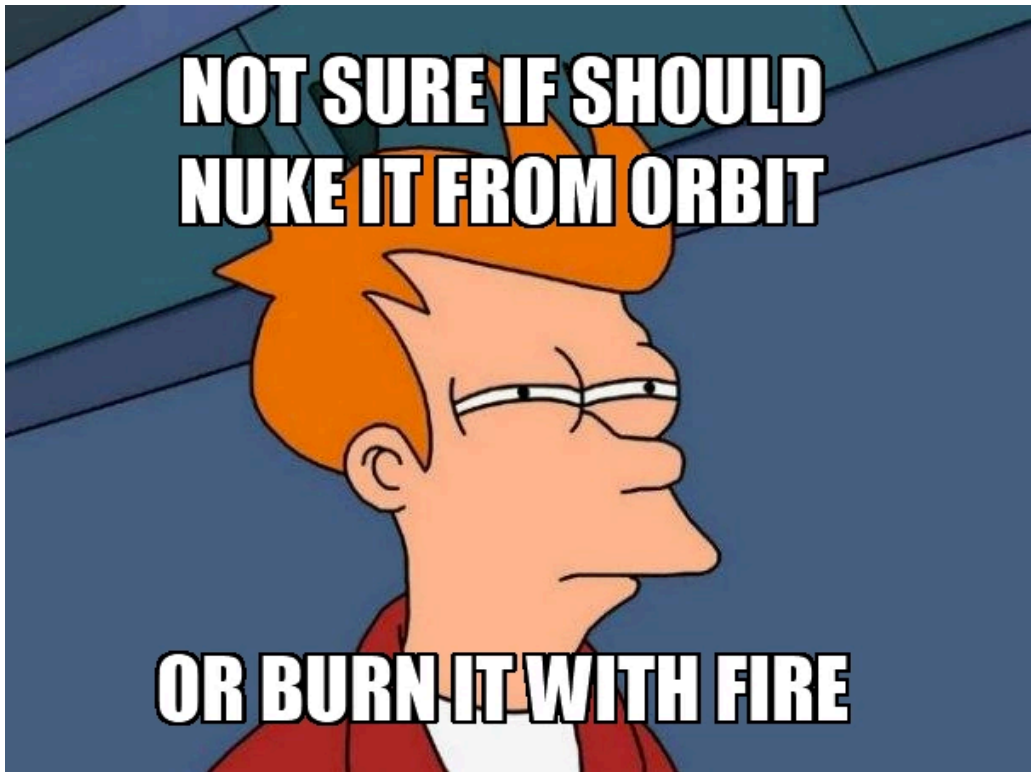
More Dealer's Choice



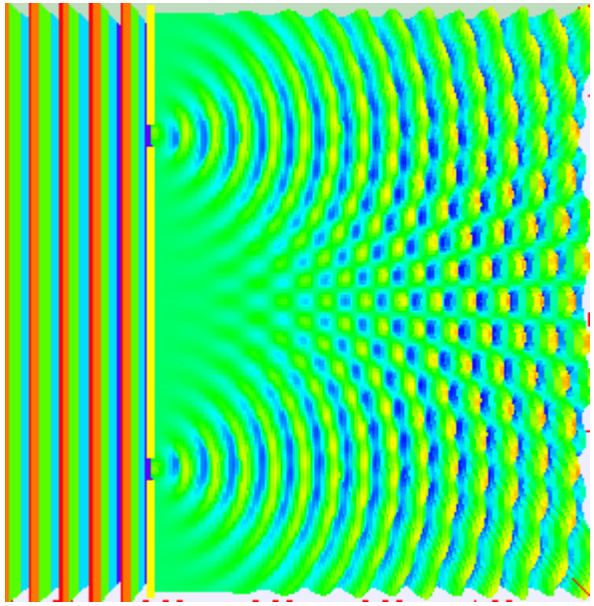
And Now Onto "Dealer's Choice"

- Dealer's Choice material is always implicit blue-slide
  - These are interesting, useful, but not easily testable.
  - But you want to learn the lessons of these
- Today:
  - Quantum
  - Nukes
  - Sidechannelss

Quantum Mechanics: The Weird Reality...

- At the scale of individual atoms, our intuition breaks down...
  - Things behave like both particles and waves
  - Things can pass through other things
  - Things can be in multiple states at once
  - Probabilities matter
- I don't think anyone really intuitively ***understands*** Quantum...
  - But it works...
- Disclaimer: I'm a failure at Quantum:
  - I got a C (I deserved an F) in Physics 137A, this is truly weird stuff!

Example Weirdness: The Double Slit Experiment



- If you beam a light at a set of double slits
  - You get a wave diffraction pattern 🙂
- If you bean a beam of electrons...
  - You get a wave diffraction pattern?! 🤔
- But light is composed of "photons" and electrons are clearly particles
  - If you send them one at a time, each one arrives at single points, but…
  - Taken together you get a diffraction pattern 🤷
- But if you ***measure*** which slit each particle went through...
  - You eliminate the diffraction pattern!
  - Single electrons and photon "particles" are interfering ***with themselves*** like a wave does! 😕

So What Does This Mean?

- Things are both particles and waves?!?
- Things can be in two places at once?
- When you measure something, it behaves differently?
- EG, Schrodinger's cat...
  - A thought problem: You have a cat in a sealed box, a vial of poison, and a single radioactive atom...
    - At time T, there is a 50% chance the atom decayed, broke the poison, and killed the cat
  - Is the cat alive? Dead? Both?
    - "Yes", until you open the box!

Another Weirdness: EPR entanglement

- Einstein ***hated*** quantum mechanics...
  - "God does not play dice with the universe"
  - Plus his genius idea, relativity, doesn't actually work with quantum...
    - If you can unite general relativity and quantum mechanics, you are getting a flight to Sweden to pick up your Nobel prize
- Einstein–Podolsky–Rosen came up with a "paradox"...
  - The "EPR pair"
  - Intended to go "See, this Quantum 💩 makes no sense..."
  - The problem is, it actually ***works!***

EPR "Paradox" in action

- We have two particles, A and B...
  - A is in an unknown state, 50% of the time A = 0, 50% of the time A = 1
    - Really, A is in a superposition of both states:
      The cat is alive and dead
  - If we measure A, we have a 50/50 chance at the time of measurement
  - But until we measure A, it continues to exist as probabilistic superposition of both states
- We then "entangle" B without measuring A
  - So that A=0 <-> B=0 and A=1 <-> B=1
  - And then separate the two, perhaps even by light years distance!
    - Note we really generate A and B at the same time in a random entangled state...
      We can't clone A->B perfectly but only with partial fidelity
- Now when we measure
  - If A = 0 we will ALWAYS see B =0…
    - But if A = 1 we will see B = 1
- And it doesn’t matter which way we order our observations
  - and it is still random which one it is?!?

As long as we maintain coherence...

- We can keep this up!
  - So lets take several bits, B_0, B_1, B_2
  - Put each one in an independent 50/50 state. These are now qbits (Quantum Bits)
- Now we do a computation:
  - $B_3 = B_0 \oplus B_1 \oplus B_2$
  - But while maintaining coherence
- Now the spooky thing...
  - We've really computed all quantum superposition of all possible values of B_3 as a function of B_0-B_2...
    - In hardware language it is like we computed the ***entire*** truth table in one go and things are existing in that superposition
- But if we ***measure*** them, we get just a single input/output pair

And Now The Quantum Miracle...

- So far, this is no more powerful than a conventional computer
  - After all, we still only get a single output for a single set of inputs...
- But then we get to the Quantum magic...
- We now take B_0-B_3 and pass them through another transformation
  - That basically self-interferes between the superposition of all the input/output pairs
- And now when we look...
  - We see some information about the ***relationship*** between all the bits!
- But we need to maintain this in a quantum state (coherence) to work...
  - Any little noise or interaction with the outside world and the wave function collapses to a single Input/Output pair!

So What Good Is This?

- Shor developed an algorithm to solve two different & related group theory problems
  - Find the order of a group
    - Given a group ***G***, a generator ***g***, how many elements are in the group?
    - You can reduce factoring to this problem
  - Find the discrete log
    - Given a group ***G*** of known order, a generator ***g***, and a value g^x mod ***G***, what is ***x***?
- The number of quantum bits (qbits) required:
  - O((log ***N***)^2 (log log ***N***) (log log log ***N***)) with ***N*** the number to be factored
  - So still a lot of quantum state: millions of qbits for a 2048b RSA key
- Oh, and this is just about the only thing it is good for

This Breaks All Major Public Key

- Diffie/Hellman: Break discrete log
- RSA: Break factoring
- Elliptic Curve
  - It's more complicated because you don't know the order of the group...
  - Well, its not actually. See the footnote on the "factoring" algorithm!
    - You use the RSA algorithm to get the order of the group
    - And then use the discrete log problem
- But what does this actually mean?

Implications #1: Is ECC better?

- In conventional computing: ECC is the same strength with fewer bits
  - 256b ECC ~= 2048b RSA & DH
    - There are sub-exponential shortcuts for the group-theory problems in the integers not present on elliptic curves
- But this isn't the case with quantum computing!
  - So if we could only build a "medium-sized-ish" quantum computer (tens of thousands rather than millions of qbits), ECC breaks first
- Speculation: Is this why in going from Suite B to CNSA, the NSA said...
  - "Whoah, hold off on going to ECC until we have post-Quantum public key... and until then you can use 3096b RSA and DH as well"

Implication #2:
Lots of work on "Post-Quantum Public Key"

- A major area of active research: public key algorithms without a quantum shortcut
  - Significantly larger keys: 400B (same as 3096b RSA) to 10,000B depending on the algorithm
- In practice, never used alone!
- EG, the "NewHope" TLS handshake experiment
  - Does both an ECDHE and post-quantum public key agreement: Both would have to be broken to break the system

Implication #3:
***Don't Worry*...**

- There may be exponential or near exponential difficulties in maintaining coherence as a function of the # of qbits
  - Open question: There is a lot of work on this, but 🤷.
  - I've heard "Quantum Computers Real Soon Now" for nearly 25 years!
- The current "Quantum" computers really aren't
  - D-Wave is actually "quantum annealing":
    2-D simulated annealing with Quantum acceleration. Open question whether it is fundamentally faster
  - Google's "Quantum Supremacy":
    Better than a classical computer at computing how it will compute?!?
    Again, only 2D not generic operations
- True generic quantum computers have been built...
  - But it is unclear whether the "factoring" exercises are generic, given the # of qbits themselves are relatively small

Post Quantum Cryptography...

- Just because ***you*** don't need to worry...
  - Doesn't mean the cryptographers aren't worried
- So repeating the success of AES and SHA3:
  - A NIST organized contest to develop new algorithms
    Now a set of finalists
- Two main primitives:
  - Key exchange (analogous to Diffie/Hellman)
  - Signatures
- Designed to be used in concert with a conventional key exchange
  - That way you'd have to break both:
    Use ***KDF(PostQuantum || Classic)*** to generate the session keys

But What About "Quantum Cryptography"

- Really, its Quantum Key Exchange...
- Take a single photon:
  - We can measure its polarization in either **+** or **X** orientation, and select it randomly
    - Gives us a single photon with a random polarization
  - We then transmit to the photon to the recipient... Who then does the same thing
- We then broadcast which orientations we used...
  - If they chose the different orientation, they end up with a random value
  - If they chose the same one, they end up with the ***same*** value...
    - But if there is an eavesdropper, an eavesdropper adds noise 50% of the time: And that noise corrupts the result 50% of ***that*** time
  - "Provably secure" assuming our knowledge of physics is correct
    - All noise is treated as being introduced by the eavesdropper

Quantum Key Exchange is a Total Waste...

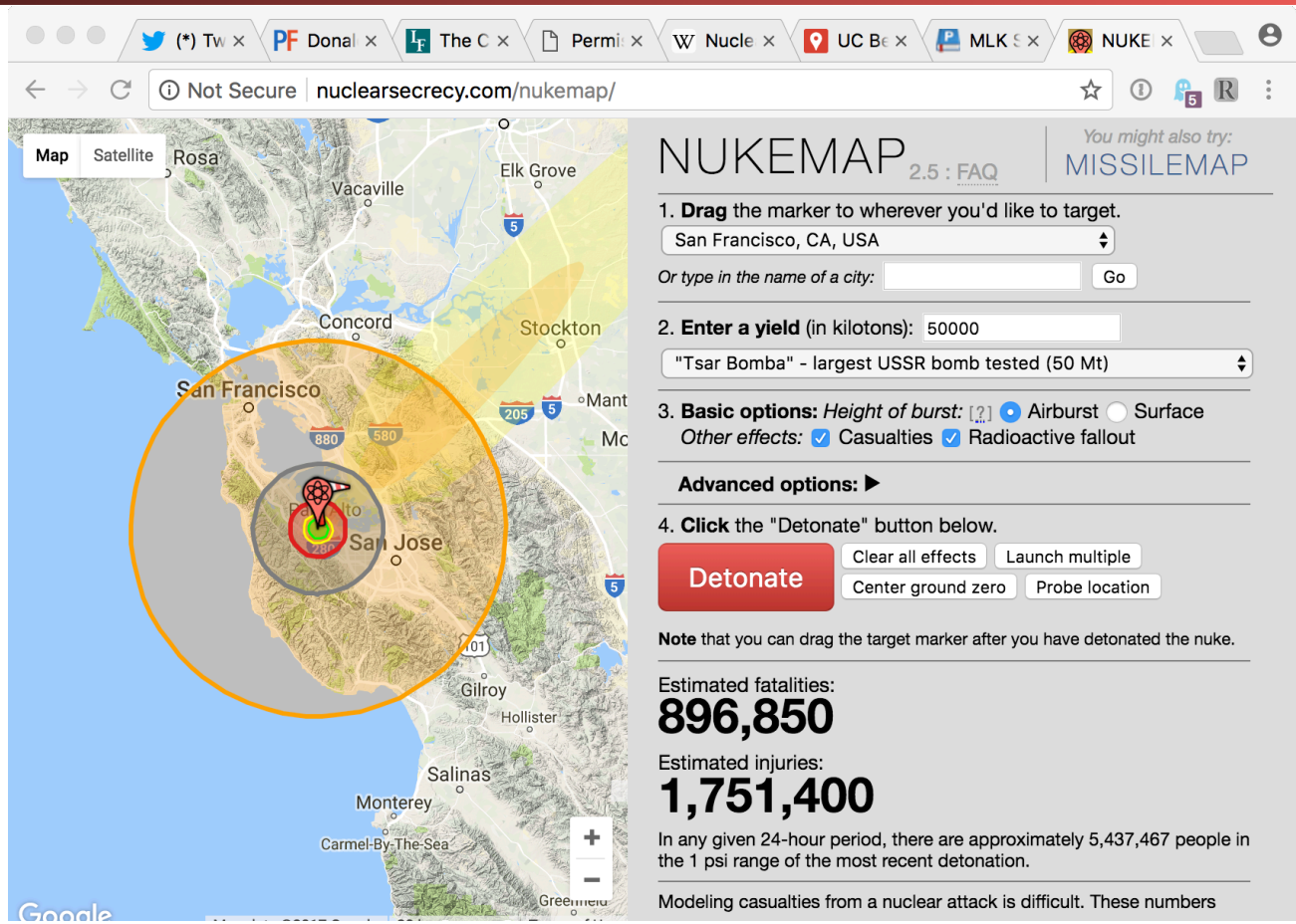
- Of course this requires sending single photons with no effective noise to a recipient
  - Point to point without routing and with ***insanely*** low noise requirements...
- If we can't build a large Quantum computer or break existing public key, ***it is completely useless***
- If we can build a large Quantum computer but can make post-Quantum public key work, ***it is completely useless***
- If we can build a large Quantum computer and post-Quantum Public Key fails, it is ***still*** completely useless!
  - This only works for point to point, so you might as well just ship around USBs full of random key material!
  - And then scale beyond using Kerberos type systems:
    A trusted third party has pairwise links to Alice and Bob, key is generate and shared by the trusted third party

Oh, and it is worse than you think...

- Quantum Key Exchange ***requires*** the two parties to have an ***authenticated*** channel
  - So the security proof ***requires*** authentication work already!
  - Otherwise you can just do a DH style MitM attack...
- But if Alice and Bob can distribute the necessary key material to guarantee channel integrity beforehand....
  - They could have ***also*** included a shared symmetric key for confidentiality!
  - Or heck, if everything broke, 4 TB of data on a removable drive for a one-time pad!

Why talk about nukes?

- Nukes are big and scary and in the news...
  - But have interesting security and safety properties
- Lots of material ~~stolen~~ borrowed from Steve Bellovin's excellent talk on PALs



How a Nuclear Weapon Works...

- 1960s-level technology...
  - A hollow sphere of fissile material
    - Plutonium and/or Plutonium + Uranium
  - Use this as a primary to ignite a Teller/Ulam secondary to make it a hydrogen bomb...

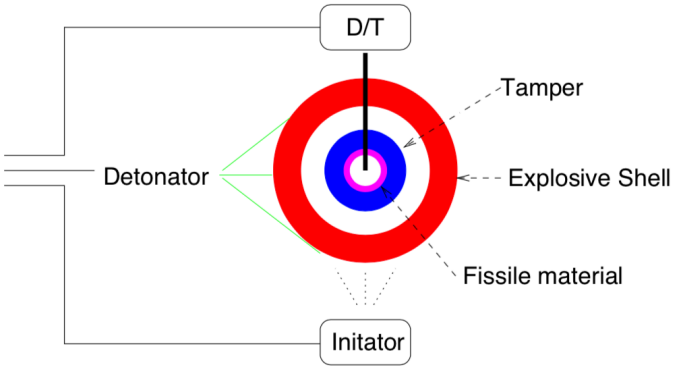
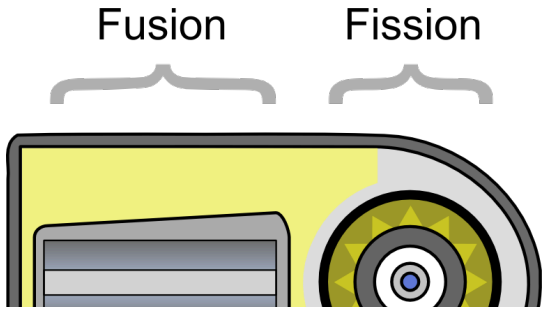


Diagram by Steve Bellovin

- Very careful sequencing needed
  - D/T pump to fill the hollow with Deuterium & Tritium ("Boost gas")
    - Not needed for the earliest bombs, but most modern bombs need boosting to work
  - Initiator sprays neutrons to start the chain reaction
  - Detonator needs to trigger multiple points on the explosive shell
    - Squiggly-traces of explosive so that all around the shell everything detonates at once

And H-Bombs...

- A "Tellar/Ulam" 2-stage device:
  A A-bomb ignites a fusion stage
  - Fusion stage has Lithium Deuteride...
    - Neutrons and pressure from the A-bomb convert the Lithium to Tritium
    - Then Deuterium/Tritium fusion makes it go boom!
- Still 1960s technology!
  - Biggest issue overall is materials:
    6 or 7 countries have built H-Bombs



And How To Deliver Them...

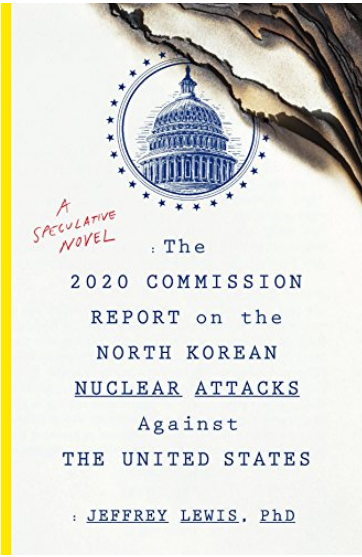
- Stick em on a rocket
  - This ***is*** rocket science: It is far easier to build the nuke than build the ICBM...
  - Alternatively, stick it on an unmanned miniature airplane ("Cruise Missile") or just hang it under a plane as a old-fashioned bomb
- Then stick the rocket on something
  - In a hardened silo
    - But the other side can drop a nuke on it...
  - On a truck
  - In a sub
  - On a plane...

The Problem: When To Use Nukes...

- Nuclear weapon systems can fail in two ways:
  - Launch the nukes when you shouldn't...
  - Fail to launch the nukes when you should...
- The latter is (badly) addressed by how our nuclear decision making happens
  - "Launch on warning": If we ***think*** we are under attack, the President has a couple minutes to decide to order a nuclear strike before the attacker hits our ICBMs!
    - This is often regarded as ***insanely*** stupid: We have both nuclear bombers with long-range cruise missiles and nuclear armed submarines, both of which ***will*** be able to launch enough retaliatory hellfire
  - Far better is the "French model" (cite @armscontrolwonk): "We have subs. You nuke us ***or*** attack our strategic weapons and we nuke you":
    - This removes the time pressure which can cause errors

"Launch on Warning" and North Korea...

- Let us assume that North Korea's leadership are ***rational*** actors
  - They act in what they perceive as their self interest: survival!
- North Korean leadership ***will eventually lose*** a war with South Korea and the US
  - So they may be provocative, but they want to make ***sure*** the US and South Korea won't start a war
- Nukes are a critical deterrent for them
  - Especially when Donald Trump didn't seem to care that a war would kill hundreds of thousands in South Korea
- IRBMs and ICBMs are as important as the nukes themselves!
  - Need to be able to hit the US bases in Okinawa and Guam as military targets
  - And last year Mar-a-lago and Washington DC to dissuade Trump personally: The Hwasong-15 ICBM can just barely range South Florida.
- "***Empathy*** for the devil"
  - Computer security is adversarial, think about your adversary's needs, wants, and desires



Launch on Warning and the US C&C Structure

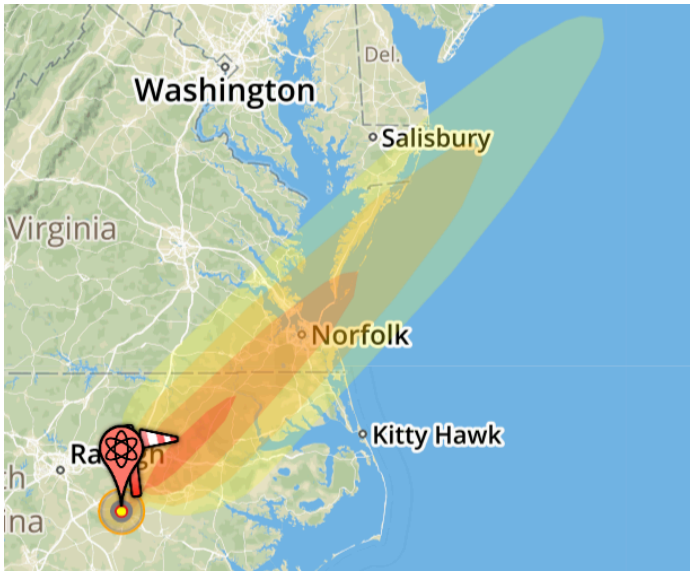
- The President has three items:
  - A “biscuit” of authentication codes kept on his person
  - The “football”: containing a menu of options for ordering a nuclear strike
  - An encrypted secure phone
- The President has a bad day…
  - He calls over the football
    - Picks out the menu option he wants to use..
  - He calls NORAD on the phone
    - Taking out the biscuit, opening it, and getting the authentication code of the day
    - Saying what menu option he wants
  - < 5 minutes later, the ICBMs leave their silos
    - And there is no “recall code”

The Interesting Problem: Limiting Use

- Who might use a nuke without authorization?
  - Our "allies" where we station our nukes
    - Original motivation: Nukes stored in Turkey and Greece
  - Someone who can capture a nuke
    - This is what sold the military on the need for the problem:
      We had nukes in Germany which ***would*** be overrun in case of a war with the USSR
  - Our own military
    - General Jack D Ripper scenario
- The ***mandated*** solution:
  - Permissive Access Link (PAL)

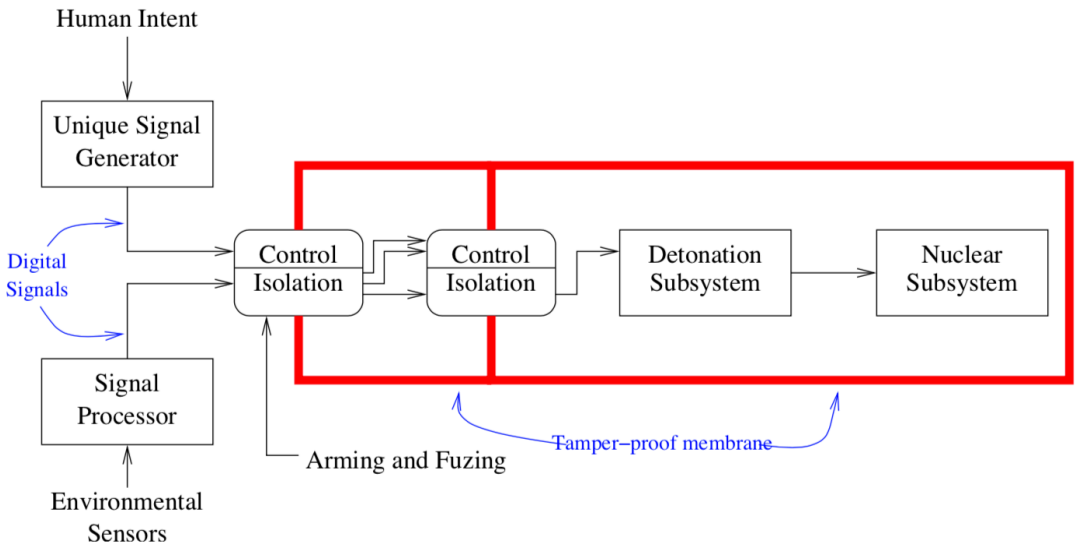
Nuke Safety Features

- One-point safety – no nuclear yield from detonation of one explosive charge.
- Strong link/weak link –
  - strong link provides electrical isolation;
  - weak link fails early under stress (heat, etc.)
- Environmental sensors – detect flight trajectory.
- Unique signal generator – digital signal used for coupling between stages.
- Insulation of the detonators from electrical energy.
- "Human intent" input.
- Tamper-resistant skin
- Use Control Systems
- Not always the case: In 1961 in South Carolina a B52 broke up
  - One of the two 4MT bombs ***almost*** detonated on impact, since it thought it was being dropped!



Bomb Safety Systems

- We have a "trusted base"
  - Isolated inside a tamper-detecting membrane
    - Breach the membrane -> disable the bomb
- We have human input
  - Used to generate a signal saying "its OK to go boom"
    - The user interface to the PAL can follow the same path/concepts
- We have critical paths that we can block
  - Complete mediation of the signal to go boom!



Unique Signal Generator

- Part of the strong link
  - Prevent any detonation without clear, unambiguous showing of “human intent”
- A ***safety*** system, not a security system
- Looks for a 24-bit signal that is extremely unlikely to happen during any conceivable accident. (Format of input bits not safety-critical)
  - Accidents can generate random or non-random data streams
  - Desired signal pattern is unclassified!
- Unique signal discriminator locks up on a ***single*** erroneous bit
- At least partially mechanical

PALs

- Originally electromechanical. (Some weapons used combination locks!)
- Newest model is microprocessor-based. There may still be a mechanical component.
  - Recent PAL codes are 6 or 12 digits.
- The weapon will permanently disable itself if too many wrong codes are entered.
- PALs respond to a variety of codes – several different arming codes for different groups of weapons, disarm, test, rekey, etc.
- It was possible, though difficult, to bypass early PALs.
  - Some even used false markings to deceive folks who didn't have the manual.
- It does not appear to be possible to bypass the newest "CAT F" PAL.
  - Modern bombs don't work without the tritium boost-gas:
    If you blow the gas you disable the nuke. Don't know if this is done or not

How are PALs built?

- We don't know, but some informed speculation from Steve...
- It is ***most likely*** based around the same basic mechanism as the unique signal generator
  - Gives a single point of control already in the system
  - Reports about it indicate that it was successfully evaluated in isolation
  - Take advantage of the existing trusted base of the tamper-resistant barrier around the warhead to protect the device

Deployment History

- Despite Kennedy's order, PALs were not deployed that quickly.
  - In 1974, there were still some unprotected nukes in Greece or Turkey
- PALs and use control systems were deployed on US-based strategic missiles by then
  - But the launch code was set to 00000000
  - Rational: the Air Force was more worried about failure to launch!
- A use control system was added to submarine-based missiles by 1997
- In 1981, half of the PALs were still mechanical combination locks

Steve Bellovin's Lessons Learned

- Understand what problem you're solving
- Understand ***exactly*** what problem you're solving
- If your abstraction is right:
  you can solve the key piece of the overall puzzle
- For access control, find the One True Mandatory Path — and block it.
  - And if there is more than one, you're doing it wrong!
- What is the real TCB of our systems?

Side Channels & Other Hardware Attacks: Worry

- A side channel attack requires measuring some other piece of information
  - EG, time, cache state, power consumption, etc...
- And using it to deduce a secret about the system
- Side channels are very, ***very*** powerful

Requirements

- Often the biggest limitation is attacker requirements
- Timing attack
  - Need to measure the timing of the operation with potentially very high precision
- Power attack
  - Need physical access to the device:
    Generally only applicable to smart-cards and similar devices
- EMF ("Tempest")
  - Need close physical access
- Processor side-channel attacks
  - Need to co-locate the attacker code:
    EG, cloud computing, web browsers, etc

Example Timing Attack: Keystrokes...

- User is inputting a password
  - And the user is using a Bluetooth keyboard...
  - Or the user is using a remote connection over ssh
- Someone nearby can observe when keys are pressed
  - They are sent immediately
  - But not ***what*** keys are pressed
- Can this leak sensitive information? Of course!

Timing Leakage

- Some keys are faster to press
- Can use this to model timing
  - Either generically or specific to the user
- Lots of ways to do this
  - Hidden markov models
  - Throw machine learning at it...
- Really really hard to hide
  - Can't delay interactive requests without adding latency
  - "Cover traffic" only adds additional data, can't remove the underlying signal
- From https://people.eecs.berkeley.edu/~daw/papers/ssh-use01.pdf

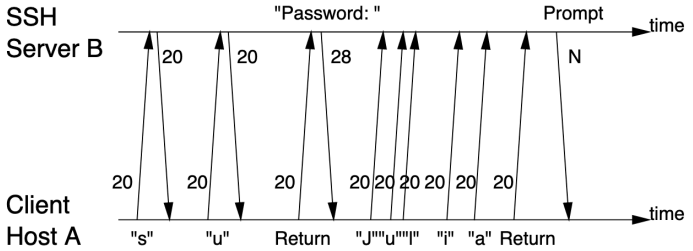


Figure 1: The traffic signature associated with running `SU` in a `SSH` session. The numbers in the figure are the size (in bytes) of the corresponding packet payloads.

Timing Attacks & Cryptography

- The classic timing attack:
  - Compute $y^x \bmod n$
- Easy solution ends up being

```
Let s_0 = 1.
For k = 0 upto w − 1:
  If (bit k of x) is 1 then
    Let R_k = (s_k · y) mod n.
  Else
    Let R_k = s_k.
  Let s_{k+1} = R_k^2 mod n.
EndFor.
Return (R_{w−1}).
```

- https://www.paulkocher.com/TimingAttacks.pdf

Implications:
Public Key Operations Need "Constant Time"

- Optimizing cryptographic code can be dangerous...
  - Instead it needs to take the same amount of time no matter what the input is
  - Even compiler optimizations can be a problem
- First identified 20 years ago...
  - So you think we'd have solved it...
    But you'd be wrong

Reminder DSA/ECDSA Brittleness...

- DSA algorithm
  - Global parameters: primes ***p*** and ***q***, generator ***g***
  - Message ***m***, private key ***x***, public key $y=g^x \bmod p$
  - Sign: select random ***k*** from 1 to ***q***-1
    $r = (g^k \bmod p) \bmod q$ (retry if r = 0)
    $s = (k^{-1} (H(M) + xr)) \bmod q$ (retry if s = 0)
- ***k*** needs to be random and secret and unique
  - An attacker who learns or guesses ***k*** can find ***x***
    - An attacker can even just try all possible ***k***s if the entropy of ***k*** is low
  - Even just learning a few bits of ***k***, and then having several signatures with different ***k*** for each one, and you break it!

Just ***A YEAR AGO***: The Minerva Attack

- A timing side-channel attack to get a few bits of ***k*** from the ECDSA signatures on Athena smart cards and lots of others
  - So have the smart card generate a lots of signatures
  - Then some math and brute force to get the actual ***x***
- These devices were certified…
  Including that they were supposed to resist timing attacks!
  - But, naturally, the certification doesn't actually test whether they are vulnerable to timing attacks...
- The root cause for many was a common code component: The Atmel Toolbox 00.03.11.05 library

Guess the Problem Here...

- M10.6 the TSF shall provide digital signature confirming to EC-DSA standard.
    - Secure digital signature generate
    - Secure digital signature verify
    - Fast digital signature generate **(see note*)**
    - Fast digital signature verify **(see note*)**

- M10.7 the TSF shall provide point multiplication on an elliptical curve, conforming to EC-DSA standard.
    - Secure multiply
    - Fast multiply **(see note*)**

    * The **Fast** functions of M10.3, M10.4, M10.5, M10.7, M10.8, M10.9, do not offer any DPA/SPA protection and **must not** be used for secure data.

Guess the Problem Here...

- M10.6 the TSF shall provide digital signature confirming to EC-DSA standard.
    - Secure digital signature generate
    - Secure digital signature verify
    - Fast digital signature generate **(see note*)**
    - Fast digital signature verify **(see note*)**

- M10.7 the TSF shall provide point multiplication on an elliptical curve, conforming to EC-DSA standard.
    - Secure multiply
    - Fast multiply **(see note*)**

    * The **Fast** functions of M10.3, M10.4, M10.5, M10.7, M10.8, M10.9, do not offer any DPA/SPA protection and **must not** be used for secure data.

Once Again: Bad API

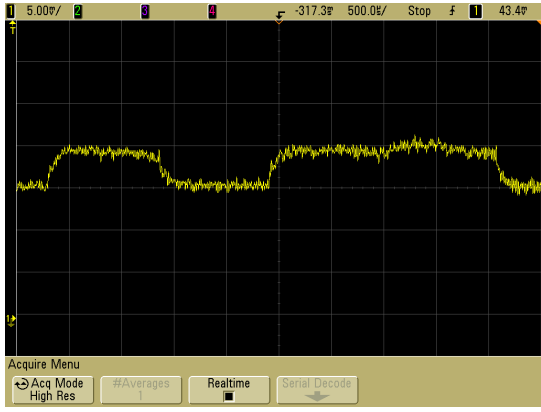
- Once again we have a case of "If you offer a programmer two ways, >50% of the time they will chose the wrong way"
  - In this case "why wouldn't I chose the fast version?"
- You have a now growing list of "red flag/canary APIs"
  - system(), raw SQL, now this example
- Keep a growing list as a "cheat sheet"
- When you get to an existing software project…
  - Search the code for these APIs
- When you start a new project
  - ***NEVER*** use the dangerous version, even if you are using it safely…
    (EG, never use system(), only execve())

Power Attacks:
The Bane of Smart Cards...

- Smart Cards are effectively small computers
  - In a handy credit-card sized package...
- Some are used to hold secrets on behalf of the cardholder
  - So really, if the person holding the card can get the secrets, 🤷‍♂️
  - Your credit card is this:
    It has cryptographic secrets to keep from a reader but not secrets that need to be kept from you
- Some are used to hold secrets ***from*** the cardholder
  - So if the user can extract the secrets, 🤦‍♂️
- The bane: Power Analysis
  - SPA == Simple Power Analysis
  - DPA == Differential Power Analysis

The Idea...

- Different operations use different amounts of power
  - EG, square vs multiply in RSA
- Hook up smart card to a reader that can measure the power
  - Have it encrypt/sign something
  - Look at the power trace to get information about hidden secrets
    - Including statistical techniques



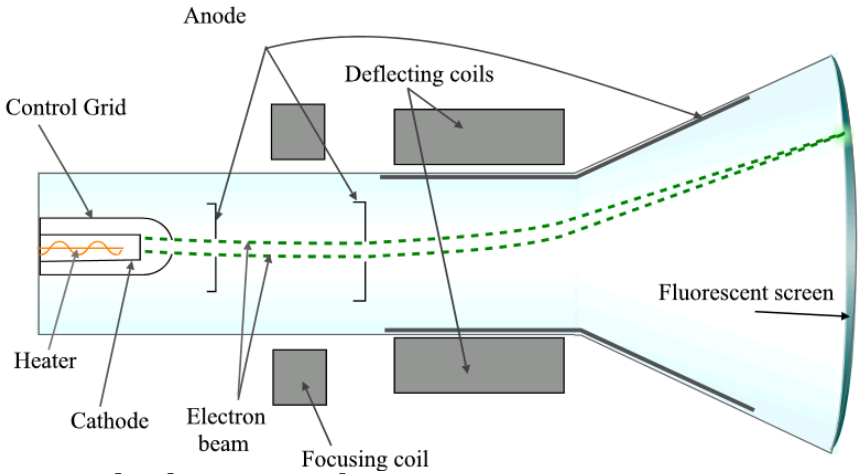
https://en.wikipedia.org/wiki/Power_analysis#/media/File:Power_attack_full.png

Countermeasures...

- Lots of work can make "simple" power analysis not work
  - But now you are using more power: Have to use the max all the time for the encryption
- Harder for more detailed differential analysis
  - Which can detect even small leaks
- If possible, punt!
  - Use your systems in a way where the person who holds the card is not your adversary!
- EG, you are building a “stored value” smart card
  - Option #1: The smart card has the value itself:
    If you tamper with the smart card, you can change the value
  - Option #2: The smart card just has an ID:
    You actually look up in the central database
    - Of course, this now means you need to be online to check the database, or have a cached copy of the database locally

Real Freaky: Elecromagnetic Emissions...

- Every time a circuit switches...
  - It leaks out some radio frequency energy
- Some sources are even easier
  - A old-school monitor paints the image with an electron beam on the screen...
- Which means it is a radio!
  - Transmitting an image of the screen!
- Cheap, too
  - $15 in 1984 for van Eck to read images off a monitor!



Solution:
The SCIF

- The US government's paranoia: The SCIF (Sensitive Compartmented Information Facility)
  - A room (or even a whole building) specifically designed for Top Secret "stuff"
  - Paranoia further enhanced by incidents like the "Project Gunman" Bug
- Multiple layers of security:
  - Physical access to the building
  - No outside electronics
    - With some caveats, fit bits can be OK depending...
  - No windows
    - Beam a laser at a window and can detect vibrations!
  - Electromagnetic shielding
    - So your cellphone wouldn't work in there anyway

And An Asside:
The Second Coolest Bug ***EVER!***

- The "Project Gunman" bug
  - https://www.cryptomuseum.com/covert/bugs/selectric/
  - "Project Gunman" was the NSA effort to ***remove*** the bug...
- In the late 70s and early 80s, the USSR bugged the electric typewriters in the US embassy!
  - Modify the mechanism that selects which character the print he magnetically tagged pieces
  - Hide a pickup & transmitter in an aluminum support rail
  - Broadcast really close in spectrum to a major TV station
- We call this a "keylogger" when done in software

Project GUNMAN exhibit at the
National Cryptologic Museum, Ft. Meade, MD,
showing the metal bar that concealed the typewriter bug

And Funky Hardware SideChannels...

- The recent Meltdown and Spectre Intel bugs...
  - Both were effectively side-channels
- The key idea:
  - You could trick the speculative execution engine to compute on memory that you don't own
  - And that computation will take a different amount of time depending on the memory contents
- So between the two, you could read past isolation barriers
  - Meltdown: Read operating system (and other) memory from user level
  - Spectre: Read in JavaScript from other parts of the web browser

How Meltdown Works...

- In a CPU, precise exceptions are hard: that is, stopping things when something "happens" at a specific instruction
- x86 actually provides two page table hardware pointers
  - One for the current user program, one for supervisor mode
  - Allows the OS to have virtual memory for the interrupt handler and other things
- Concept behind meltdown:
  - x86 allows "load whatever that memory location points to + base register"
- Do a bunch of loops that are always taken
  - Now the CPU will predict that the next time this loop is taken...
- Now do a load of memory you aren't supposed to read belonging to the OS
  - CPU guesses branch will taken, so is just going to do it speculatively.
    Only when it finally writes to a register will the exception be checked
- Now have the results of that load do a load to memory you are supposed to read
  - But dependent on what was in the memory you weren't supposed to read
- Now CPU finds that branch wasn't taken after all
  - And so nothing happens, neither the illegal load nor the "load not taken"

But Something ***Did*** Happen!

- The final "load not taken" got taken!
  - So it will be cached
- And that load was dependent on the illegal load
- So we can discover which "load not taken" got actually taken!
  - Allowing us to read memory we aren't supposed to!
- Fix involves the OS flushing the TLB and presenting a dummy OS page-table when returning to a user process
  - Greatly increasing the cost of a context switch or interrupt

Countering Meltdown and Spectre...

- Meltdown was really a bug...
  - TLB check not acted on right away
- Spectre and variants are really features of caches
  - You could train a branch-prediction buffer that you won't do it..
    Then you did it anyway
- Countering Spectre requires flushing ***all*** caches on ***every*** context switch
  - No such thing as a lightweight isolation barrier
  - This is why chrome & firefox eat ram with abandon:
    Every web origin runs in a different OS process

The Ultimate Page-Table Trick: Rowhammer

- An ***unspeakably cool*** security vulnerability…
- DRAM (unless you pay for error correcting (ECC) memory) is actually unreliable
  - Can repeatedly read/write the same location ("hammer the row" and eventually cause an error in ***some physically distinct memory location***
- Can tell the OS "I want to map this same block of memory at multiple addresses in my process…"
  - Which creates additional page table entries, lots of them. Lots and lots of them. Lots and lots and lots and lots and lots of them...
- Enter ***Rowhammer***
  - It seems all vunerabilities get named now, but this one is cool enough to deserve a name!
- Touches on virtual memory, hardware failures, and breaks security

How RowHammer Works

- Step 1: Allocate a single page of memory
- Step 2: Make the OS make a gazillion page-table entries pointing to the same page
- Step 3: Hammer the DRAM until one of those entries gets corrupted
  - Now causes that memory page to point to a set of page table entries instead
- Step 4: ***Profit***
  - Well, the ability to read and write to any physical address in the system, same difference