CS162
Operating Systems and Systems Programming
Lecture 25

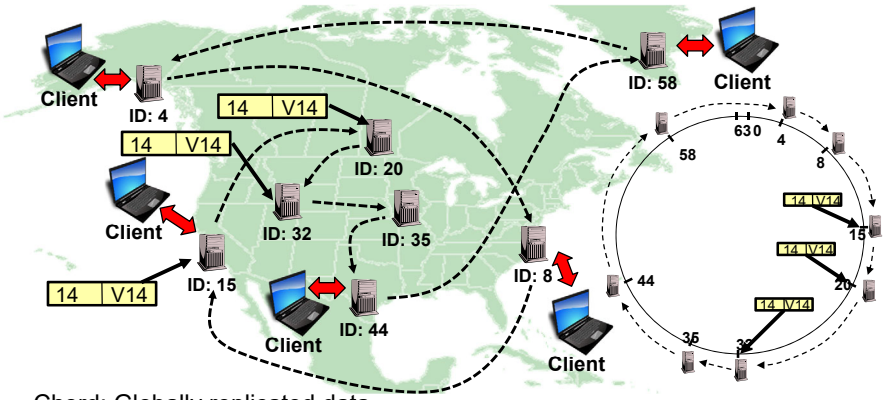
Security, Edge Computing
Quantum Computing

May 7th, 2019

Prof. John Kubiatowicz

http://cs162.eecs.Berkeley.edu

Recall: Chord Replication in Physical Space



- Chord: Globally replicated data
  - But – Is it secure?
  - Resilient to Denial of Service?
- Replicating in Adjacent nodes of virtual space ⇒ Geographic Separation in physical space
  - Avoids single-points of failure through randomness
  - More nodes, more replication, more geographic spread
  - But – Are all the copies identical and authentic???

What is Computer Security Today?

- Computing in the presence of an adversary!
  - Adversary is the security field's defining characteristic
- Reliability, robustness, and fault tolerance
  - Dealing with Mother Nature (random failures)
- Security
  - Dealing with actions of a knowledgeable attacker dedicated to causing harm
  - Surviving malice, and not just mischance
- Wherever there is an adversary, there is a computer security problem!

CIMPLICITY®
BlackEnergy SCADA malware
(Supervisory Control and Data Acquisition)

Mirai IoT botnet

On The Importance of Data Integrity

- In July (2015), a team of researchers took total control of a Jeep SUV remotely
- They exploited a firmware update vulnerability and hijacked the vehicle over the Sprint cellular network
- They could make it speed up, slow down and even veer off the road

- Machine-to-Machine (M2M) communication has reached a dangerous tipping point
  - Cyber Physical Systems use models and behaviors that from elsewhere
  - Firmware, safety protocols, navigation systems, recommendations, …
  - IoT (whatever it is) is everywhere
- Do you know where your data came from? *PROVENANCE*
- Do you know that it is ordered properly? *INTEGRITY*
- *The rise of Fake Data!*
  - *Much worse than Fake News…*
  - *Corrupt the data, make the system behave very badly*

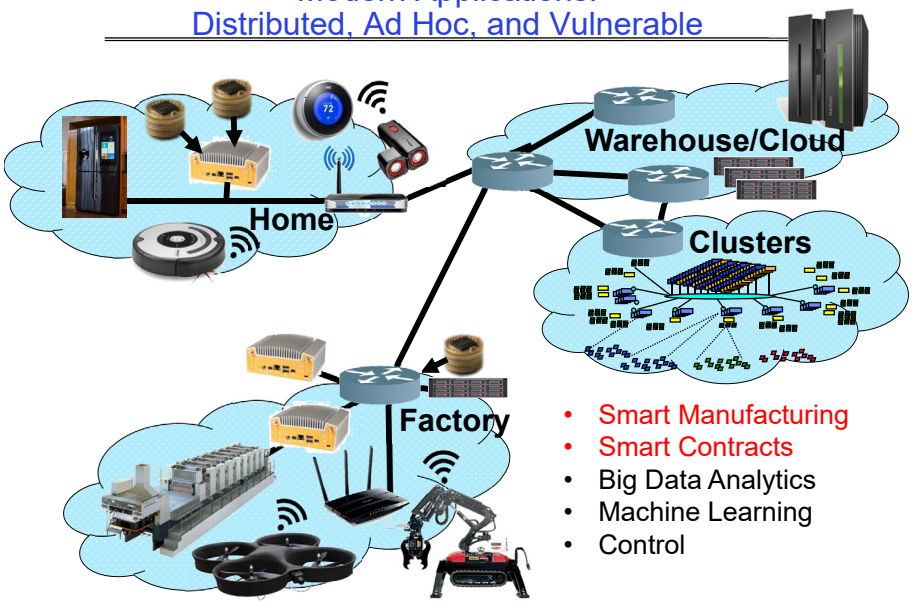
Protection vs. Security

- Protection: mechanisms for controlling access of programs, processes, or users to resources
  - Page table mechanism
  - Round-robin schedule
  - Data encryption
- Security: use of protection mechanisms to prevent misuse of resources
  - Misuse defined with respect to policy
    - » E.g.: prevent exposure of certain sensitive information
    - » E.g.: prevent unauthorized modification/deletion of data
  - Need to consider external operational environment
    - » Most well-constructed system cannot protect information if user accidentally reveals password – social engineering challenge

Security Requirements

- Authentication
  - Ensures that a user is who is claiming to be
- Data integrity
  - Ensure that data is not changed from source to destination or after being written on a storage device
- Confidentiality
  - Ensures that data is read only by authorized users
- Non-repudiation
  - Sender/client can't later claim didn't send/write data
  - Receiver/server can't claim didn't receive/write data

Modern Applications: Distributed, Ad Hoc, and Vulnerable



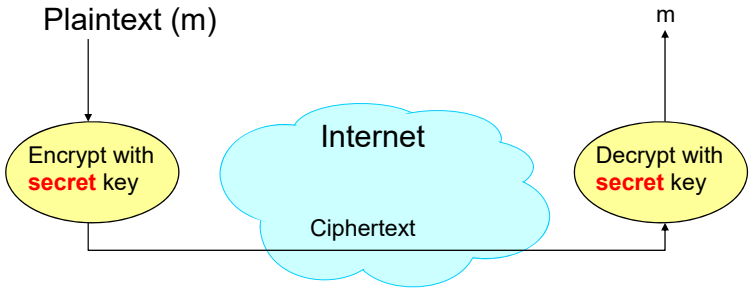
- Smart Manufacturing
- Smart Contracts
- Big Data Analytics
- Machine Learning
- Control

Start Here: Securing Communication via Cryptography

- Cryptography: communication in the presence of adversaries
- Studied for thousands of years
  - See the Simon Singh's The Code Book for an excellent, highly readable history
- Central goal: confidentiality
  - How to encode information so that an adversary can't extract it, but a friend can
- General premise: there is a key, possession of which allows decoding, but without which decoding is infeasible
  - Thus, key must be kept secret and not guessable

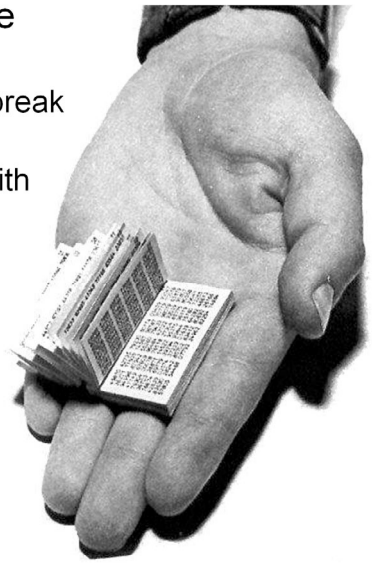
Basic Tool: Using Symmetric Keys

- Same key for encryption and decryption
- Achieves confidentiality
- Vulnerable to tampering and replay attacks



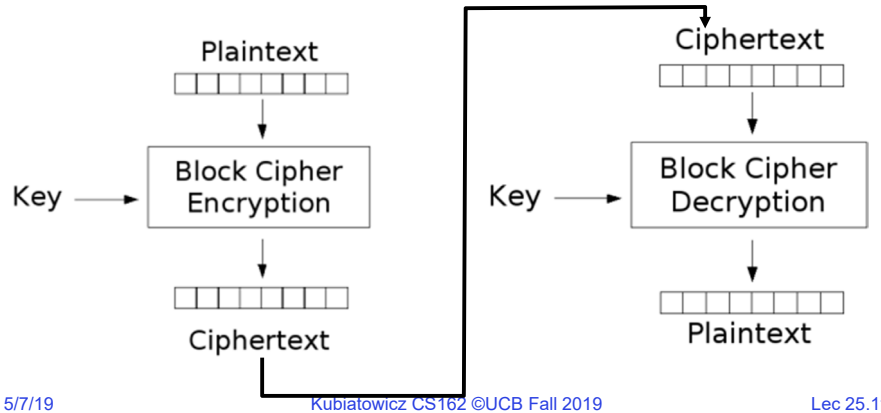
Symmetric Keys

- Can just XOR plaintext with the key
  - Easy to implement, but easy to break using frequency analysis
  - Unbreakable alternative: XOR with one-time pad
    - » Use a different key for each message



Block Ciphers with Symmetric Keys

- More sophisticated (e.g., block cipher) algorithms
  - Works with a block size (e.g., 64 bits)
- Can encrypt blocks separately:
  - Same plaintext⇒same ciphertext
- Much better:
  - Add in counter and/or link ciphertext of previous block



Symmetric Key Ciphers - DES & AES

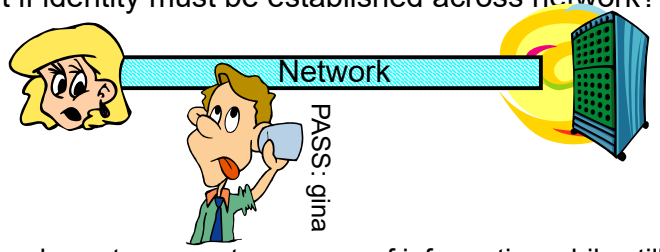
- Data Encryption Standard (DES)
  - Developed by IBM in 1970s, standardized by NBS/NIST
  - 56-bit key (decreased from 64 bits at NSA's request)
  - Still fairly strong other than brute-forcing the key space
    - » But custom hardware can crack a key in < 24 hours
  - Today many financial institutions use Triple DES
    - » DES applied 3 times, with 3 keys totaling 168 bits
- Advanced Encryption Standard (AES)
  - Replacement for DES standardized in 2002
  - Key size: 128, 192 or 256 bits
- How fundamentally strong are they?
  - No one knows (no proofs exist)

Why are Data Breaches so Frequent?

- State of the art: AdHoc boundary construction!
  - Protection mechanisms are "roll-your-own" and different for each application
  - Use of encrypted channels to "tunnel" across untrusted domains
- Data is typically protected at the Border rather than Inherently
  - Large Trusted Computing Base (TCB): huge amount of code must be correct to protect data
  - Make it through the border (firewall, OS, VM, container, etc…) data compromised!
- What about data integrity and provenance?
  - Any bits inserted into "secure" environment get trusted as authentic ⇒ manufacturing faults or human injury or exposure of sensitive information

Authentication in Distributed Systems

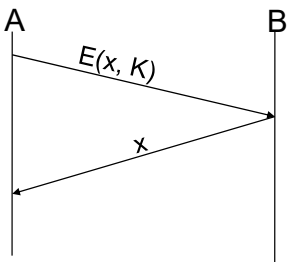
- What if identity must be established across network?



  - Need way to prevent exposure of information while still proving identity to remote system
  - Many of the original UNIX tools sent passwords over the wire "in clear text"
    - » E.g.: telnet, ftp, yp (yellow pages, for distributed login)
    - » Result: Snooping programs widespread
- What do we need? Cannot rely on physical security!
  - Encryption: Privacy, restrict receivers
  - Authentication: Remote Authenticity, restrict senders

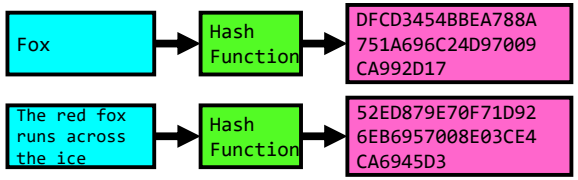
Authentication via Secret Key

- Main idea: entity proves identity by decrypting a secret encrypted with its own key
  - K – secret key shared only by A and B
- A can asks B to authenticate itself by decrypting a nonce, i.e., random value, x
  - Avoid replay attacks (attacker impersonating client or server)
- Vulnerable to man-in-the middle attack



Notation: E(m,k) – encrypt message m with key k

Basic Tool: Secure Hash Function

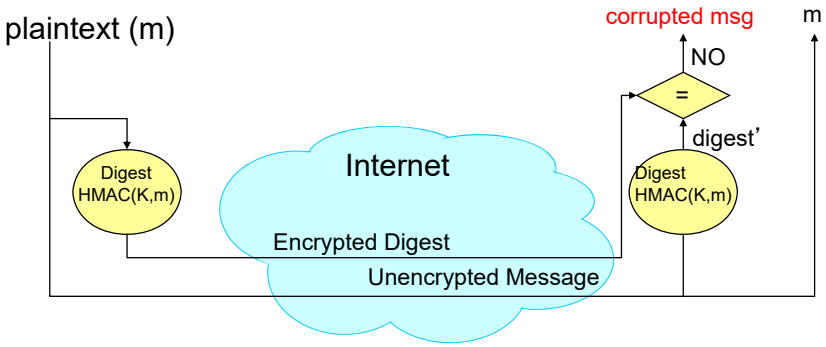


- Hash Function: Short summary of data (message)
  - For instance, $h_1=H(M_1)$ is the hash of message M_1
    - » h_1 fixed length, despite size of message M_1
    - » Often, h_1 is called the "digest" of M_1
- Hash function H is considered secure if
  - It is infeasible to find M_2 with $h_1=H(M_2)$; i.e., can't easily find other message with same digest as given message
  - It is infeasible to locate two messages, m_1 and m_2, which "collide", i.e. for which $H(m_1) = H(m_2)$
  - A small change in a message changes many bits of digest/can't tell anything about message given its hash
- Best Current Example: SHA-2 (2001)
  - Family of SHA-224, SHA-256, SHA-384, SHA-512 functions

Integrity: Cryptographic Hashes

- Basic building block for integrity: cryptographic hashing
  - Associate hash with byte-stream, receiver verifies match
    - » Assures data hasn't been modified, either accidentally – or maliciously
- Approach:
  - Sender computes a secure digest of message m using H(x)
    - » H(x) is a publicly known hash function
    - » Digest d = HMAC (K, m) = H (K | H (K | m))
    - » HMAC(K, m) is a hash-based message authentication function
  - Send digest d and message m to receiver
  - Upon receiving m and d, receiver uses shared secret key, K, to recompute HMAC(K, m) and see whether result agrees with d
- Another use of Hashes: A fixed-length name for data
  - Instead of asking for data, ask for hash!
  - Hashes can serve as routing addresses

Using Hashing for Integrity



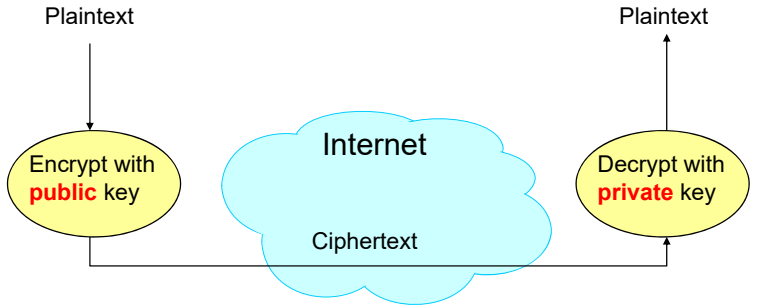
Can encrypt m for confidentiality

Asymmetric Encryption (Public Key)

- Idea: use two different keys, one to encrypt (e) and one to decrypt (d)
  - A key pair
- Crucial property: knowing e does not give away d
- Therefore e can be public: everyone knows it!
- If Alice wants to send to Bob, she fetches Bob's public key (say from Bob's home page) and encrypts with it
  - Alice can't decrypt what she's sending to Bob …
  - … but then, neither can anyone else (except Bob)

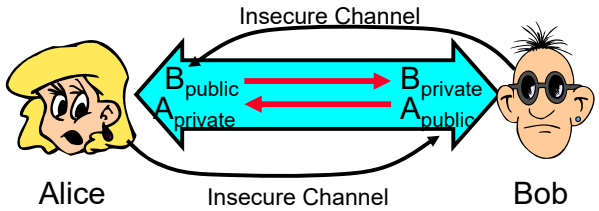
Basic Tool: Public Key / Asymmetric Encryption

- Sender uses receiver's public key
  - Advertised to everyone
- Receiver uses complementary private key
  - Must be kept secret



Public Key Encryption Details

- Idea: K_{public} can be made public, keep $K_{private}$ private



- Gives message privacy (restricted receiver):
  - Public keys (secure destination points) can be acquired by anyone/used by anyone
  - Only person with private key can decrypt message
- What about authentication?
  - Use combination of private and public key
  - Alice→Bob: [(I'm Alice)^{Aprivate} Rest of message]^{Bpublic}
  - Provides restricted sender and receiver
- But: how does Alice know that it was Bob who sent her B_{public}? And vice versa… Story for another time!

Public Key Cryptography

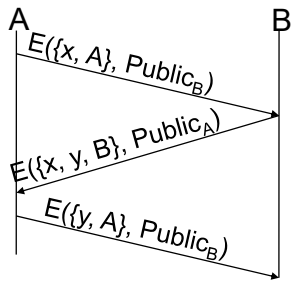
- Invented in the 1970s
  - Revolutionized cryptography
  - (Was actually invented earlier by British intelligence)
- How can we construct an encryption/decryption algorithm using a key pair with the public/private properties?
  - Answer: Number Theory
- Most fully developed approach: RSA
  - Rivest / Shamir / Adleman, 1977; RFC 3447
  - Based on modular multiplication of very large integers
  - Very widely used (e.g., ssh, SSL/TLS for https)
- Also mature approach: Eliptic Curve Cryptography (ECC)
  - Based on curves in a Galois-field space
  - Shorter keys and signatures than RSA

Properties of RSA

- Requires generating large, random prime numbers
  - Algorithms exist for quickly finding these (probabilistic!)
- Requires exponentiation of very large numbers
  - Again, fairly fast algorithms exist
- Overall, much slower than symmetric key crypto
  - One general strategy: use public key crypto to exchange a (short) symmetric session key
    - » Use that key then with AES or such
- How difficult is recovering d, the private key?
  - Equivalent to finding prime factors of a large number
    - » Many have tried - believed to be very hard (= brute force only)
    - » (Though quantum computers could do so in polynomial time!)

Simple Public Key Authentication

- Each side need only to know the other side's public key
  - No secret key need be shared
- A encrypts a nonce (random num.) x
  - Avoid replay attacks, e.g., attacker impersonating client or server
- B proves it can recover x, generates second nonce y
- A can authenticate itself to B in the same way
- A and B have shared private secrets on which to build private key!
  - We just did secure key distribution!
- Many more details to make this work securely in practice!

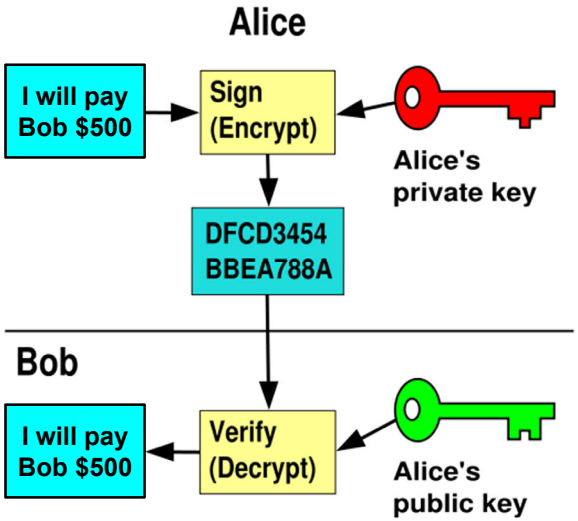


Notation: E(m,k) – encrypt message m with key k

Non-Repudiation: RSA Crypto & Signatures

- Suppose Alice has published public key K_E
- If she wishes to prove who she is, she can send a message x encrypted with her private key K_D (i.e., she sends $E(x, K_D)$)
  - Anyone knowing Alice's public key K_E can recover x, verify that Alice must have sent the message
    - » It provides a signature
  - Alice can't deny it: non-repudiation
- Could simply encrypt a hash of the data to sign a document that you wanted to be in clear text
- Note that either of these signature techniques work perfectly well with any data (not just messages)
  - Could sign every datum in a database, for instance

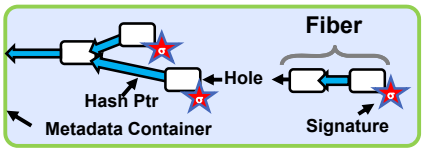
Public Key Crypto & Signatures



Digital Certificates

- How do you know K_E is Alice's public key?
- Trusted authority (e.g., Verisign) signs binding between Alice and K_E with its private key $KV_{private}$
  - $C = E(\{Alice, K_E\}, KV_{private})$
  - C: digital certificate
- Alice: distribute her digital certificate, C
- Anyone: use trusted authority's KV_{public}, to extract Alice's public key from C
  - $D(C, KV_{public}) = D(E(\{Alice, K_E\}, KV_{private}), KV_{public}) = \{Alice, K_E\}$

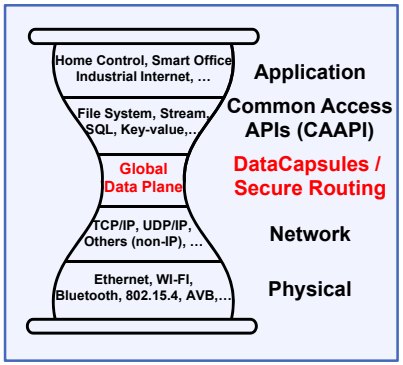
The Data-Centric Vision: Cryptographically Hardened Data Containers



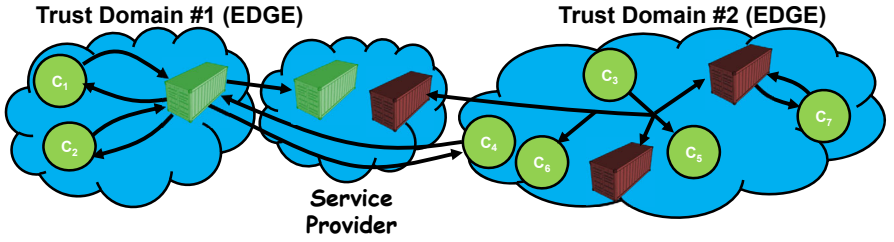
- Inspiration: Shipping Containers
  - Invented in 1956. Changed everything!
  - Ships, trains, trucks, cranes handle standardized format containers
  - Each container has a unique ID
  - Can ship (and store) anything
- Can we use this idea to help security of our systems?
  - Want Unique Name
  - Want Universal Transport
  - Want to Hold Anything
  - Want Locking (integrity of contents)
  - Want Privacy
- DataCapsule (DC):
  - Standardized metadata wrapped around opaque data transactions
  - Uniquely named (via HASH) and globally findable
  - Every transaction explicitly sequenced in a hash-chain history
  - Provenance enforced through signatures
  - All Data Encrypted Except When in Use
- Underlying infrastructure assists and improves performance
  - Anyone can verify validity, membership, and sequencing of transactions (like blockchain)

Refactoring of Applications around Security, Integrity, and Provenance of Information

- Goal: A thin Standardized entity that can be easily adopted and have immediate impact
  - Can be embedded in edge environments
  - Can be exploited in the cloud
  - Natural adjunct to Secure Enclaves for computation
- DataCapsules ⇒ bottom-half of a blockchain?
  - Or a GIT-style version history
  - Simplest mode: a secure log of information
  - Universal unique name ⇒ permanent reference
- Applications writers think in terms of traditional storage access patterns:
  - File Systems, Data Bases, Key-Value stores
  - Called Common Access APIs (CAAPIs)
  - DataCapsules are always the Ground Truth



Global Data Plane (GDP) and the Secure Datagram Routing Protocol



- Flat Address Space Routing
  - Route queries to DCs by names, independent of location (e.g. no IP)
  - Example: use Chord to map names to locations!
  - DCs move, network deals with it
- Black Hole Elimination
  - Only servers authorized by owner of DC may advertise DC service
- Routing only through domains you trust!
  - Secure Delegated Flat Address Routing
- Secure Multicast Protocol
  - Only clients/DC storage servers with proper (delegation) certificates may join
- Queries (messages) are Fibers
  - Self-verifying chunks of DataCapsules
  - Writes include appropriate credentials
  - Reads include proofs of membership

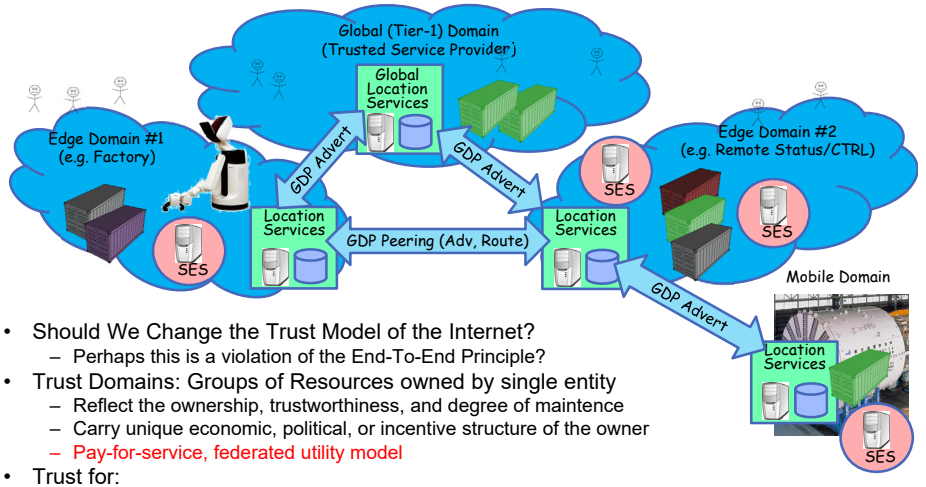
Incremental deployment as an overlay

- Prototype tunneling protocol ("GDPinUDP")
- Federated infrastructure w/routing certificates

Why the Global Data Plane (GDP) ?

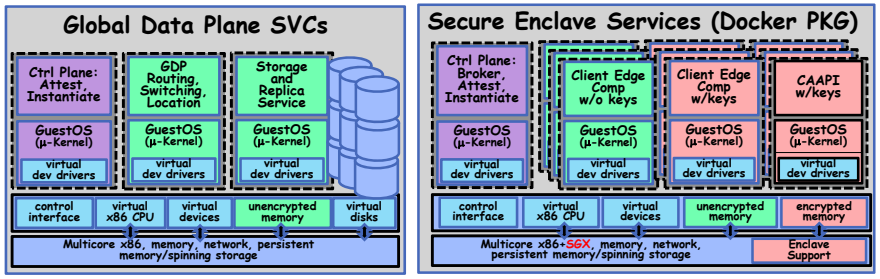
- Yes, you *could:*
  - *Provide your own infrastructure for everything*
  - *Provide your own storage servers*
  - *Provide your own networking, location resolvers, intermediate rendezvous points*
- But: *Why?*
  - Standardization is what made the IP infrastructure so powerful
  - Utilize 3rd-party infrastructure owned (and constantly improved) by others
  - Sharing is much harder with stovepiped solutions!
- The Global Data Plane provides *standardized infrastructure support*
  - It provides a standardized substrate for secure flat routing and publish-subscribe multicast
  - It provides a provides the ability to reason about infrastructure providers (Trust Domains)
  - It frees DataCapsules from being tied to a particular physical location
  - ⇒ Analogous to ships, planes, trains, and cranes that support shipping containers
- The GDP routes conversations between endpoints such as DataCapsules, sensors, actuators, services, clients, etc.
- Information protected in DataCapsules, but freed from physical limitations by the GDP
  - Correctness and Provenance *enforced* by DataCapsules
  - Performance, QoS, and Delegation of Trust handled by the GDP

Reasoning about the infrastructure: Trust Domains



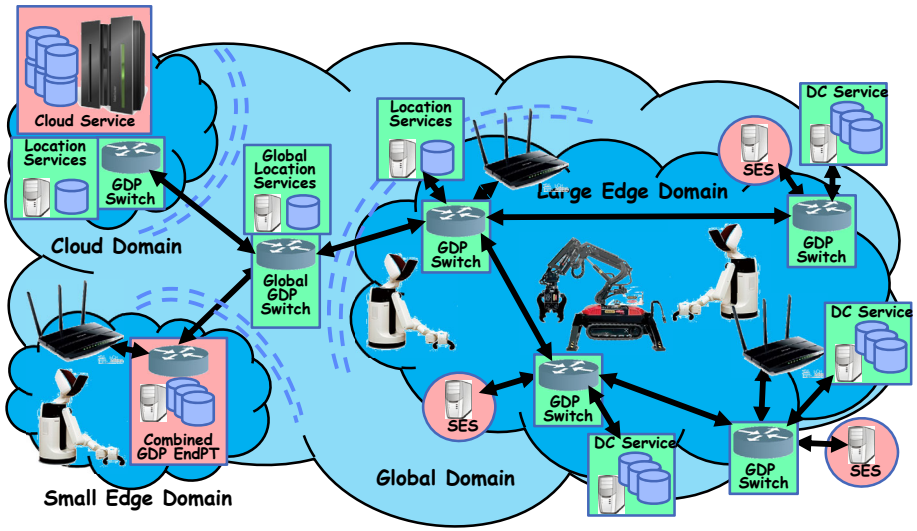
- Should We Change the Trust Model of the Internet?
  - Perhaps this is a violation of the End-To-End Principle?
- Trust Domains: Groups of Resources owned by single entity
  - Reflect the ownership, trustworthiness, and degree of maintence
  - Carry unique economic, political, or incentive structure of the owner
  - Pay-for-service, federated utility model
- Trust for:
  - Message Transport, Location Resolution, DataCapsule Service, Secure Enclave Service (SES)
  - Conversations routed according to DataCapsule owner's Trust Preferences

How to make DataCapsule Vision a Reality?

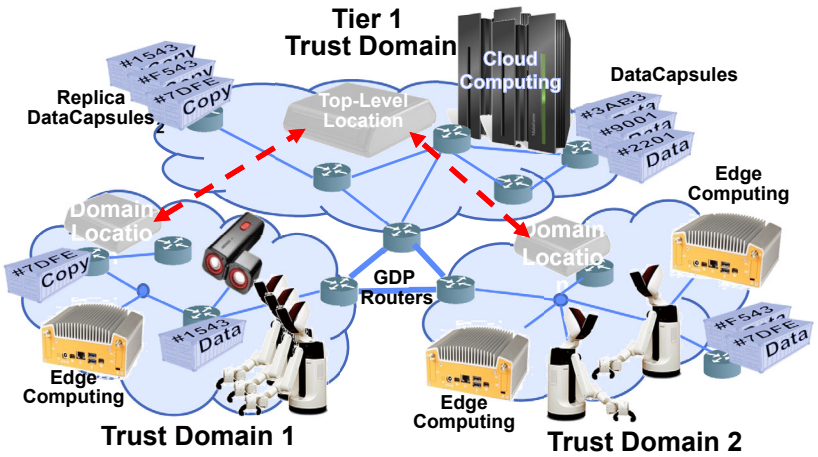


- Active Routing/Switching Components
  - Federated/Utility storage infrastructure
  - Edge-local support for multicast
  - Data Location Services
- Owned by service provider (trust domain)
  - Secure boot/validated code in DataCapsule
  - Multiple providers may own equipment in single physical environment
- Multi-Tenant Secure Computation Services
  - Secure Enclaves on Demand with specified attributes (e.g. GPU, special accelerator, etc.)
  - Standardized packaging (e.g. Docket)
  - Trustable computation through attestation, key exchange, resistance to physical attacks
- Computation is *fungible:*
  - Executable and state stored in DataCapsules!

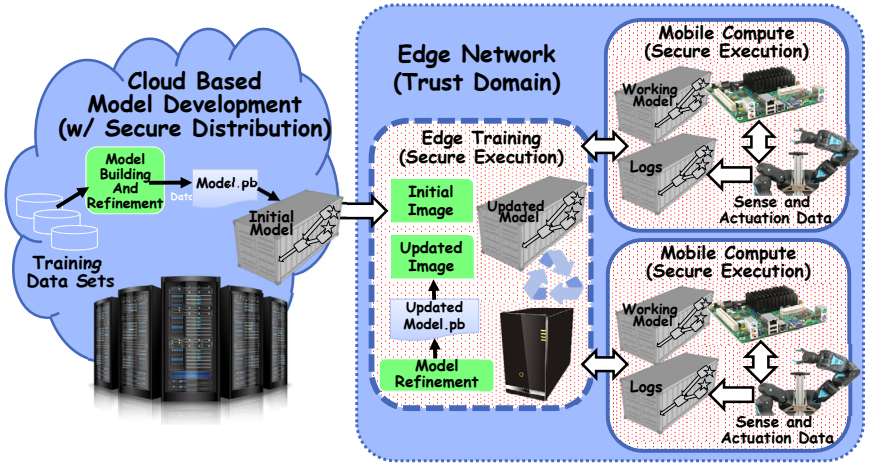
DataCapsule Infrastructure Initially Build Network As an Overlay!



Fog Robotics on the Global Data Plane: SwarmLab/RiseLab/Robotics



Training Models for Robots at the Edge



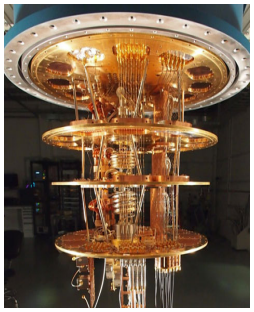
- Proprietary model developed in the cloud
  - Secure distribution to the edge for use!
- Edge Computing Domain makes local updates to models
  - Also secure, doesn't leak private information outside Edge Network

BREAK

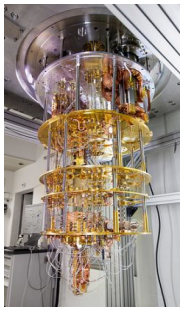
Use Quantum Mechanics to Compute?

- Weird but useful properties of quantum mechanics:
  - Quantization: Only certain values or orbits are good
    - » Remember orbitals from chemistry???
  - Superposition: Schizophrenic physical elements don't quite know whether they are one thing or another
- All existing digital abstractions try to eliminate QM
  - Transistors/Gates designed with classical behavior
  - Binary abstraction: a "1" is a "1" and a "0" is a "0"
- Quantum Computing:
  Use of Quantization and Superposition to compute.
- Interesting results:
  - Shor's algorithm: factors in polynomial time!
  - Grover's algorithm: Finds items in unsorted database in time proportional to square-root of n.
  - Materials simulation: exponential classically, linear-time QM

Current "Arms Race" of Quantum Computing



Google: Superconducting
Devices up 72-qubits

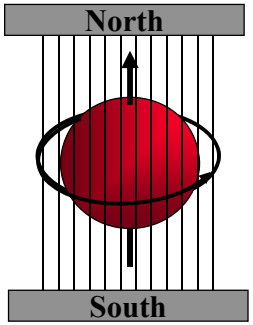


IBM: Superconducting
Devices up to 50 qubits

- Big companies looking at Quantum Computing Seriously
  - Google, IBM, Microsoft
- Current Goal: Quantum Supremacy
  - Show that Quantum Computers faster than Classical ones
  - "If a quantum processor can be operated with low enough error, it would be able to outperform a classical supercomputer on a well-defined computer science problem, an achievement known as quantum supremacy."

Quantization: Use of "Spin"

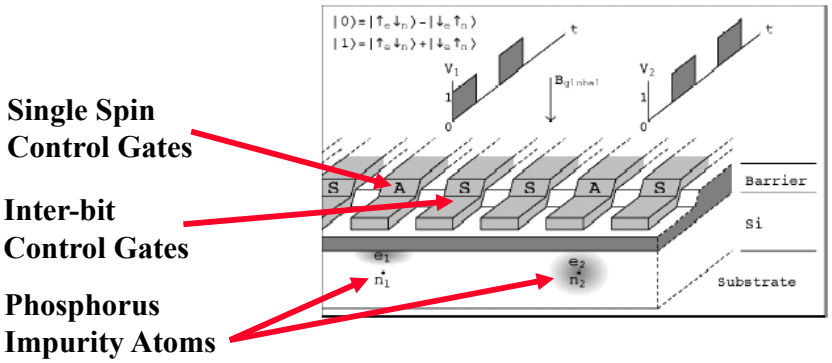
Spin ½ particle:
(Proton/Electron)



Representation:
|0> or |1>

- Particles like Protons have an intrinsic "Spin" when defined with respect to an external magnetic field
- Quantum effect gives "1" and "0":
  - Either spin is "UP" or "DOWN" nothing between

Kane Proposal II
(First one didn't quite work)



- Bits Represented by combination of proton/electron spin
- Operations performed by manipulating control gates
  - Complex sequences of pulses perform NMR-like operations
- Temperature < 1° Kelvin!

Now add Superposition!

- The bit can be in a combination of "1" and "0":
  - Written as: $\Psi = C_0|0> + C_1|1>$
  - The C's are *complex numbers!*
  - Important Constraint: $|C_0|^2 + |C_1|^2 = 1$
- If *measure* bit to see what looks like,
  - With probability $|C_0|^2$ we will find |0> (say "UP")
  - With probability $|C_1|^2$ we will find |1> (say "DOWN")
- Is this a real effect? Options:
  - This is just statistical – given a large number of protons, a fraction of them ($|C_0|^2$) are "UP" and the rest are down.
  - This is a real effect, and the proton is really both things until you try to look at it
- Reality: second choice!
  - There are experiments to prove it!

A register can have many values!

- Implications of superposition:
  - An *n*-bit register can have 2^n values simultaneously!
  - 3-bit example:

    $\Psi = C_{000}|000> + C_{001}|001> + C_{010}|010> + C_{011}|011> + C_{100}|100> + C_{101}|101> + C_{110}|110> + C_{111}|111>$
- Probabilities of measuring all bits are set by coefficients:
  - So, prob of getting |000> is $|C_{000}|^2$, etc.
  - Suppose we measure only one bit (first):
    - » We get "0" with probability: $P_0 = |C_{000}|^2 + |C_{001}|^2 + |C_{010}|^2 + |C_{011}|^2$
      Result: $\Psi = (C_{000}|000> + C_{001}|001> + C_{010}|010> + C_{011}|011>)$
    - » We get "1" with probability: $P_1 = |C_{100}|^2 + |C_{101}|^2 + |C_{110}|^2 + |C_{111}|^2$
      Result: $\Psi = (C_{100}|100> + C_{101}|101> + C_{110}|110> + C_{111}|111>)$
- Problem: Don't want environment to *measure* before ready!
  - Solution: Quantum Error Correction Codes!

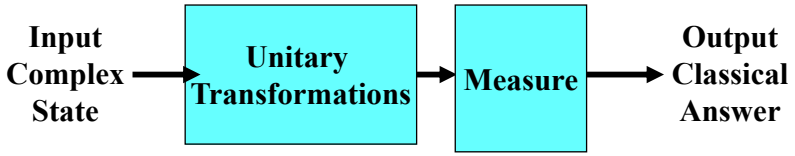
Spooky action at a distance

- Consider the following simple 2-bit state:

  $\Psi = C_{00}|00> + C_{11}|11>$
  - Called an "EPR" pair for "Einstein, Podolsky, Rosen"
- Now, separate the two bits:

- If we measure one of them, it instantaneously sets other one!
  - Einstein called this a "spooky action at a distance"
  - In particular, if we measure a |0> at one side, we get a |0> at the other (and vice versa)
- Teleportation
  - Can "pre-transport" an EPR pair (say bits X and Y)
  - Later to transport bit A from one side to the other we:
    - » Perform operation between A and X, yielding two classical bits
    - » Send the two bits to the other side
    - » Use the two bits to operate on Y
    - » Poof! State of bit A appears in place of Y

Model: Operations on coefficients + measurements



- Basic Computing Paradigm:
  - Input is a register with superposition of many values
    - » Possibly all 2n inputs equally probable!
  - Unitary transformations compute on coefficients
    - » Must maintain probability property (sum of squares = 1)
    - » Looks like doing computation on all 2n inputs simultaneously!
  - Output is one result attained by measurement
- If do this poorly, just like probabilistic computation:
  - If 2n inputs equally probable, may be 2n outputs equally probable.
  - After measure, like picked random input to classical function!
  - All interesting results have some form of "fourier transform" computation being done in unitary transformation

Shor's Factoring Algorithm

- The Security of RSA Public-key cryptosystems depends on the difficulty of factoring a number N=pq (product of two primes)
  - Classical computer: sub-exponential time factoring
  - Quantum computer: polynomial time factoring
- Shor's Factoring Algorithm (for a quantum computer)

Easy 1) Choose random $x : 2 \le x \le N\text{-}1$.
Easy 2) If $\gcd(x,N) \neq 1$, Bingo!
Hard 3) Find smallest integer $r : x^r \equiv 1 \pmod N$
Easy 4) If r is odd, GOTO 1
Easy 5) If r is even, $a \equiv x^{r/2} \pmod N \Rightarrow (a\text{-}1)\times(a+1) = kN$
Easy 6) If $a \equiv N\text{-}1 \pmod N$ GOTO 1
Easy 7) ELSE $\gcd(a \pm 1, N)$ is a non trivial factor of N.

Finding r with $x^r \equiv 1 \pmod N$

$$\sum_k |k\rangle |1\rangle \rightarrow \sum_k |k\rangle |x^k\rangle$$

$$= \sum_{w=0}^{r-1} \sum_y |w + ry\rangle |x^w\rangle$$

Quantum Fourier Transform

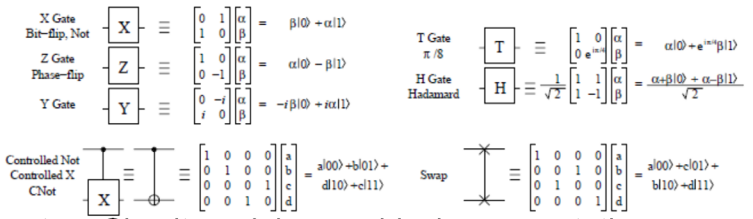
$$\rightarrow \sum_{w=0}^{r-1} \left(\int \frac{0}{r} + \int \frac{1}{r} + \int \frac{k}{r} + \int \right) |x^w\rangle$$

- Finally: Perform measurement
  - Find out r with high probability
  - Get $|y\rangle|a^{w'}\rangle$ where y is of form k/r and w' is related

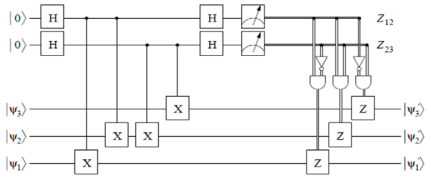
Quantum Computing Architectures

- Why study quantum computing?
  - Interesting, says something about physics
    - » Failure to build ⇒ quantum mechanics wrong?
  - Mathematical Exercise (perfectly good reason)
  - Hope that it will be practical someday:
    - » Shor's factoring, Grover's search, Design of Materials
    - » Quantum Co-processor included in your Laptop?
- To be practical, will need to hand quantum computer design off to classical designers
  - Baring Adiabatic algorithms, will probably need 100s to 1000s (millions?) of working logical Qubits ⇒ 1000s to millions of physical Qubits working together
  - Current chips: ~1 billion transistors!
- Large number of components is realm of *architecture*
  - What are optimized structures of quantum algorithms when they are mapped to a physical substrate?
  - Optimization not possible by hand
    - » Abstraction of elements to design larger circuits
    - » Lessons of last 30 years of VLSI design: USE CAD

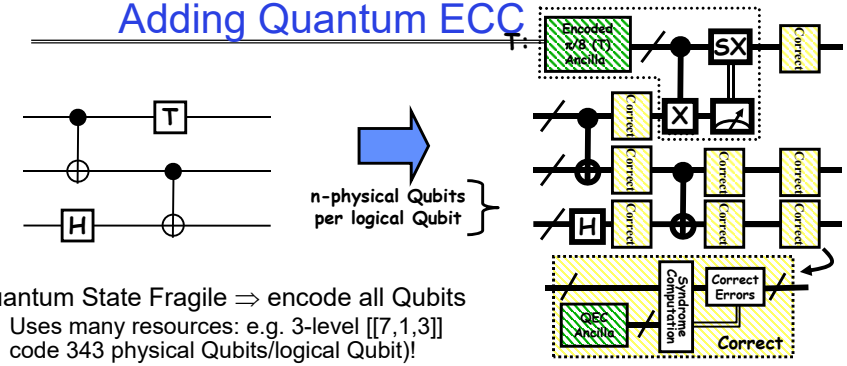
Quantum Circuit Model



- Quantum Circuit model – graphical representation
  - Time Flows from left to right
  - Single Wires: persistent Qubits, Double Wires: classical bits
    - » Qubit – coherent combination of 0 and 1: $\psi = \alpha|0\rangle + \beta|1\rangle$
  - Universal gate set: Sufficient to form all unitary transformations
- Example: Syndrome Measurement (for 3-bit code)
  - Measurement (meter symbol) produces classical bits
- Quantum CAD
  - Circuit expressed as netlist
  - Computer manpulated circuits and implementations



Adding Quantum ECC



- Quantum State Fragile ⇒ encode all Qubits
  - Uses many resources: e.g. 3-level [[7,1,3]] code 343 physical Qubits/logical Qubit)!
- Still need to handle operations (fault-tolerantly)
  - *Some* set of gates are simply "transversal:"
    - » Perform identical gate between each physical bit of logical encoding
  - Others (like T gate for [[7,1,3]] code) cannot be handled transversally
    - » Can be performed fault-tolerantly by preparing appropriate ancilla
- Finally, need to perform periodical error correction
  - Correct after every(?): Gate, Long distance movement, Long Idle Period
  - Correction reducing entropy ⇒ Consumes Ancilla bits
- Observation: ≥ 90% of QEC gates are used for ancilla production
  ≥ 70-85% of all gates are used for ancilla production

Outline

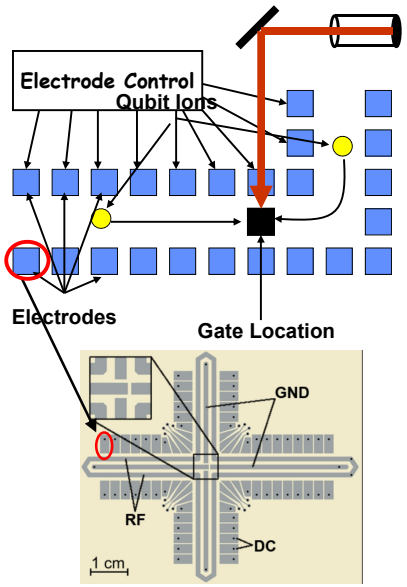
- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

MEMs-Based Ion Trap Devices

- Ion Traps: One of the more promising quantum computer implementation technologies
  - Built on Silicon
    - » Can bootstrap the vast infrastructure that currently exists in the microchip industry
  - Seems to be on a "Moore's Law" like scaling curve
    - » Many researchers working on this problem
  - Some optimistic researchers speculate about room temperature
- Properties:
  - Has a long-distance Wire
    - » So-called "ballistic movement"
  - Seems to have relatively long decoherence times
  - Seems to have relatively low error rates for:
    - » Memory, Gates, Movement

Quantum Computing with Ion Traps

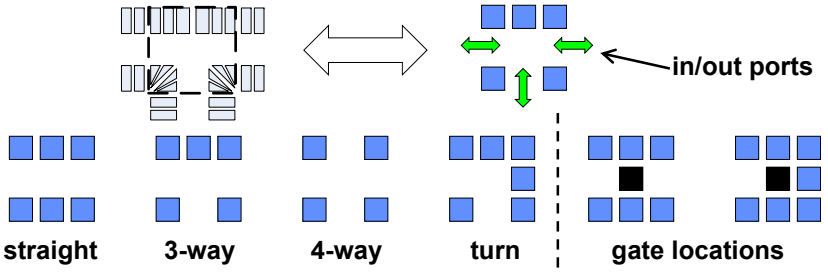
- Qubits are atomic ions (e.g. Be^+)
  - State is stored in hyperfine levels
  - Ions suspended in channels between electrodes
- Quantum gates performed by lasers (either one or two bit ops)
  - Only at certain trap locations
  - Ions move between laser sites to perform gates
- Classical control
  - Gate (laser) ops
  - Movement (electrode) ops
    - Complex pulse sequences to cause Ions to migrate
    - Care must be taken to avoid disturbing state
- Demonstrations in the Lab
  - NIST, MIT, Michigan, many others



Courtesy of Chuang group, MIT

An Abstraction of Ion Traps

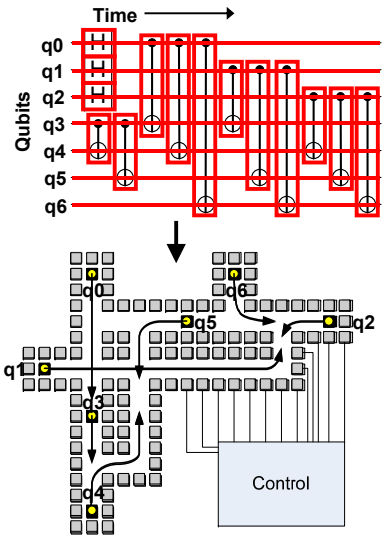
- *Basic block* abstraction: Simplify Layout



- Evaluation of layout through simulation
  - Yields Computation Time and Probability of Success
- Simple Error Model: Depolarizing Errors
  - Errors for every Gate Operation and Unit of Waiting
  - Ballistic Movement Error: Two error Models
    1. Every Hop/Turn has probability of error
    2. Only Accelerations cause error

Ion Trap Physical Layout

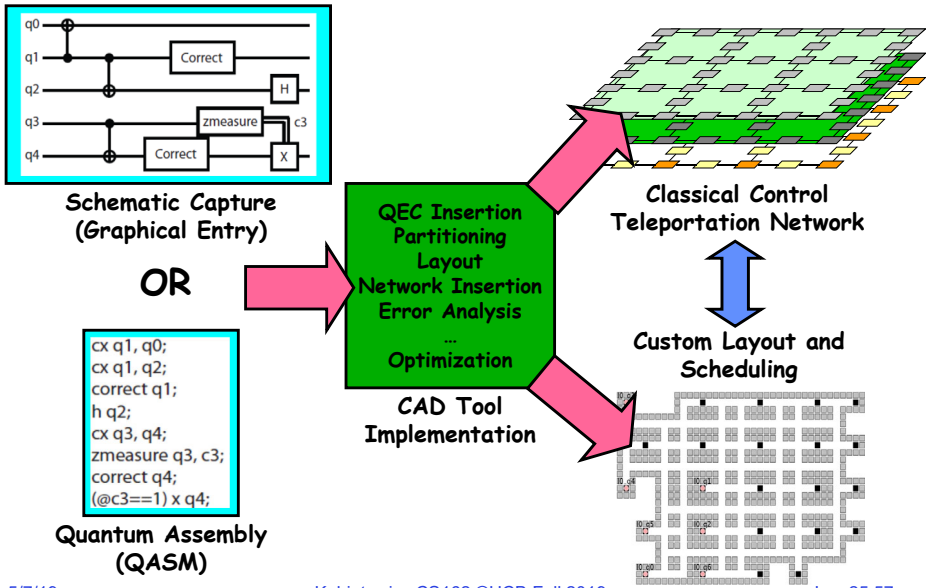
- Input: Gate level quantum circuit
  - Bit lines
  - 1-qubit gates
  - 2-qubit gates
- Output:
  - Layout of channels
  - Gate locations
  - Initial locations of ions
  - Movement/gate schedule
  - Control for schedule



Outline

- Quantum Computering
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

Vision of Quantum Circuit Design

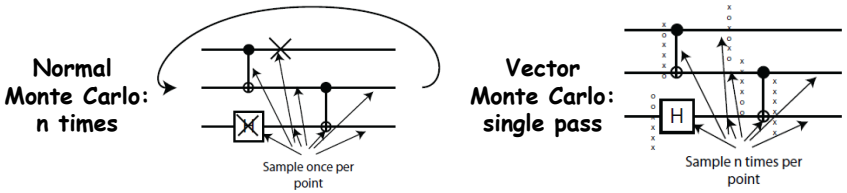


Important Measurement Metrics

- Traditional CAD Metrics:
  - Area
    - » What is the total area of a circuit?
    - » Measured in macroblocks (ultimately μm^2 or similar)
  - Latency ($Latency_{single}$)
    - » What is the total latency to compute circuit *once*
    - » Measured in seconds (or μs)
  - Probability of Success ($P_{success}$)
    - » Not common metric for classical circuits
    - » Account for occurrence of errors and error correction
- Quantum Circuit Metric: ADCR
  - Area-Delay to Correct Result: Probabilistic Area-Delay metric
  - ADCR = Area × E(Latency) = $\frac{Area \times Latency_{single}}{P_{success}}$
  - $ADCR_{optimal}$: Best ADCR over all configurations
- Optimization potential: Equipotential designs
  - Trade Area for lower latency
  - Trade lower probability of success for lower latency

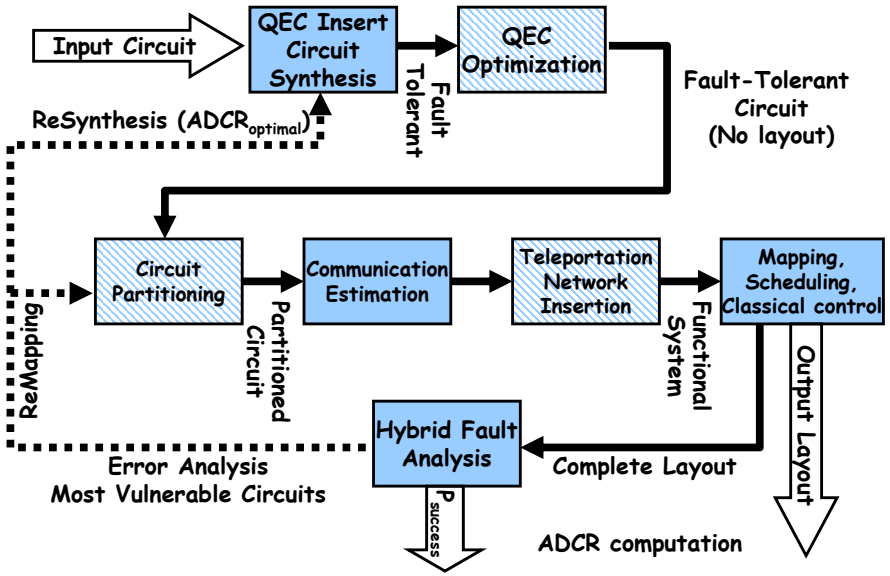
How to evaluate a circuit?

- First, generate a physical instance of circuit
  - Encode the circuit in one or more QEC codes
  - Partition and layout circuit: Highly dependant of layout heuristics!
    - » Create a physical layout and scheduling of bits
    - » Yields area and communication cost



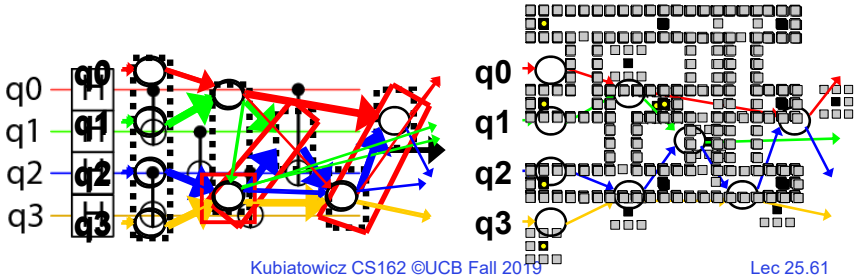
- Then, evaluate probability of success
  - Technique that works well for depolarizing errors: Monte Carlo
    - » Possible error points: Operations, Idle Bits, Communications
  - Vectorized Monte Carlo: n experiments with one pass
  - Need to perform hybrid error analysis for larger circuits
    - » Smaller modules evaluated via vector Monte Carlo
    - » Teleportation infrastructure evaluated via fidelity of EPR bits
- Finally – Compute ADCR for particular result

Quantum CAD flow



Example Place and Route Heuristic: Collapsed Dataflow

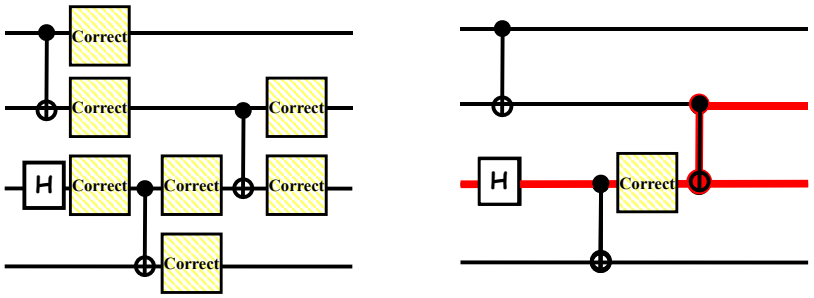
- Gate locations placed in dataflow order
  - Qubits flow left to right
  - Initial dataflow geometry folded and sorted
  - Channels routed to reflect dataflow edges
- Too many gate locations, collapse dataflow
  - Using scheduler feedback, identify latency critical edges
  - Merge critical node pairs
  - Reroute channels
- Dataflow mapping allows pipelining of computation!



Outline

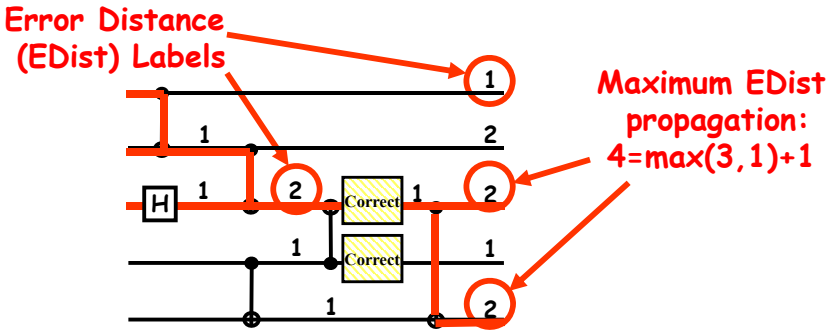
- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

Reducing QEC Overhead



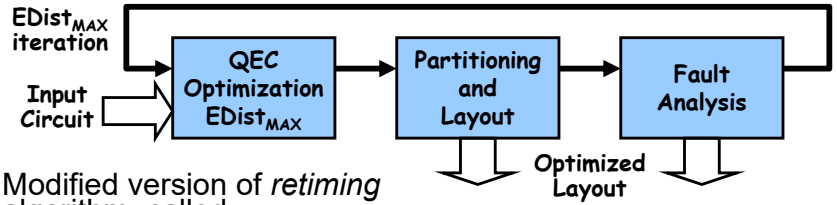
- Standard idea: correct after every gate, and long communication, and long idle time
  - This is the easiest for people to analyze
- This technique is suboptimal (at least in some domains)
  - Not every bit has same noise level!
- Different idea: identify critical Qubits
  - Try to identify paths that feed into noisiest output bits
  - Place correction along these paths to reduce maximum noise

Simple Error Propagation Model



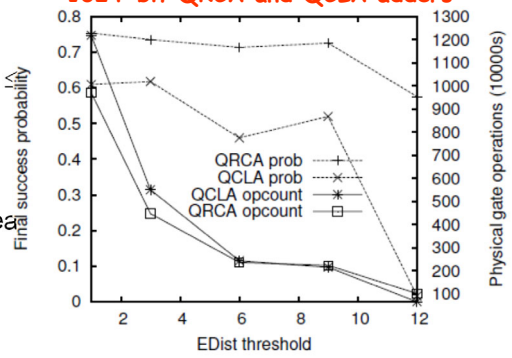
- EDist model of error propagation:
  - Inputs start with EDist = 0
  - Each gate propagates max input EDist to outputs
  - Gates add 1 unit of EDist, Correction resets EDist to 1
- Maximum EDist corresponds to Critical Path
  - Back track critical paths that add to Maximum EDist
- Add correction to keep EDist below critical threshold

QEC Optimization

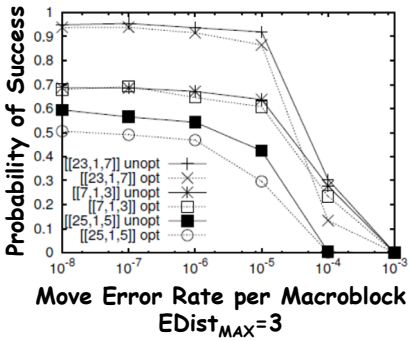


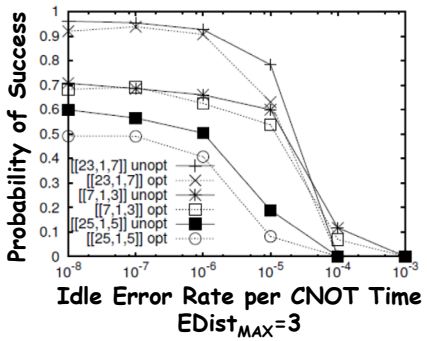
- Modified version of *retiming* algorithm: called "recorrection:"
  - Find minimal placement of correction operations that meets specified MAX(EDist) ≤ EDist$_{MAX}$
- Probably of success *not* always reduced for EDist$_{MAX}$ > 1
  - But, operation count and area drastically reduced
- Use Actual Layouts and Fault Analysis
  - Optimization *pre-layout*, evaluated *post-layout*

1024-bit QRCA and QCLA adders



Recorrection in presence of different QEC codes



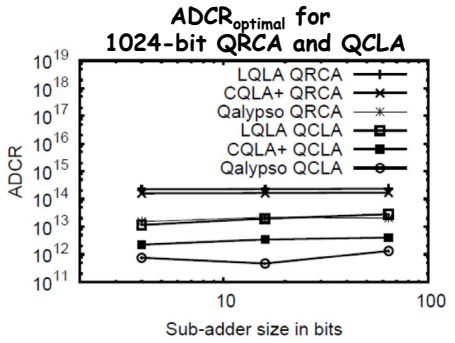


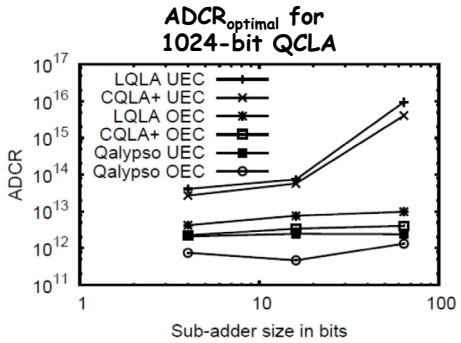
- 500 Gate Random Circuit (r=0.5)
- Not all codes do equally well with Recorrection
  - Both [[23,1,7]] and [[7,1,3]] reasonable candidates
  - [[25,1,5]] doesn't seem to do as well
- Cost of communication and Idle errors is clear here!
- However – real optimization situation would vary EDist to find optimal point

Outline

- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

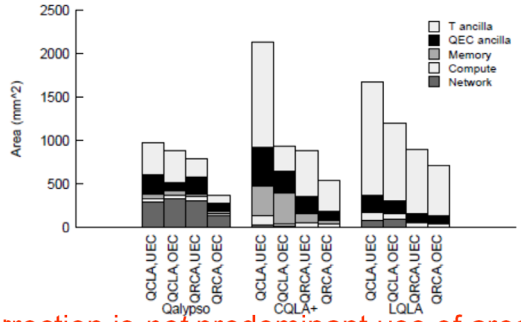
Comparison of 1024-bit adders





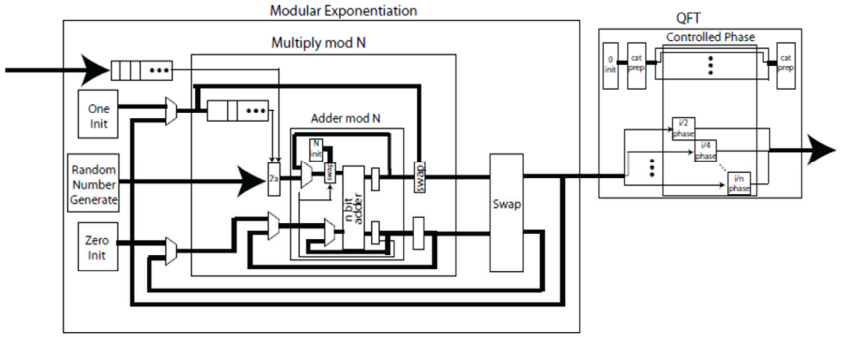
- 1024-bit Quantum Adder Architectures
  - Ripple-Carry (QRCA)
  - Carry-Lookahead (QCLA)
- Carry-Lookahead is better in all architectures
- QEC Optimization improves ADCR by order of magnitude in some circuit configurations

Area Breakdown for Adders



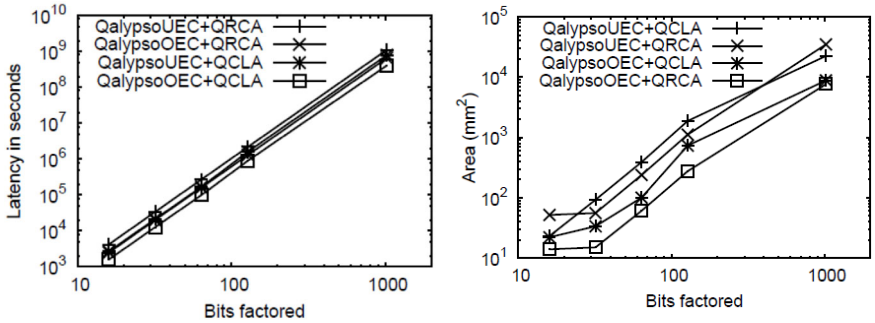
- Error Correction is *not* predominant use of area
  - Only 20-40% of area devoted to QEC ancilla
  - For Optimized Qalypso QCLA, 70% of operations for QEC ancilla generation, but only about 20% of area
- T-Ancilla generation is major component
  - Often overlooked
- Networking is significant portion of area when allowed to optimize for ADCR (30%)
  - CQLA and QLA variants didn't really allow for much flexibility

Investigating 1024-bit Shor's



- Full Layout of all Elements
  - Use of 1024-bit Quantum Adders
  - Optimized error correction
  - Ancilla optimization and Custom Network Layout
- Statistics:
  - Unoptimized version: 1.35×10^{15} operations
  - Optimized Version 1000X smaller
  - QFT is only 1% of total execution time

1024-bit Shor's Continued



- Circuits too big to compute $P_{success}$
  - Working on this problem
- Fastest Circuit: 6×10^8 seconds ~ 19 years
  - Speedup by classically computing recursive squares?
- Smallest Circuit: 7659 mm^2
  - Compare to previous *estimate* of 0.9 m^2 = 9×10^5 mm^2

In Conclusion

- Cryptography is a mechanism that is helpful for enforcing a security policy
  - Encryption, Hashing, Digital Signatures
- It's all about the Data!
  - Hardening the Data while freeing it to reside anywhere
  - Edge Computing Enabled by DataCapsules
- Quantum Computing
  - Computing using interesting properties of Physics
  - Achieving Quantum Supremacy: Proof that Quantum Computers are more powerful than Classical Ones
    - » Not there yet!
- Most interesting Applications of Quantum Computing:
  - Materials Simulation
  - Optimization problems
  - Machine learning?