Innovation and Cryptoventures

Cryptographic Hashing

Campbell R. Harvey

Duke University, NBER and

Investment Strategy Advisor, Man Group, plc

January 30, 2017

Overview

Cryptographic hashing is not “encryption”

- We encrypt something with the purpose of decrypting
- A hash is a one-way function
- The only way to recover the original message is by brute force
- Hashes are also known as “Message Digests” or “Fingerprints”

Overview

Property of good cryptographic hashes

- Easy to compute – but not too easy
- Same input generates the same output
- Cannot recreate original message from the hash
- Improbable to alter message without changing hash
- Improbable for two messages to create the same hash (called a collision)

SHA-1

Secure Hash Algorithm

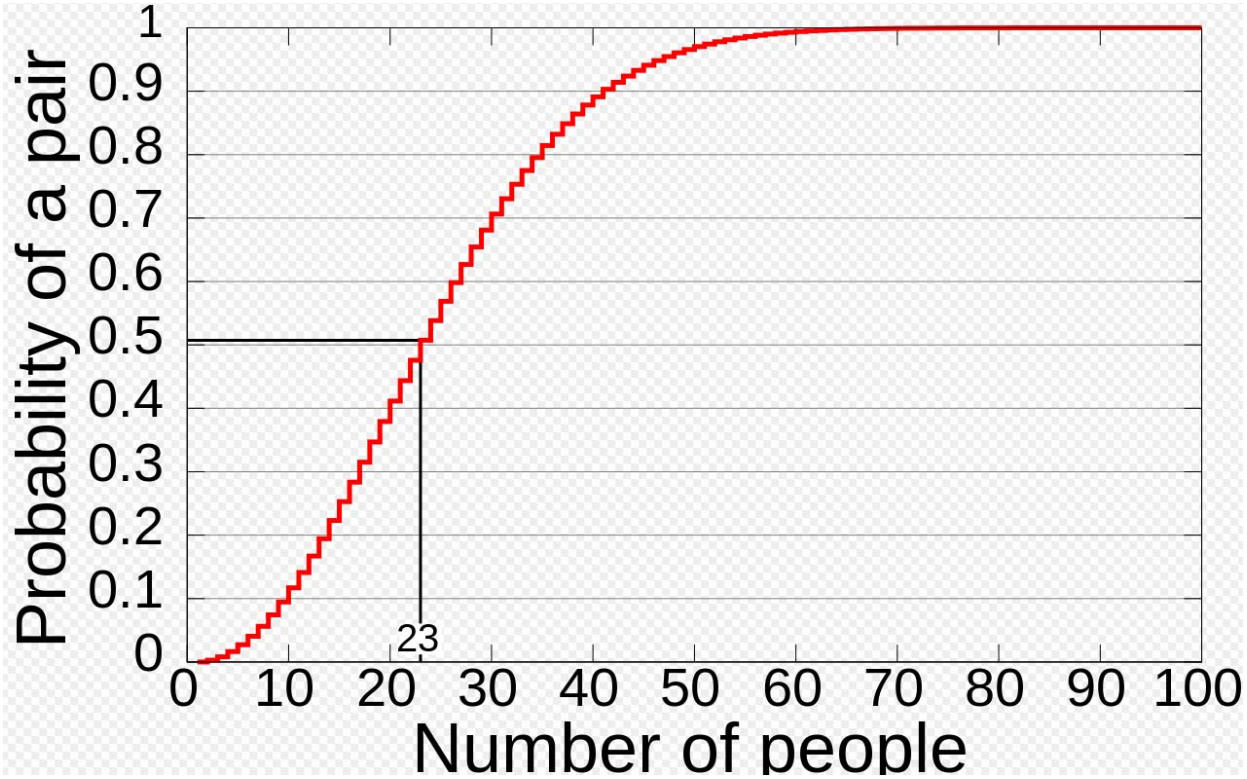
- SHA-256 is standard and its output is 256 bits
- SHA-1 has been replaced by SHA-2
- SHA-1 output is 160 bits
- We will focus on SHA-1 because it is very similar to SHA-256
- SHA-256 is preferred because it is harder to find collisions

SHA-1

Collisions

- To undo a SHA-1, you need to brute force 2^{160} combinations of bits (0s and 1s). One of these combinations will be the original message.
- However, to find a collision is a lot easier because of the birthday paradox
- How many people do you need in a classroom to have a 50% probability that two people have the same birthday?

SHA-1


Probability of a pair
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
23
0 10 20 30 40 50 60 70 80 90 100
Number of people

SHA-1

Collisions

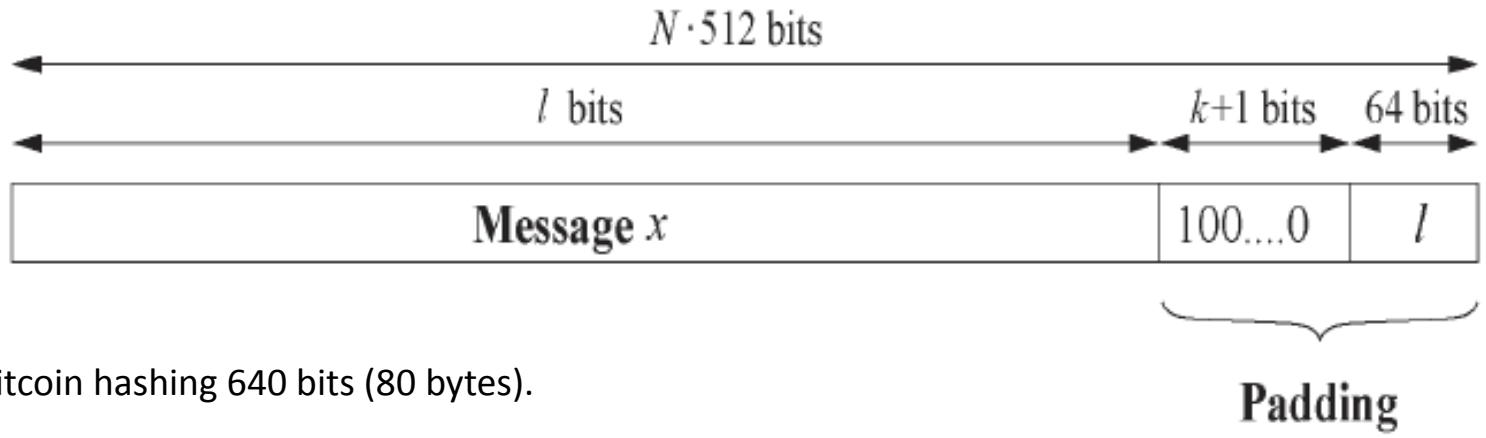
- To undo a SHA-1, you need to brute force 2^{160}
- However, to find a collision is a lot easier because of the birthday paradox
- Need approximately 2^{80} tries

SHA-1

Original message can be long. It is processed in 512 bit chunks. Let's call the first message "x" and it is 512 bits

- Message x has to be padded to fit a size of a multiple of 512 bit.
- $k \equiv 512 - 64 - 1 - l = 448 - (l + 1) \bmod 512$.

Message is padded if original is less than 512 bits



Note: Miners in bitcoin hashing 640 bits (80 bytes).

SHA-1

Inputs: one piece at a time

- 512 bits are divided up into 16 32-bits pieces, x_j j=0,…15
- "Words" $\mathbf{W}_j$ are then formed
- There are 80 of these words because there are 80 "iterations" of activity

SHA-1

Inputs: 80 “words” formed

- The first 16 are exactly the original message divided into 32 bit pieces: $\mathbf{W}_j = x_j$ for j=0 to 15
- The next 64 words are a mash using the following:

$$\mathbf{W}_j = (\mathbf{W}_{j-16} \oplus \mathbf{W}_{j-14} \oplus \mathbf{W}_{j-8} \oplus \mathbf{W}_{j-3})_{<<1} \quad \text{for j=16 to 79}$$

  - Where, $\oplus$ is “XOR”; and $_{<<}$ shifts left one bit

SHA-1

Digression: bitwise Boolean operations

- **NOT** **0**111 (number 7 in base 10) = (opposite)
  1000 (number 8)

- 010**1** (number 5) **AND** (multiply)
  001**1** (number 3) =
  000**1** (number 1)

SHA-1

Digression: bitwise boolean operations

- 0**101** (number 5)

OR 0**011** (number 3) = (if both 0, then 0, otherwise 1

0**111** (number 7)

- 0**10**1 (number 5) **XOR**

0**01**1 (number 3) = (1 if either first or second bit 1)

0**11**0 (number 6)

SHA-1

The math is not hard – it is mechanical. This is hard:

$$F^a_{\mu\nu} = \frac{\phi^a}{\phi_0} F_{\mu\nu} \tag{3.45}$$

where

$$F_{\mu\nu} = \frac{1}{e\phi_0^3}\epsilon_{abc}\phi^a\partial_\mu\phi^b\partial_\nu\phi^c + \partial_\mu A_\nu - \partial_\nu A_\mu \tag{3.46}$$

The first term in the $U(1)$ field strength contain the winding of the Higgs field. In fact from 3.41 it is clear that the field strength satisfies the Maxwell equation with a possible monopole configuration given by the Higgs field, i.e.

$$\partial_\mu F^{\mu\nu} = 0 \qquad\qquad \partial_\mu {}^*F^{\mu\nu} = \frac{4\pi}{e} J^\mu \tag{3.47}$$

The magnetic part of the $U(1)$ field strength is given by

$$B^i = \frac{\phi^a}{\phi_0} B^i_a = \frac{1}{2}\epsilon^{ijk} F_{jk} = \frac{1}{2}\epsilon^{ijk}\frac{1}{e\phi_0^3}\epsilon_{abc}\phi^a\partial_j\phi^b\partial_k\phi^c \tag{3.48}$$

and the magnetic flux is given by

$$q_m = \int_{S^2_\infty} B^i dS_i = \int d^3x \partial_i B^i = \frac{4\pi}{e}\int d^3x J^0 = \frac{4\pi}{e} Q \tag{3.49}$$

SHA-1

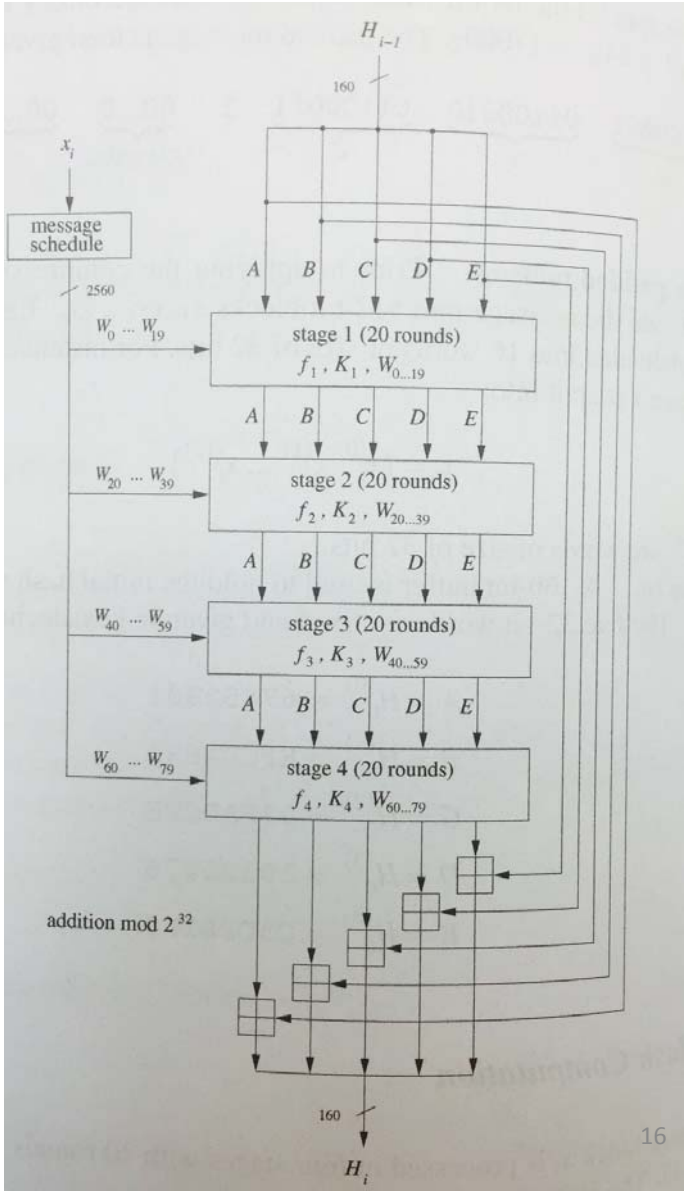
If message longer than 512 bits…

- Initial 512 bit message will hashed to 160 bits
- This 160 bit hash forms the “initial values” for the hash that will result from the next 512 bits

SHA-1

There are 4 rounds each with 20 steps

- Initial 160 bit hash is split into 5 pieces (each 32 bits): A,B,C,D,E
- Notice the modular addition of the original A-E with results of 80 iterations



Diagrams from Paar and Pelzl, Understanding Cryptography

SHA-1

Initial values of A,B,C,D,E, i.e. the starting hash, H_0

- $A=H_0(0)$ = 67452301
- $B=H_0(1)$ = EFCDAB89
- $C=H_0(2)$ = 98BADCFE
- $D=H_0(3)$ = 10325476
- $E=H_0(4)$ = C3D2E1F0

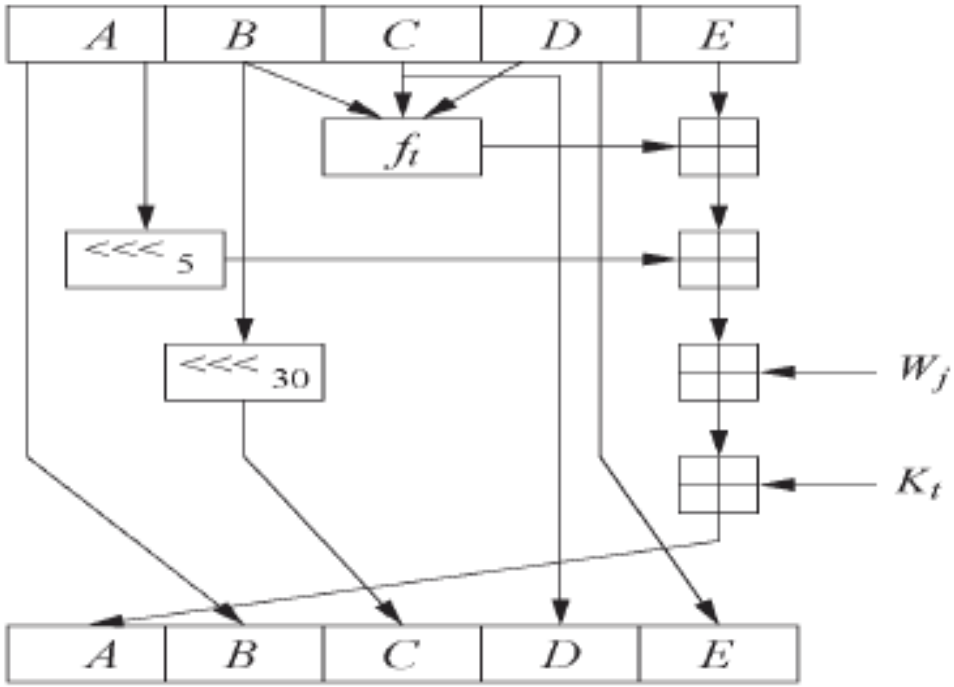
Each is 32 bits long and the above in hexadecimal form (0-9,a,b,c,d,e,f); the initial values are the same for every hash calculated

SHA-1

We now need to figure out what is happening in each of the 4 rounds. This is very convoluted as you will see.

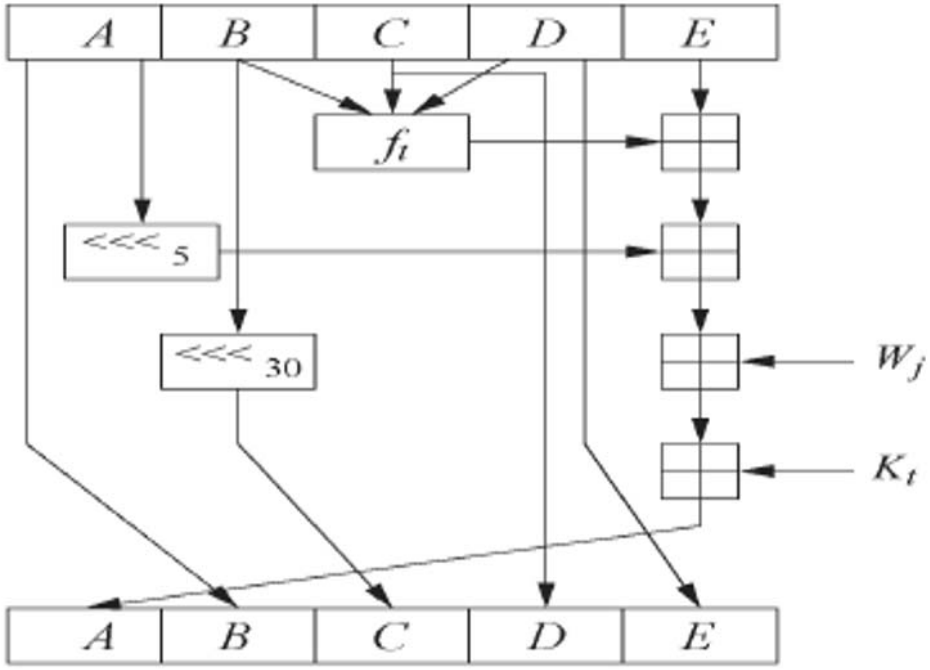
Note rectangle symbol with cross is addition

mod 2^{32}



*Think of a clock which is mod 12. If we add 3 hours past 12 noon, that is 15 – but in mod 12 we call it 3.

SHA-1



In round t, for each W_j

- New A=(E + f_t(B,C,D) + $(A)_{<<5}$ + **W_j** + K_t)
- New B=A
- New C=$(B)_{<<30}$
- New D=C
- New E=D

The function f_t changes each round as does K_t

There are a total of 80 iterations (4 rounds x 20 steps)

SHA-1

The jumbling functions and the constants:

Stage t	Round j	Constant K_t	Function f_t
1	00…19	K=5A827999	$f(B,C,D)=(B \wedge C) \vee (\overline{B} \wedge D)$
2	20…39	K=6ED9EBA1	$f(B,C,D)=B \oplus C \oplus D$
3	40…59	K=8F1BBCDC	$f(B,C,D)=(B \oplus C) \vee (B \oplus D) \vee (C \oplus D)$
4	60…79	K=CA62C1D6	$f(B,C,D)=B \oplus C \oplus D$

Note ^ = AND; v = OR; $^-$ = NOT; $\oplus$= XOR

SHA-1

The initial hash values and the constants are arbitrary but are not NSA backdoor codes:

- H_0 is a simple four bit counter (in reverse)*
- Round constants, K_t, are just square roots of 2,3,5,10**
- The slang for this is “nothing up my sleeve” – important because proposed by NSA!

*e.g. 0111 0110 0101 0100 0011 0010 0001 0000 = LE (little endian) 76543210 = BE (big endian) 67452301

**take square root in binary and use the first 32 bits (this includes both left and right of decimal)

SHA-1

The result of the 80 iterations is H_1.

The 160 bit hash is now the starting point (the new initial A,B,C,D,E) to process the second 512 bit part of the message (assuming the message is longer than 512 bits).

- Again, SHA-256 very similar but the output is 256 bits not 160 bits.
- Like SHA-1, the initial input to a SHA-2 is 512 bits

SHA-1 vs. SHA-256 some differences

- Produces 256 bits not 160 bits
- Values A-E (SHA-1, i.e. 5x32) are A-H (SHA-256, i.e. 8x32)
- Starting A-H determined by the fractional part of the square roots of the first 8 primes
- 64 rounds – not 80 iterations (4 rounds each with 20 steps)
- Constants for every step in SHA-256, fractional part of cube root of first 64 primes

Mining in Excel

It is a bad idea

- Notice there is no need for floating point operations
- Hence, we do not need CPUs to do this. Miners use a specialized ASIC technology
- However, if you wanted to mine in Excel…

  https://www.youtube.com/watch?v=UZBZPOEVyJA

Application: Passwords

Bad ideas

- Website stores your password on their site. You can tell if they do this if you can request that your password be sent back to you if you forget it.
- Website encrypts your password on their site. However, two problems. First, if the hacker gets the encryption key, all passwords are at risk. Second, many people have the same password. (Also, bad idea to store hints.)
- Website stores hash of your password. This is the industry standard but still a bad idea. Again, there are many common passwords and Google has a database of hashes associated with common passwords. Try Googling 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

Application: Passwords

Good idea

- Salted hashes. Some random data is added to your password so that the hash is unique. Even though many people have the same password, they all appear different to the attacker.
- Company keeps your username and salt. When you enter the password, the company appends the salt and then hashes. This makes an attack much more difficult even if the attacker steals the usernames and salts.

Application: File Transfer

Download a file from a website

- On the original website, there is often a “checksum” which is a hash of the file and details of the type of hashing algorithm.
- Once you download the file, you can hash it and verify that the downloaded file is exactly the same has the file on website (it has not been corrupted).

Application: File Transfer

Download a file from a website

MD5 Hash

Efficiency

Hashing very efficient: R4 does 8.6 billion hashes per second

BITMAIN Businesses Products Blog Support About Us

Antminer R4, 8.6TH/s

The most silent miner. Shipping out within 7 days of order confirmation.

1395 USD (1.573 BTC)

Weight : 9kg

Quantity :

Application: Brute forcing SHA-256

The SHA-256 hash has 2^{256}-1 different outputs. However, you should be able to get a collision with just one more possible input, i.e. the 2^{256th} must cause a collision.

Application: Brute forcing SHA-256

- “One of the consequences of the second law of thermodynamics is that a certain amount of energy is necessary to represent information.
- To record a single bit by changing the state of a system requires an amount of energy no less than kT, where T is the absolute temperature of the system and k is the Boltzman constant. (Stick with me; the physics lesson is almost over.)

Application: Brute forcing SHA-256

- "Given that k = 1.38×10^{-16} erg/°Kelvin, and that the ambient temperature of the universe is 3.2°Kelvin, an ideal computer running at 3.2°K would consume 4.4×10^{-16}ergs every time it set or cleared a bit.
- To run a computer any colder than the cosmic background radiation would require extra energy to run a heat pump.

Application: Brute forcing SHA-256

- "Now, the annual energy output of our sun is about 1.21×10^{41} ergs. This is enough to power about 2.7×10^{56} single bit changes on our ideal computer; enough state changes to put a 187-bit counter through all its values [i.e. $2.7\times10^{56} = 2^{187}$].
- If we built a Dyson sphere* around the sun and captured all its energy for 32 years, without any loss, we could power a computer to count up to 2^{192} [i.e. $32\times2.7\times10^{56} = 2^{192}$].
- Of course, it wouldn't have the energy left over to perform any useful calculations with this counter.

*Freeman Dyson, 1960. The search for artificial stellar sources of infrared radiation.

Application: Brute forcing SHA-256

- “But that's just one star, and a measly one at that. A typical supernova releases something like 10^{51} ergs. (About a hundred times as much energy would be released in the form of neutrinos, but let them go for now.)
- If all of this energy could be channeled into a single orgy of computation, a 219-bit counter could be cycled through all of its states [i.e. 2^{219}].

Application: Brute forcing SHA-256

- “These numbers have nothing to do with the technology of the devices; they are the maximums that thermodynamics will allow. And they strongly imply that brute-force attacks against 256-bit keys will be infeasible until computers are built from something other than matter and occupy something other than space.

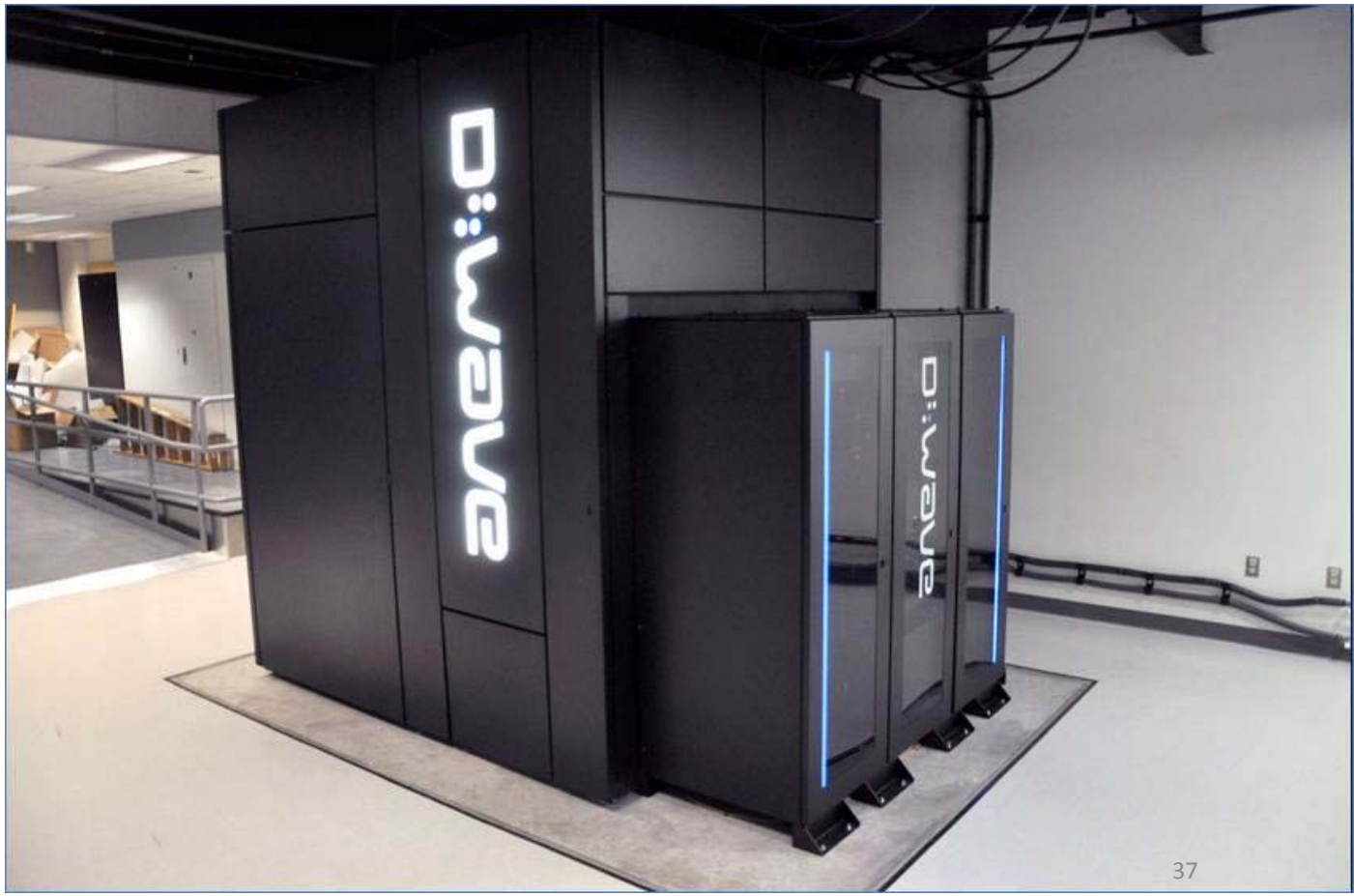
Application: Brute forcing SHA-256

- "…Even a mythical quantum computer won't be able to brute-force that large a keyspace."

From Bruce Schneiner https://www.schneier.com/blog/archives/2009/09/the_doghouse_cr.html [my brackets]

Application: Brute forcing SHA-256

- The “mythical” quantum computer in 2009 is no longer a myth.



Application: Brute forcing SHA-256

http://www.qxbranch.com