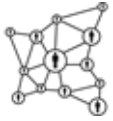
Basics of Cryptography for Blockchain (part 1)

To sum up … what is blockchain?

Network

[peer-to-peer network]



Data structure

[distributed ledger]

Protocol

[consensus mechanism].

Hashing

[Immutability of ledger]

Digital signatures

[identity of nodes]

Basics of cryptography for blockchain

Super-mini tutorial on bit encoding

Cryptographic hashing

Symmetric encryption

Asymmetric encryption

Super-mini tutorial on bit encoding

Computers, messages and bits

Computers are like “big calculators”, can only process numbers

Numbers processed by computers are expressed as sequences of bits

- A bit assumes the value 0 (low voltage) or 1 (high voltage)
- 1B (1 byte) = 8 bits, e.g. 0110 1101
- 1MB (1 Megabyte) = 1,048,576 bytes = 2^20 bytes ~ 1,000,000 bytes

Any message M (a file, an email, a Bitcoin address, a password, your final report, a facebook post, a message on Kakaotalk,…) must be converted into a sequence of bits, through some sort of encoding, in order to be processed by a computer

Two problems

How to represent a sequence of bit in a “readable” manner?

How to encode a message m into a sequence of bits?

Base-2 number system

We are used to work and think with numbers expressed using 10 digits: 0,1,…,9 (Base-10)

26 = 20 + 6 = (2 * 10^1) + (6 * 10^0)

345 = 300 + 40 + 5 = (3 * 10^2) + (4 * 10^1) + (5 * 10^0)

- … but computers only work with 0's and 1's

Any number can be represented using only 0 and 1 using the "Base-2" system (binary system)

Base-2 (binary)	**Base-10**	
0	0	0*2^0
1	1	1*2^0
10	2	2 + 0 = (1 * 2^1) + (0 * 2^0)
1011	11	8 + 0 + 2 + 1= (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0)

Binary to Decimal – do it yourself

- 11 bin = ?
- 100011 bin = ?
- 1110 bin = ?

https://www.binaryhexconverter.com/binary-to-decimal-converter

Hexadecimal (Hex or Base-16) number system

Binary numbers are long, boring, and hard to read …

Hex encoding considers a 16 digits system: 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
(Capital letters A,..,F are also sometimes used)

The prefix “0x” is often used to signal the use of hex system

Hex to decimal?
0x0=0, …,0x5 =5, ...0xa =10,…,0x11 =17 ,…, 0xb6 = 182, …, 0xcd67 = 52583
$0xb6 = (11 \times 16^1) + (6 \times 16^0) = 182$
$0xcd67 = (12 \times 16^3) + (13 \times 16^2) + (6 \times 16^1) + (7 \times 16^0) = 52583$

Hexadecimal to decimal - exercises

0x45 = ?

0xABC = ?

0xFFF1 = ?

Hexadecimal to decimal - exercises

$0x45 = ?$

$0x45 = (4 \times 16^1) + (5 \times 16^0) = 69$

$0xABC = ?$

$0xABC = (10 \times 16^2) + (11 \times 16^1) + (12 \times 16^0) = 2748$

$0xFFF1 = ?$

$0xFFF1 = (15 \times 16^3) + (15 \times 16^2) + (15 \times 16^1) + (1 \times 16^0) = 65521$

Hex to Binary?

Hex can be used as a compact way to represent binary numbers (and, therefore, any string of bits)

1 hex digit is used to represents 4 bits

- 0x 0 = 0000
- 0x 1 = 0001
- 0x 2 = 0010
- …
- 0x a = 1010
- …
- 0x f = 1111
- 0x af = 1010 1111

https://www.binaryhexconverter.com/hex-to-binary-converter

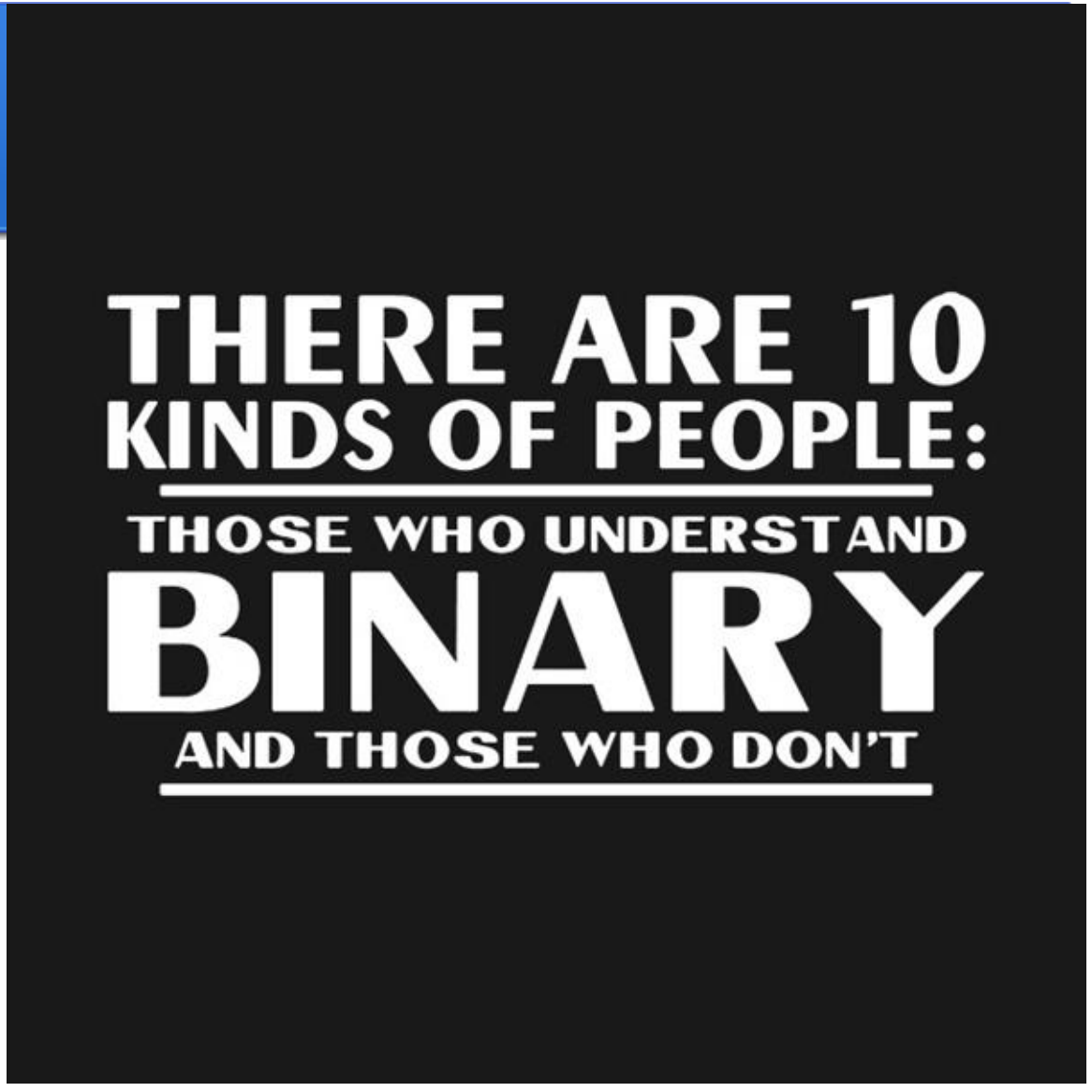
Converting decimal to other bases

14 (Base-10)	
	remainder
14:2 = 7	0
7:2 = 3	1
3:2 = 1	1
1:2 = 1	1
1110 (Base-2)	

4253 (Base-10)	
	remainder
4253:16 = 265	D (13)
265:16 = 16	9
16:16 = 1	0
1:16 = 1	1
109D (Base-16)	

27 (Base-10)	
	remainder
27:2 = 13	1
13:2 = 6	1
6:2 = 3	0
3:2 = 1	1
1:2=0	1
11011 (Base-2)	

27 (Base-10)	
	remainder
27:16 = 1	B (11)
1:16 = 0	1
1B (Base-16)	


THERE ARE 10
KINDS OF PEOPLE:
THOSE WHO UNDERSTAND
BINARY
AND THOSE WHO DON'T

Encoding messages

We know how to represent numbers as sequences of bits

Now we need a way to “encode” generic messages into numbers (which we can then represent as sequences of bits)

There are different types of encoding for different types of data (text, images, etc.)

ASCII

An encoding method for keyboard input

ASCII (American Standard Code for Information Interchange)

ASCII considers 127 characters that can be encoded

32 control characters (non-printable, like Delete, cancel, shift,…)

95 printable characters

1111111 bin = 127 (= 95 + 32), so 7 bits are enough to map each character into a binary sequence

ASCII uses 8 bits (1 byte), first bit always 0

More complex encoding have been defined, with more characters/symbols

- UTF-8 commonly used for encoding Web content
- A variety of encoding schemes for Asian languages

ASCII Table

Control characters

Dec	Bin	Oct	Hex	Char	Description
0	0000 0000	000	00	NUL	null
1	0000 0001	001	01	SOH	start of heading
2	0000 0010	002	02	STX	start of text
3	0000 0011	003	03	ETX	end of text
4	0000 0100	004	04	EOT	end of transmission
5	0000 0101	005	05	ENQ	enquiry
6	0000 0110	006	06	ACK	acknowledge
7	0000 0111	007	07	BEL	bell
8	0000 1000	010	08	BS	backspace
9	0000 1001	011	09	TAB	horizontal tab
10	0000 1010	012	0A	LF	line feed, new line
11	0000 1011	013	0B	VT	vertical tab
12	0000 1100	014	0C	FF	form feed, new page
13	0000 1101	015	0D	CR	carriage return
14	0000 1110	016	0E	SO	shift out
15	0000 1111	017	0F	SI	shift in
16	0001 0000	020	10	DLE	data link escape
17	0001 0001	021	11	DC1	device control 1
18	0001 0010	022	12	DC2	device control 2
19	0001 0011	023	13	DC3	device control 3
20	0001 0100	024	14	DC4	device control 4
21	0001 0101	025	15	NAK	negative acknowledge
22	0001 0110	026	16	SYN	synchronous idle
23	0001 0111	027	17	ETB	end of transmission block
24	0001 1000	030	18	CAN	cancel
25	0001 1001	031	19	EM	end of medium
26	0001 1010	032	1A	SUB	substitute
27	0001 1011	033	1B	ESC	escape
28	0001 1100	034	1C	FS	file separator
29	0001 1101	035	1D	GS	group separator
30	0001 1110	036	1E	RS	record separator
31	0001 1111	037	1F	US	unit separator
127	0111 1111	177	7F	DEL	delete

ASCII table

Printable characters

Dec	Bin	Oct	Hex	Char
32	0010 0000	040	20	space
33	0010 0001	041	21	!
34	0010 0010	042	22	"
35	0010 0011	043	23	#
36	0010 0100	044	24	$
37	0010 0101	045	25	%
38	0010 0110	046	26	&
39	0010 0111	047	27	'
40	0010 1000	050	28	(
41	0010 1001	051	29	)
42	0010 1010	052	2A	*
43	0010 1011	053	2B	+
44	0010 1100	054	2C	,
45	0010 1101	055	2D	-
46	0010 1110	056	2E	.
47	0010 1111	057	2F	/
48	0011 0000	060	30	0
49	0011 0001	061	31	1
50	0011 0010	062	32	2
51	0011 0011	063	33	3
52	0011 0100	064	34	4
53	0011 0101	065	35	5
54	0011 0110	066	36	6
55	0011 0111	067	37	7
56	0011 1000	070	38	8
57	0011 1001	071	39	9
58	0011 1010	072	3A	:
59	0011 1011	073	3B	;
60	0011 1100	074	3C	<
61	0011 1101	075	3D	=
62	0011 1110	076	3E	>
63	0011 1111	077	3F	?

Dec	Bin	Oct	Hex	Char
64	0100 0000	100	40	@
65	0100 0001	101	41	A
66	0100 0010	102	42	B
67	0100 0011	103	43	C
68	0100 0100	104	44	D
69	0100 0101	105	45	E
70	0100 0110	106	46	F
71	0100 0111	107	47	G
72	0100 1000	110	48	H
73	0100 1001	111	49	I
74	0100 1010	112	4A	J
75	0100 1011	113	4B	K
76	0100 1100	114	4C	L
77	0100 1101	115	4D	M
78	0100 1110	116	4E	N
79	0100 1111	117	4F	O
80	0101 0000	120	50	P
81	0101 0001	121	51	Q
82	0101 0010	122	52	R
83	0101 0011	123	53	S
84	0101 0100	124	54	T
85	0101 0101	125	55	U
86	0101 0110	126	56	V
87	0101 0111	127	57	W
88	0101 1000	130	58	X
89	0101 1001	131	59	Y
90	0101 1010	132	5A	Z
91	0101 1011	133	5B	[
92	0101 1100	134	5C	\
93	0101 1101	135	5D	]
94	0101 1110	136	5E	^
95	0101 1111	137	5F	_

Dec	Bin	Oct	Hex	Char
96	0110 0000	140	60	`
97	0110 0001	141	61	a
98	0110 0010	142	62	b
99	0110 0011	143	63	c
100	0110 0100	144	64	d
101	0110 0101	145	65	e
102	0110 0110	146	66	f
103	0110 0111	147	67	g
104	0110 1000	150	68	h
105	0110 1001	151	69	i
106	0110 1010	152	6A	j
107	0110 1011	153	6B	k
108	0110 1100	154	6C	l
109	0110 1101	155	6D	m
110	0110 1110	156	6E	n
111	0110 1111	157	6F	o
112	0111 0000	160	70	p
113	0111 0001	161	71	q
114	0111 0010	162	72	r
115	0111 0011	163	73	s
116	0111 0100	164	74	t
117	0111 0101	165	75	u
118	0111 0110	166	76	v
119	0111 0111	167	77	w
120	0111 1000	170	78	x
121	0111 1001	171	79	y
122	0111 1010	172	7A	z
123	0111 1011	173	7B	{
124	0111 1100	174	7C	|
125	0111 1101	175	7D	}
126	0111 1110	176	7E	~

ASCII encoding - exercise

Write your name as a sequence of bits using both binary and hex system using the ASCII encoding

Try with both capital and small letters

ASCII encoding - exercise

- https://www.rapidtables.com/convert/number/ascii-to-hex.html

- COMUZZI = 0x 43 4f 4d 55 5a 5a 49
- Comuzzi = 0x 43 6f 6d 75 7a 7a 69
- comuzzi = 0x 63 6f 6d 75 7a 7a 69

What did we learn thus far?

Computers can only process numbers expressed as sequences of bits (0 and 1)

We can represent sequences of bits using any numbering system (e.g., hex or Base-2)

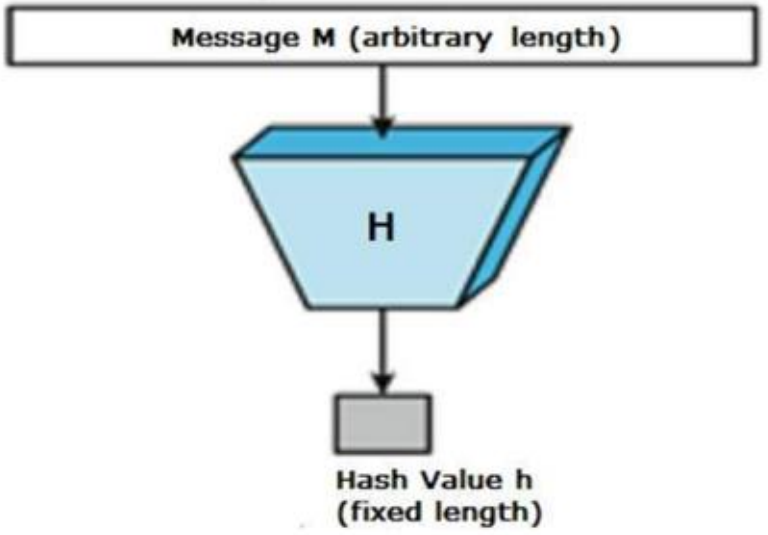
ASCII as an example scheme to encode keyboard input into a sequence of bits

Cryptographic hash functions and applications

Cryptographic hashing

A cryptographic hash function H is a mathematical function that converts any input message (M) into a hash value (h) of fixed length

h=H(M)



Features of H

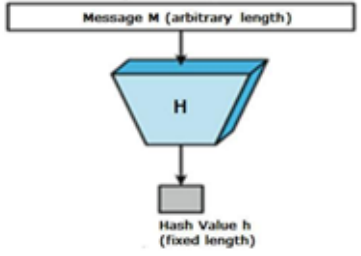
The output (h) has fixed length, no matter how large is the input

H with “n” bit output is referred to as n-bit hash function

For popular hash functions n belongs to [160 bits, 512 bits]

M generally can be much larger than h, so H can be seen as a “compression” function

h can also be seen as a (smaller) representation of M, or a “digest” of M



Properties of H

Pre-image resistance

It should be computationally hard to reverse H, that is, it should be hard to find M given h

Second pre-image resistance

Given M and h, it should be hard to find a different message N, such that H(M)=H(N)

(if attackers obtain M and h, it should be hard for them to substitute M with a new value N that looks legitimate)

Collision resistance

It should be hard to find two different message M and N that map onto the same hash value h, that is, for which H(M)=H(N)

”should be hard” = “must be impossible in practice” :-)

Collision resistance

Collision resistance

It should be hard to find two different message M and N that map onto the same hash value h, that is, for which H(M)=H(N)

When "size of h < size of message" it is impossible to guarantee collision avoidance, but we can build hash functions for which collision is "highly unlikely" in practice

"should be hard" = "must be impossible in practice" :-)

Popular cryptographic hashing functions

MD5

- Commonly used to provide assurance of integrity of transferred files (see later)
- n=128
- In 2004, collisions in MD5 were found, so no longer recommended (but still used)

SHA

- Widely used to secure Internet communications (HTTPS)
- In SHA2, n = 224, 256, 384, or 512
- No successful attacks ever reported on SHA2 (and SHA3)

Let’s play with hashing functions

MD5

- https://www.md5hashgenerator.com/
- Hash is 32 digits in base16 encoding, 32 * 4 = 128 bits

SHA 256

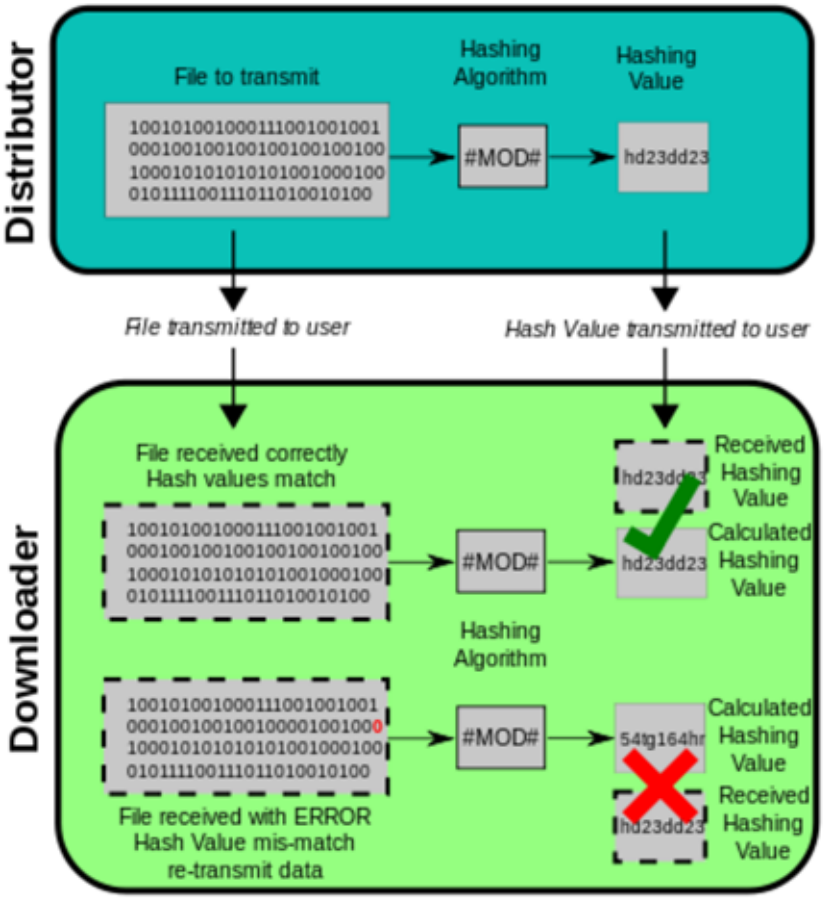
- https://xorbin.com/tools/sha256-hash-calculator
- Hash is 256 bits, 256 / 4 = 64 digits in base16

Assurance of file integrity

Associate file with its (unique) hash

- When file transmitted to a recipient
- When file made available for download on a Web site

Recipient or Web user can verify the file received or downloaded has not been corrupted, by recalculating the hash



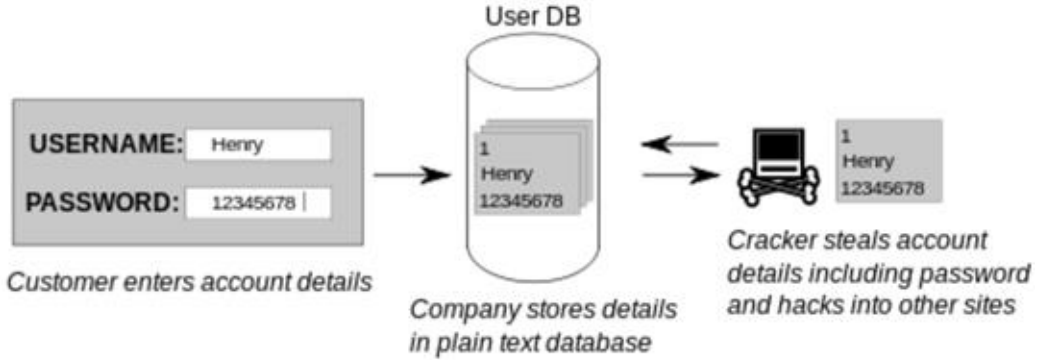
Encrypted passwords

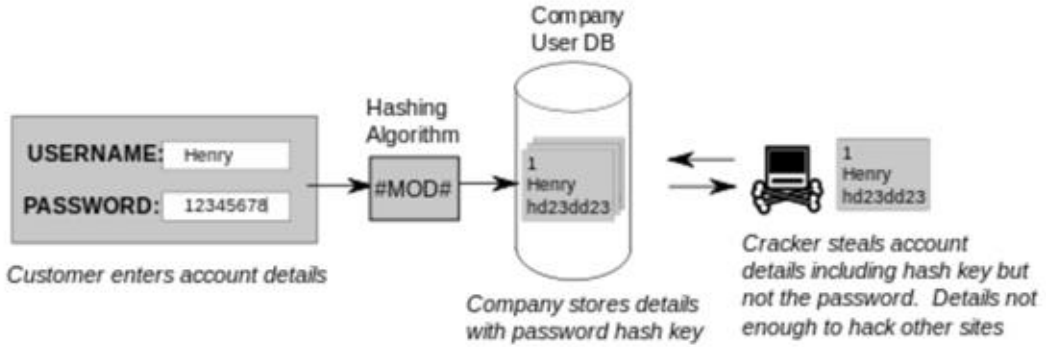
It is not desirable that providers store our passwords in plain text (can be stolen)

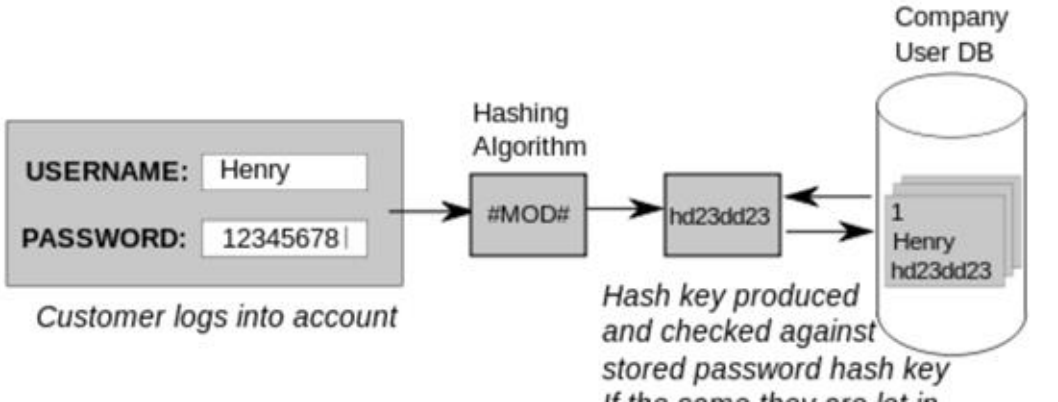
Providers store hash of passwords

–Attackers will steal, but stolen info is unusable
(H function cannot be inverted!)

Hashing functions used each time user logs in to verify login information







Digital signatures

- (we still need more)

- Cryptographic hashing combined with Asymmetric encryption (see later)

Base58-check encoding

- In order to nicely represent “big” numbers, we can use larger bases than 10 or 16

- Base64
  - 64 digits: [A-Z] (26), [a-z] (26), [0,9] (10), {/,+}

- Bitcoin uses “Base58” (actually, “Base58check”)
  - Use 58 digits
  - Same as Base64, but excluding confusing ones when read by humans (0Ol1/+)
  - Good for humans to store and exchange Bitcoin addresses etc.

Base58

Used for Bitcoin addresses, considers following 58 digits

123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

Base10	Base16	Base58
0	0x 0	1
1	0x 1	2
...	...	...
12	0x c	D
...	...	...
57	0x 39	z
58	0x 3A	21
59	0x 3B	22

Bitcoin addresses

- String of bits represented using Base58check encoding