CSBC2000

Week 2 | Class 3

Cryptography

Last Class

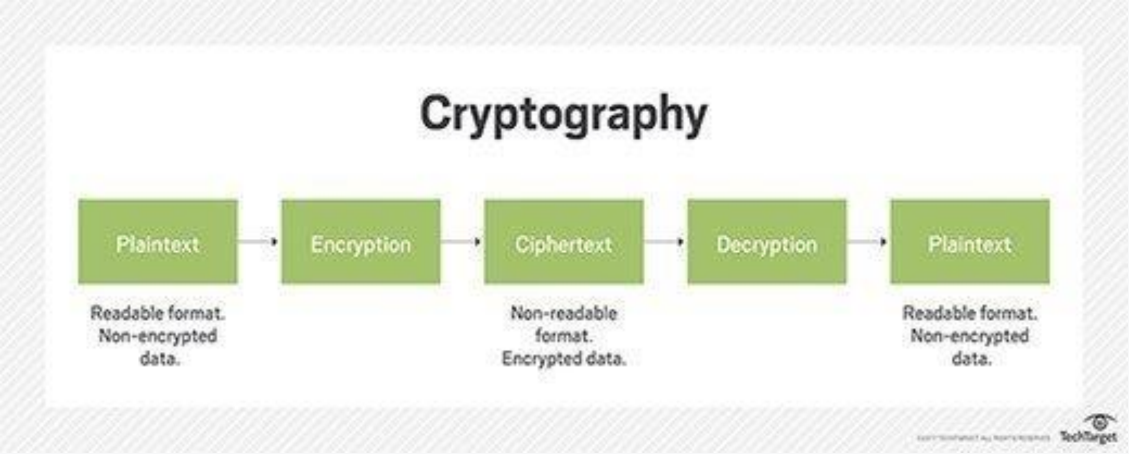
- Hashing
- Merkle Trees
- Infosec basics (CIA triad)
- How blockchains use hashing

This Class

- Confidentiality/Cryptography
- PKI
- Certificates and CAs
- Advanced Encryption

Cryptography

- Cryptography, or cryptology, is the practice and study of techniques for secure communication in the presence of third parties called adversaries
- kryptós "hidden, secret"; and γράφειν graphein, "to write", or -λογος -logos, "truth"



Impossible vs Hard

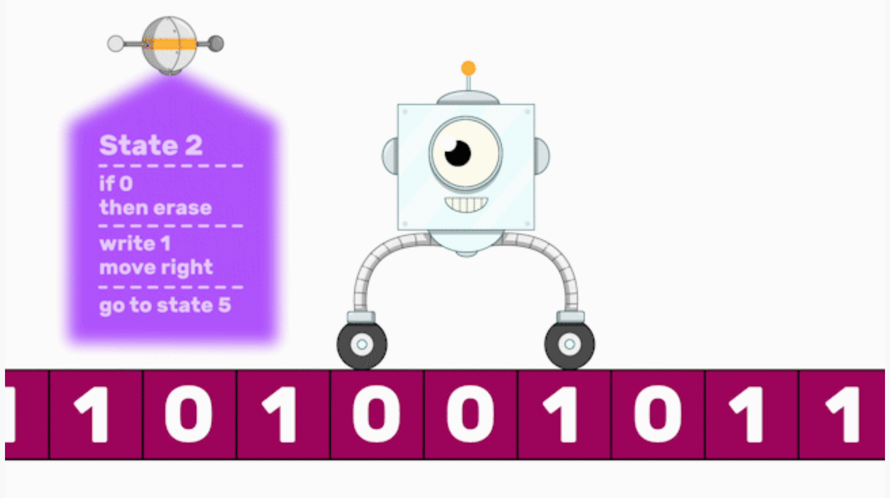
- In CS, there are formal ways to describe problems
- Algorithm = "single-purpose computer" = function
- Has inputs and an output
- There are "easy" algorithms and there are "hard" algorithms

Impossible vs Hard

- Algorithms are modelled by state machines in CS theory
- They are just like the state machines we encountered (e.g. EVM)
- Thus, given a set of inputs, 3 things can happen: success, failure, no reply (still processing)
- Hard problems are in this last category;

they take a really, really long time to

solve



Impossible vs Hard

- This property of computational hardness can be exploited for security

- In fact, we have already seen this in action!

Impossible vs Hard

- As we saw with PoW, SHA256 is 256 bits; to reverse engineer it takes 2^256 back-to-back correct guesses
- And...

$$2^{256} = 115792089237316195423570985008687907853269984665640564039457584007913129639936$$

- So it's not quite impossible, it's still finite
- But it's computationally hard

P vs NP

- Clay Mathematical Institute: "If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution."
- P = the class of problems that can be solved in **P**olynomial time (= fast)
- NP = the class of problems that can be *verified* in polynomial time
- P is a subset of NP

P vs NP

- So if it turns out that P = NP, PoW would cease to work!
- Luckily all the experts seem to think this is likely not the case
- However, there are also encryption techniques that cannot be broken even with infinite computational power

Note: Modular Arithmetic

- Arithmetic where you have a number *n* such that any mathematical operation *mod n* will "roll over" past n
- e.g. 1+7 mod 7 = 1 (=7+1)

  2*8 mod 13 = 3 (=13+3)

One-time Pad

- "Information-theoretically secure" method of encryption: No way of breaking
- All the *ciphertext* looks no different than randomly generated values
- Works by having a series of random keys
- One key is chosen and used to encrypt the message using modular addition ((a+b) mod base)
- Perfectly secure... if the pad is *truly* random

Chars as Numbers

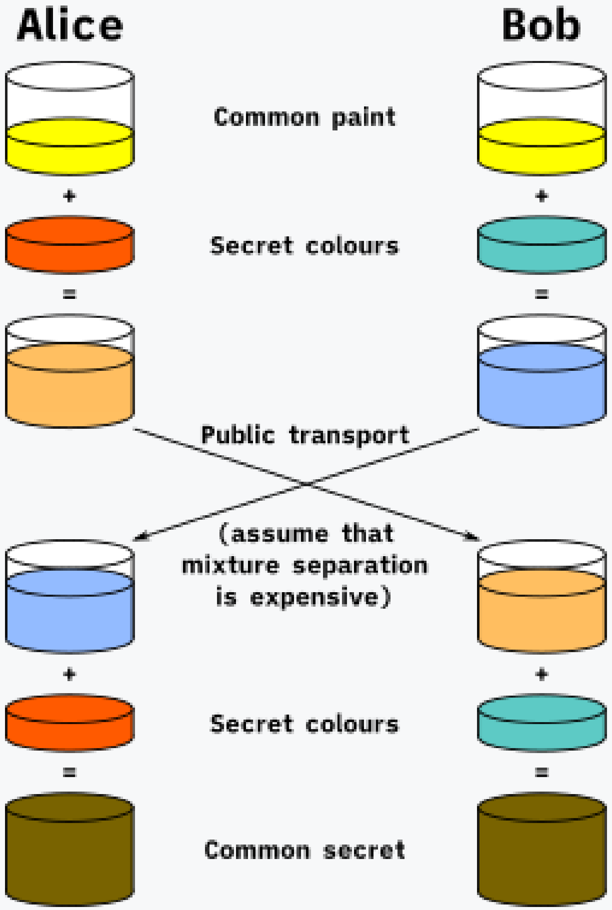
Decimal - Binary - Octal - Hex – ASCII
Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	“	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	‘	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Diffie-Hellman Key Exchange

- Ralph Merkle, Whitfield Diffie, Martin Hellman, 1976
- First publication of Public-key based cryptography
- How it works:
    - Alice and Bob agree on a shared key (can be public)
    - They each generate a secret key
    - They each "mix" their secret key and the shared key
    - They send each other the mixed result
    - They apply their secret key on the mixed result and get back their shared secret

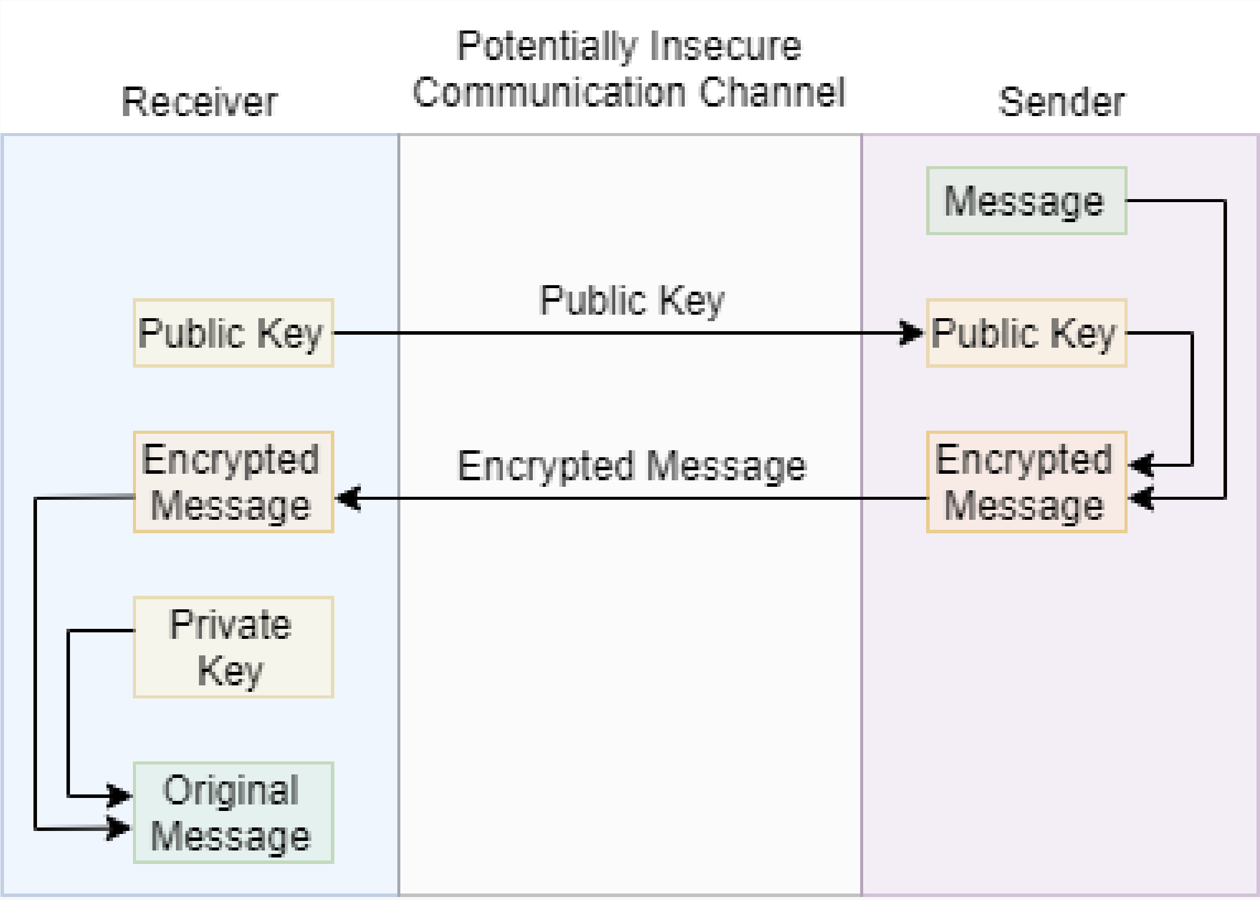
Diffie-Hellman Key Exchange



RSA

- Rivest, Shamir, Addleman, 1977
- Same concept as DH, more elaborate mathematics
- Abstractly, have a function *gen_keys()* that creates a keypair
- One key is public and can be shared with anyone
- Other key is private and should not be shared

RSA



Receiver
Potentially Insecure Communication Channel
Sender
Message
Public Key
Public Key
Public Key
Encrypted Message
Encrypted Message
Encrypted Message
Private Key
Original Message

RSA Weaknesses

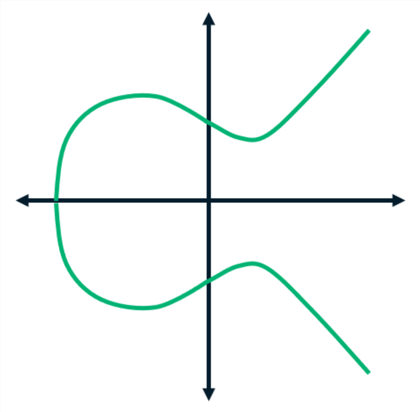
- If the same public key is used for multiple different encoding operations, it becomes harder to crack (since it is deterministic)

  - In practice, some random padding is used to make the cipher harder to crack

- Needs very large keys

ECDSA

- Uses mathematical properties of elliptic curves to generate keypairs
- "ECDSA is an asymmetric cryptography algorithm that's constructed around elliptical curves and an underlying function that's known as a “trapdoor function.” An elliptic curve represents the set of points that satisfy a mathematical equation ($y^2 = x^3 + ax + b$)"
- More complex to implement than RSA
- But is much more performant; smaller key sizes



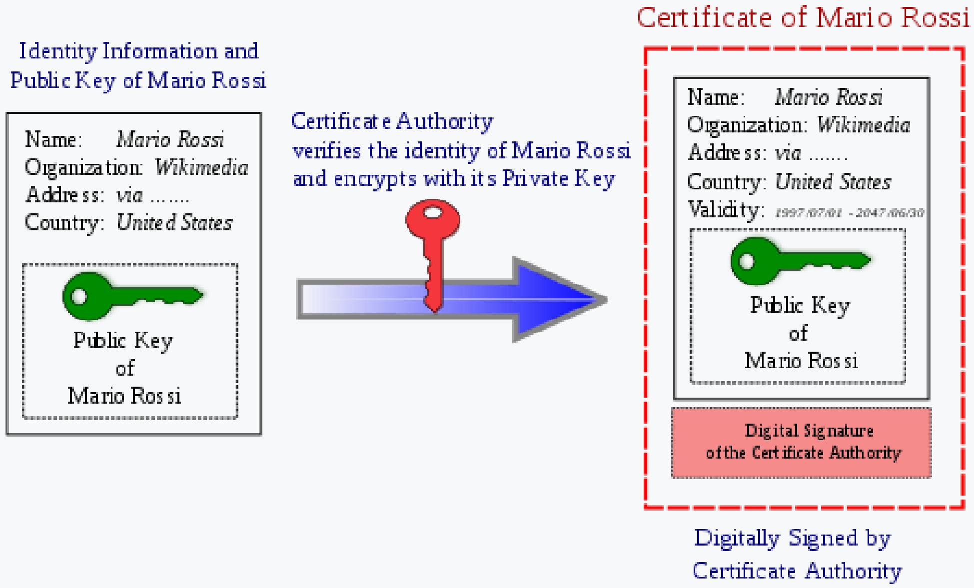
X.509

- X.509 is a standard format for public key certificates, digital documents that securely associate cryptographic key pairs with identities such as websites, individuals, or organizations
- HTTPS on your browser allows you to see this in action
- Under the hood it's just RSA/ECDSA public key with some extra information
- It is typically issued by a trusted authority known as a **Certificate Authority**
- When an X.509 certificate is signed by a **publicly trusted CA**, such as SSL.com, the certificate can be used by a third party to verify the identity of the entity presenting it

Certificate Authority

- An organization that's recognized by the internet as a trusted party
- They issue digital certificates
- A digital certificate certifies the ownership of a public key by the named subject of the certificate

Certificate Authority



Certificate Authority

- E.g. Let's Encrypt
- Me: "I own the domain abcd.com and this is my public key"
- LE: "Okay prove you own it by putting this randomly generated file somewhere in your website"
- Me: "Done, you can find it in challenge.abcd.com"
- LE: *Checks to see if the data matches*
- LE: "Okay, you've convinced me. I'll put your public key in a certificate that has my signature so you can tell others that you are verified by me"

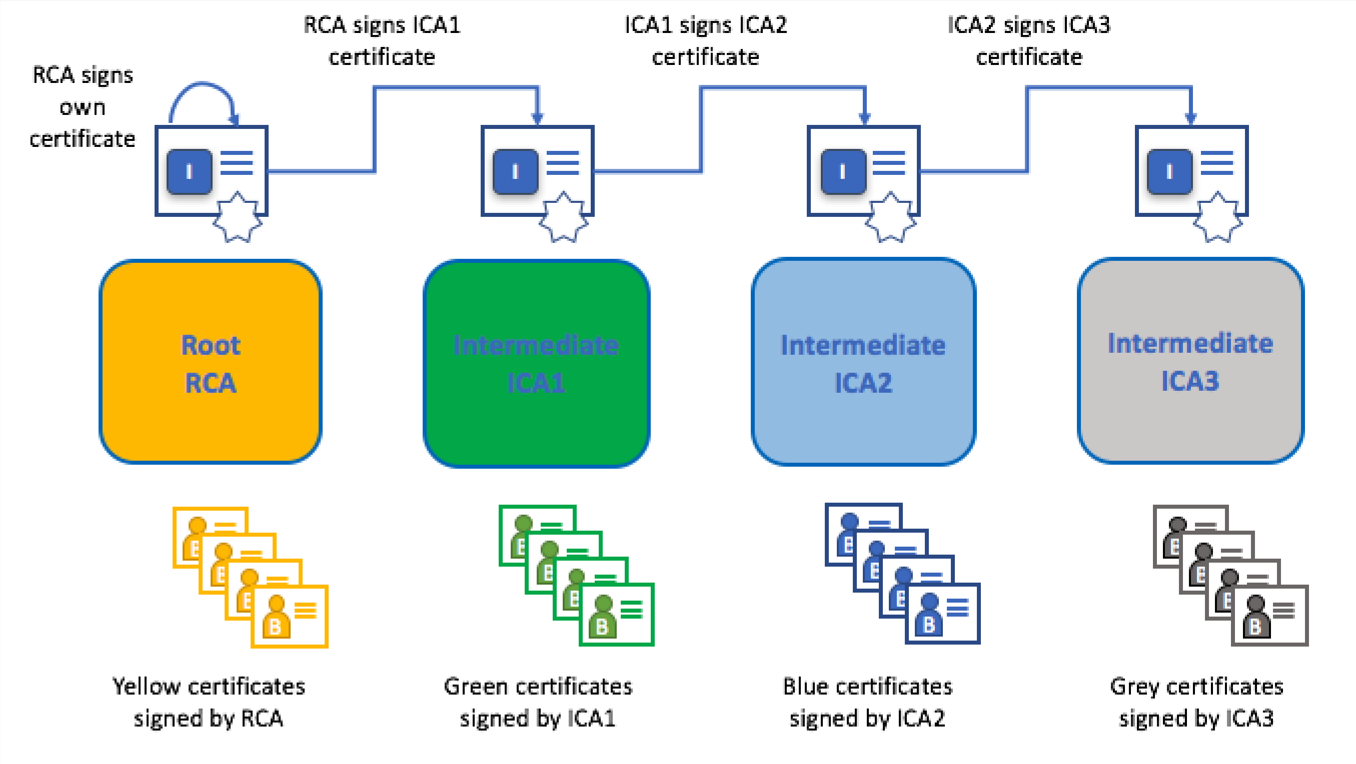
Certificate Authority

Rank	Issuer	Usage	Market share
1	IdenTrust	38.0%	51.2%
2	DigiCert	14.6%	19.7%
3	Sectigo	13.1%	17.7%
4	GoDaddy	5.1%	6.9%
5	GlobalSign	2.2%	3.0%
6	Certum	0.4%	0.7%
7	Actalis	0.2%	0.3%
8	Entrust	0.2%	0.3%
9	Secom	0.1%	0.3%
10	Let's Encrypt	0.1%	0.2%
11	Trustwave	0.1%	0.1%
12	WISeKey Group	< 0.1%	0.1%
13	StartCom	< 0.1%	0.1%
14	Network Solutions	< 0.1%	0.1%

Certificate Authority

- CAs come in two flavors: Root CAs and Intermediate CAs
- Because Root CAs have to securely distribute hundreds of millions of certificates to internet users, it makes sense to spread this process out across what are called Intermediate CAs
- These have their certificates issued by the root CA or another intermediate authority, allowing the establishment of a “chain of trust” for any certificate that is issued by any CA in the chain
- This ability to track back to the Root CA not only allows the function of CAs to scale while still providing security
- It limits the exposure of the Root CA, which, if compromised, would endanger the entire chain of trust. If an Intermediate CA is compromised, on the other hand, there will be a much smaller exposure.

Certificate Authority

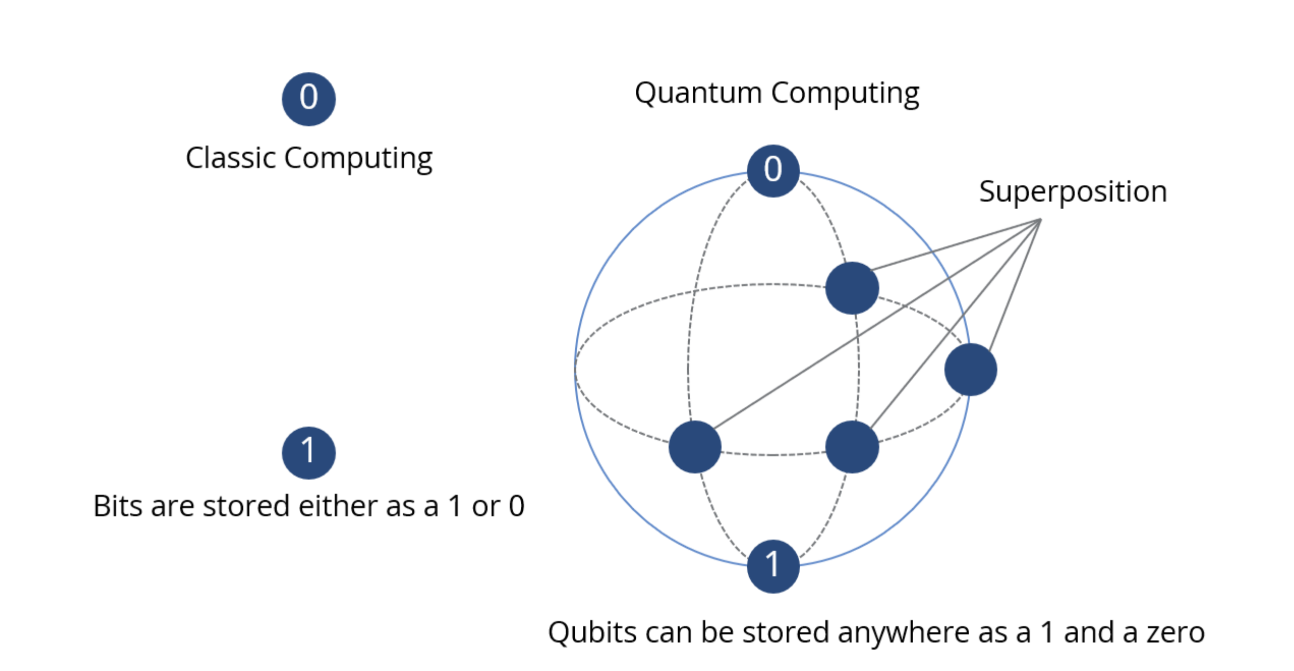


Quantum Computing

- All computing systems rely on a fundamental ability to store and manipulate information. Current computers manipulate individual bits, which store information as binary 0 and 1 states

- Quantum computers leverage quantum mechanical phenomena to manipulate information. To do this, they rely on quantum bits, or qubits.

Quantum Computing



Quantum Computation

- ?????
- Let's look at it this way; if I want to find 3 numbers that you can multiply to get the number 2095264572909, I can run a factorization algorithm that's going to explore various combos of factors
- With a quantum computer, you can explore various combos of factors *at the same time*
- This is known as superposition
- (How: complex analysis+matrices+probability)

Quantum Computation

- This is bad news for security – since pretty much all the world's security uses the fact that factorization is very difficult
- Not the end of the world though; there is *post-quantum* encryption
- Lattice-based cryptography

Homomorphic encryption

- When you have data that's encrypted, we have seen them as ciphertext so far
- Ciphertext is secure and that's good but we can't really do anything with it
- Homomorphic encryption is a form of encryption allowing one to perform calculations on encrypted data without decrypting it first
- The result of the computation is in an encrypted form, when decrypted the output is the same as if the operations had been performed on the unencrypted data

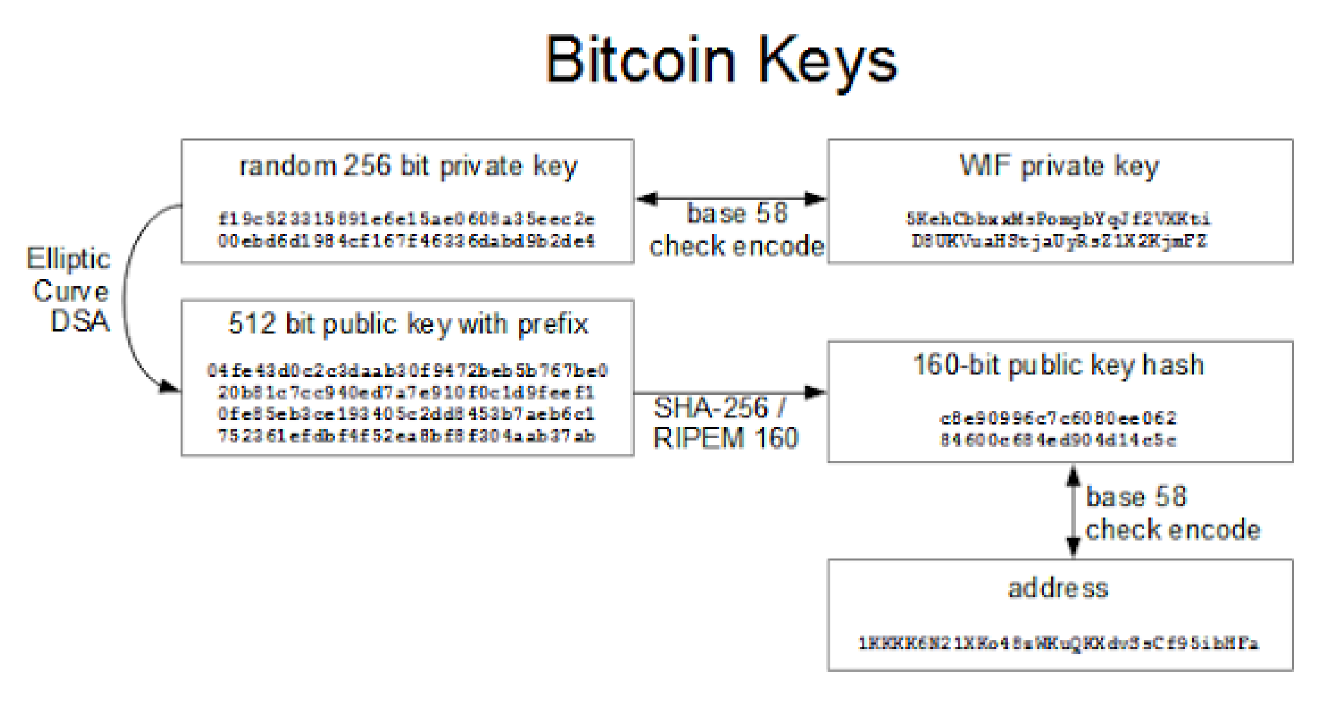
How this all relates to blockchain

- All blockchains fundamentally rely on PKI to function
- Transactions on the bitcoin blockchain occur (for example) by creating a tx skeleton that needs to be signed

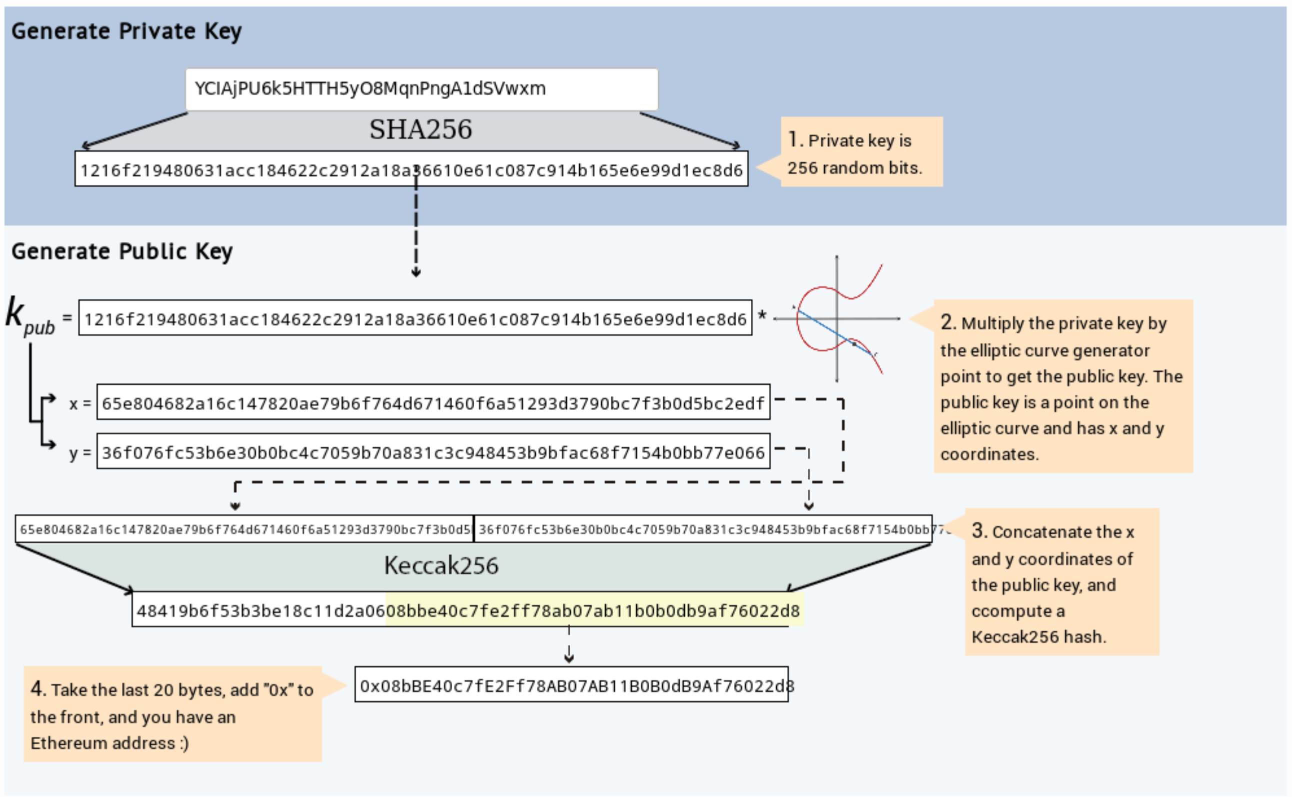
```
// get transaction skeleton
bcapi.newTX(newtx, function(err,data) {
  if(err) {
    console.log(err);
  }
  // sign transactionandadd public key
  data.pubkeys = [];
  data.signatures = data.tosign.map(function(tosign, n) {

    data.pubkeys.push(keys.getPublicKeyBuffer().toString("hex"));
    return keys.sign(new buffer.Buffer(tosign, "hex")).toDER().toString("hex");
  });
  // finally, post the transaction on the network
  bcapi.sendTX(data, printResponse);
});
```

Bitcoin Address



Ethereum Address



Homomorphic Encryption: Enigma

- Enigma uses *secure, multiparty computation* to allow user data to be stored publicly with full privacy
- "You send whatever data you want, and it runs in the black box and only returns the result. The actual data is never revealed, neither to the outside nor to the computers running the computations inside"

  \- Guy Zyskind

Homomorphic Encryption: Enigma

- Enigma encrypts data by splitting it up into pieces and randomly distributing indecipherable chunks of it to nodes
- Each node performs calculations on its discrete chunk of information before the user recombines the results to derive an unencrypted answer
- To keep track of who owns what data - and where any given data's pieces have been distributed - Enigma stores that metadata in the bitcoin blockchain

Certs: HLF

- Recall the HLF security model

- Membership Service Providers are created by an organization and allow members to use identities

Certs: HLF

- The different actors in a blockchain network include peers, orderers, client applications, administrators and more
- Each of these actors has a digital identity encapsulated in an X.509 digital certificate
- These identities really matter because they determine the exact permissions over resources and access to information that actors have in a blockchain network.

Questions/Comments?