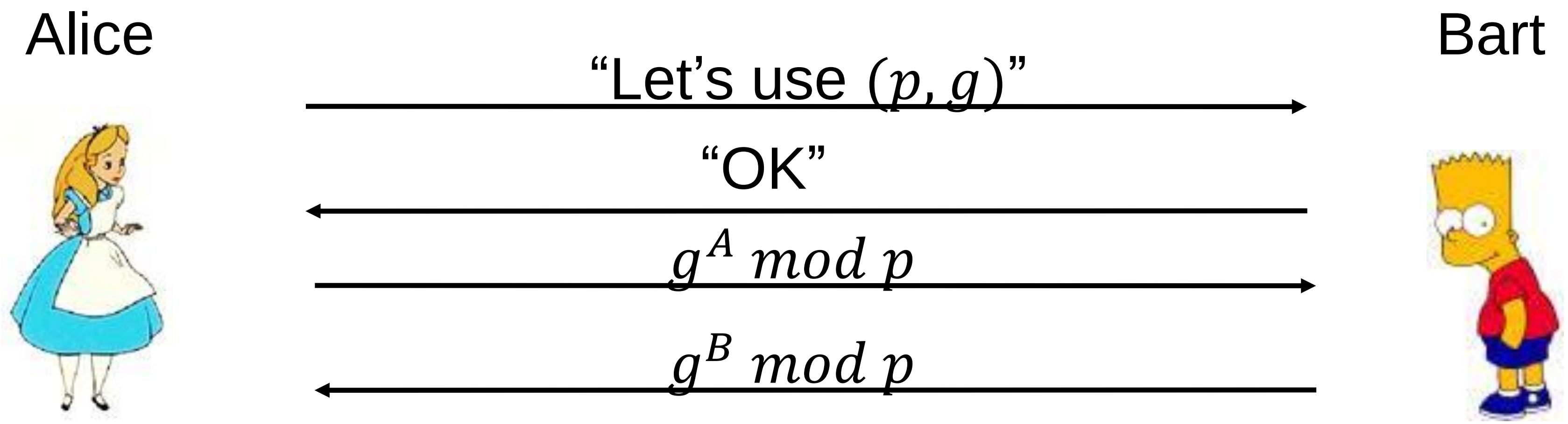


4. They exchange the mixtures over the public channel.
5. When they get the other person's mixture, they combine it with their retained secret color.
6. The secret is the resulting color: Public + Alice's + Bart's

Diffie-Hellman Key Exchange

- Choose a prime p (publicly known)
  - Should be about 512 bits or more
- Pick $g < p$ (also public)
  - g must be a *primitive root* of p.
  - A primitive root *generates* the finite field p.
  - Every $n \in \{1, 2, \ldots, p-1\}$ can be written as
    $$g^k \; mod \; p$$
  - Example: 2 is a primitive root of 5
  - $2^0 = 1 \quad 2^1 = 2 \quad 2^2 = 4 \quad 2^3 = 3 \quad (mod\ 5)$
  - Intuitively means that it's hard to take logarithms base g because there are many candidates.
- Find Primitive Roots: http://www.bluetulip.org/2014/programs/primitive.html
- Check Primitive Roots: http://www.mathcelebrity.com/primitiveroot.php

Diffie-Hellman



1. Alice & Bart decide on a public prime p and primitive root g.
2. Alice chooses secret number A. Bart chooses secret number B
3. Alice sends Bart $g^A \ mod \ p$.
4. The shared secret is $g^{AB} \ mod \ p$.

Details of Diffie-Hellman

- Alice computes $g^{AB}\ mod\ p$ because she knows A:
  - $g^{AB}\ mod\ p\ =\ (g^{B}\ mod\ p)^{A}\ mod\ p$

- An eavesdropper gets $g^{A}\ mod\ p$ and $g^{B}\ mod\ p$
  - They can easily calculate $g^{A+B}\ mod\ p$ but that doesn't help.
  - The problem of computing discrete logarithms (to recover A from $g^{A}\ mod\ p$) is hard.

Example

Alice

1. Agrees $p = 71$ and $g = 7$
2. Selects private key $A = 5$ and calculates public key $g^A = 7^5 = 51 \ (mod\ 71)$.
3. Sends 51 to Bart.
4. Calculates shared secret:
   $S = (g^B)^A = 4^5 = 30 \ (mod\ 71)$

Bart

1. Agrees $p = 71$ and $g = 7$
2. Selects private key $B = 12$ and calculates public key $g^B = 7^{12} = 4 \ (mod\ 71)$.
3. Sends 4 to Alice.
4. Calculates shared secret:
   $S = (g^A)^B = 51^{12} = 30 \ (mod\ 71)$

Why Does it Work?

- Security: the difficulty of calculating discrete logarithms.
- Feasibility:
  - The ability to find large primes
  - The ability to find primitive roots for large primes.
  - The ability to do efficient modular arithmetic.
- Correctness: an immediate consequence of basic facts about modular arithmetic.

- After 2019, NIST requires $p = 15360\ bits$ and $q\ = 512$ bits
- Do not use short or example primes from standards

Further with Diffie-Hellman (DH)

Problem: Exponentiation requires large primes and modulus

Solution: Elliptical Curve based discrete logarithm (ECDH)

More: https://www.youtube.com/watch?v=zTt4gvuQ6sY

Ephemeral DH

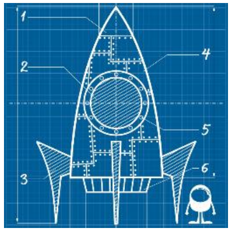
- Long term key used to set up communication
- Switch to a new key by using Diffie-Hellman
- Provides forward secrecy

DH Certificates

- Server chooses p, g, S (secret)
- Publishes $p, g, g^S \bmod p$ in a *digital certificate*
- Client who wants to communicate downloads certificate and starts communicating by completing DH exchange

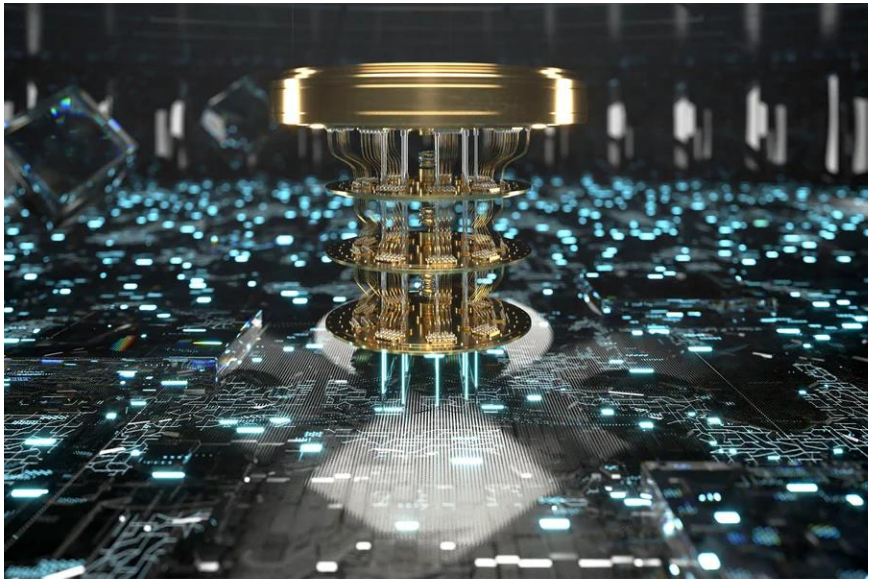
Extended Triple DH (X3DH)

- Regular DH doesn't authenticate
- X3DH gives bilateral authentication
- Much more complex!
- Uses ECDH + servers



What about quantum computers?

- Quantum computing can calculate discrete log and elliptical curve Diffie-Hellman much faster than regular computers



https://www.alamy.com/quantum-computer-with-electrical-circuits-in-the-chamber-image444111122.html

- So…?

So Far

- Merkle Trees
- Key Establishment: Diffie-Hellman
- Asymmetric Encryption
  - RSA

Public Key Cryptography

Asymmetric Cryptography

- Sender encrypts using a *public key*
- Receiver decrypts using a *private key*

Private key must be kept secret

Public key can be distributed to all

Example systems:

- RSA
- El Gamal
- DSA
- Various algorithms based on elliptic curves

Used in:

- SSL
- ssh
- PGP
- Digital signatures

Public Key Notation

Encryption algorithm

- $E : Pub_k \times plaintext \rightarrow ciphertex$
- Notation: $Pub_K\{msg\} = E(Pub_k, msg)$

Decryption algorithm

- $D : Priv_k \times ciphertext \rightarrow plaintext$
- Notation: $Priv_k\{msg\} = D(Priv_k, msg)$

- D inverts E

$$D(Priv_k, E(Pub_K, msg)) = msg$$

- Use "Pub_K" for public keys
- Use "$Priv_k$" for private keys

- In some systems, E is the same algorithm as D

Secure Channel: Private Key

Alice

$Pub_{K_B}\{Hello!\}$

$Pub_{K_A}\{Hi!\}$

Bart

Pub_{K_A}, Pub_{K_B}
$Priv_{K_A}$

Pub_{K_A}, Pub_{K_B}
$Priv_{K_B}$

Trade-offs for Public Key Crypto

Positive

1. A principal needs one private key and one public key

2. Number of keys for pair-wise communication is $O(n)$

Negative

More formal justification of difficulty

- Hardness based on complexity-theoretic results

More computationally expensive than shared key crypto

- Algorithms are harder to implement
- Require more complex machinery

RSA Algorithm

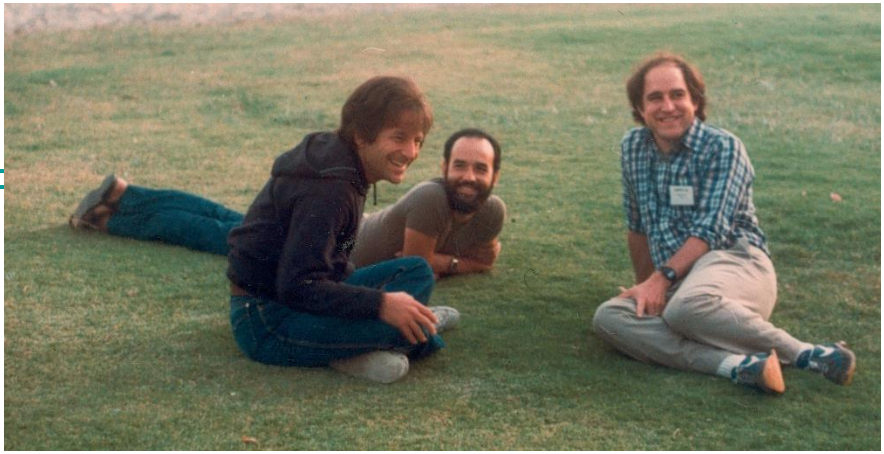


Image source: http://people.csail.mit.edu/rivest/photos/Len-Adi-Ron.jpg

- Ron Rivest, Adi Shamir, Leonard Adleman
  - Proposed in 1979
  - Won the 2002 Turing award for it

- Has withstood years of cryptanalysis
  - Not a guarantee of security!
  - But a strong vote of confidence.

- Hardware implementations: 1000 x slower than DES

RSA at a High Level

Public and private key are derived from secret prime numbers

- Keys are typically $\geq 1024\ bits$
- After 2013, NIST requires $> 2048\ bits$ (equivalent to 112 bits symmetric)

Plaintext message - sequence of bytes - treated as a (large!) binary number

Encryption is modular exponentiation

To break the encryption, conjectured that one must be able to factor large numbers

- Not known to be in P (polynomial time algorithms)

Equivalent Key Lengths

Bits of Security	Symmetric Key Algorithm	Diffie-Hellman (p,g)	RSA (n)	Elliptical Curve (ECDH)
112	Triple-Des	(2048, 224)	2048	224-225
128	AES-128	(3072, 256)	3072	256-383
192	AES-192	(7680, 384)	7680	384-511
256	AES-256	(15,360, 512)	15,360	512+

All lengths in bits

Source: Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program, updated Oct 2023

Algorithm Security Lifetime

Bits of Security	**Symmetric Key Algorithm**	**Diffie-Hellman (p,g)**	**RSA (n)**	**Elliptical Curve (ECDH)**
Through 2030 (112 bits)	Triple-Des AES-128 AES-192 AES-256	Minimum (2048, 224)	Minimum 2048	Minimum 224
Beyond 2030 (128+ bits)	AES-128 AES-192 AES-256	Minimum (3072, 256)	Minimum 3072	Minimum 256

All lengths in bits

Source: Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program, updated Oct 2023

RSA Encryption and Decryption

- Message: m
- Assume $m < n$
  - If not, break up message into smaller chunks
  - Good choice: largest power of 2 smaller than n

- Encryption: $E((e,n),m) = m^e \, mod \, n$
- Decryption: $D((d,n),c) = c^d \, mod \, n$

Conclusion

- Merkle Trees
- Key Establishment: Diffie-Hellman
- Asymmetric Encryption
  - RSA