RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID:

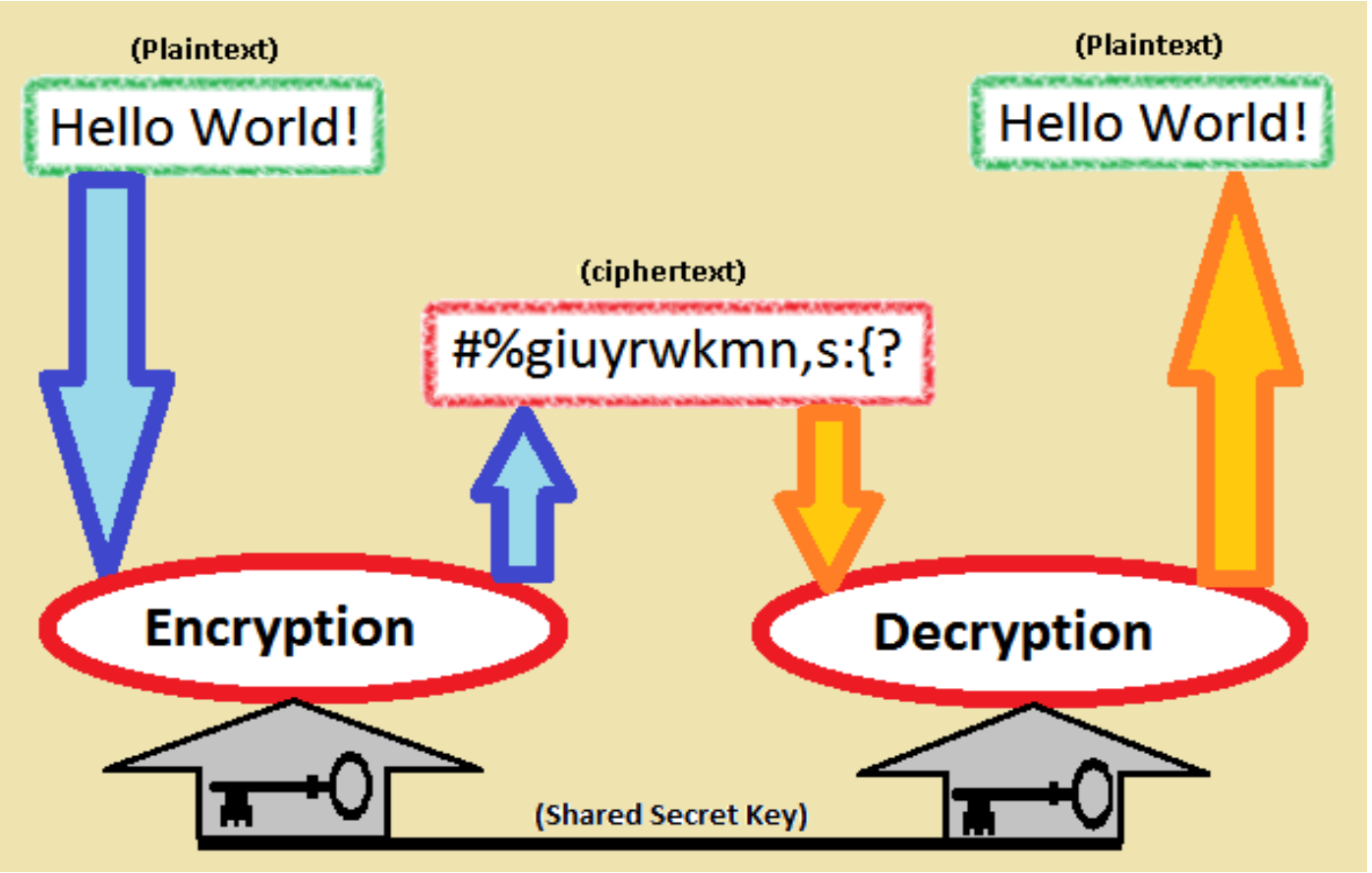
Crypto Basics: History, Applied Cryptography in IT Security Today and in the Next Year

Dan Bailey

Chief Security Architect
Carbonite
@dansinferno

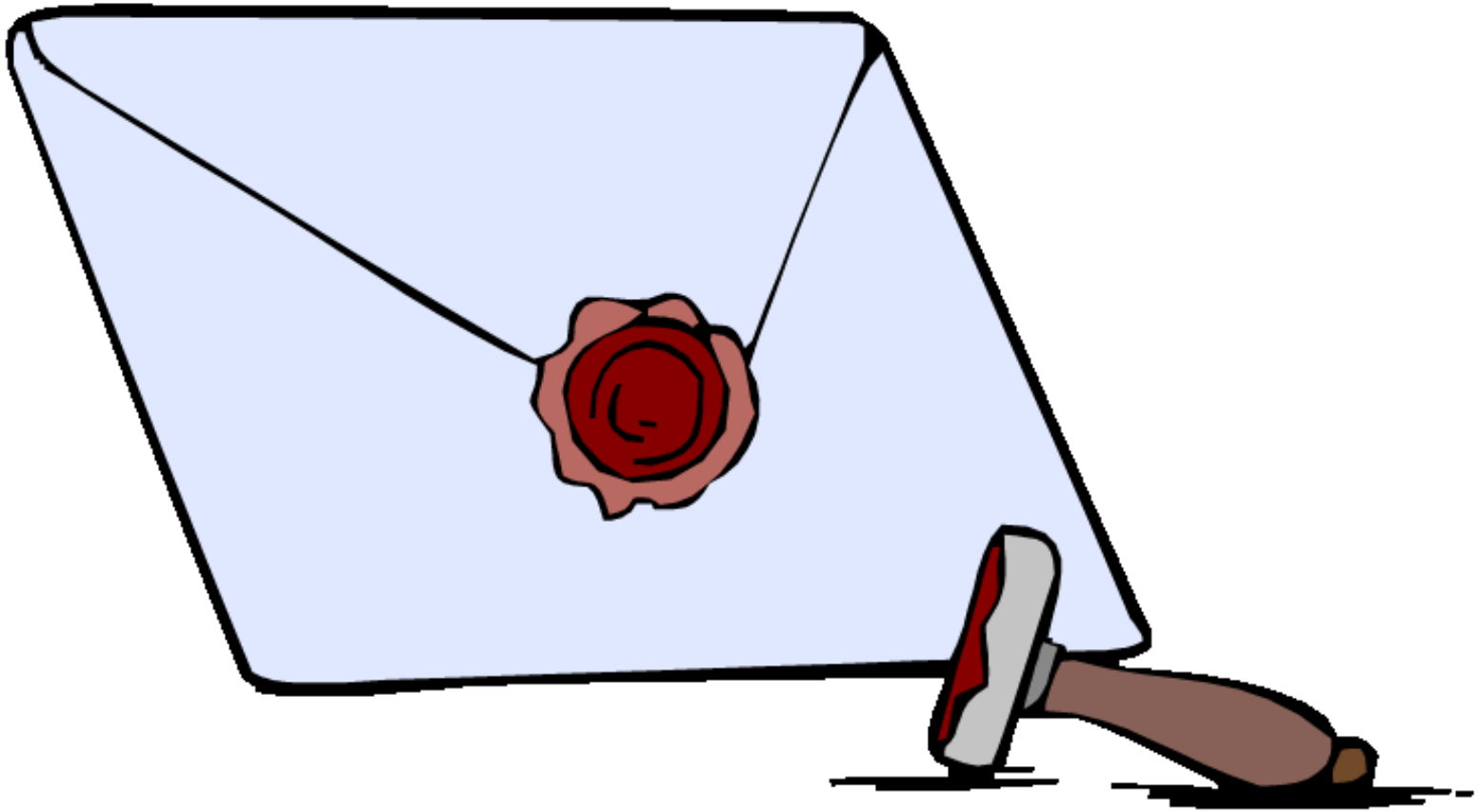
What is Applied Cryptography?

- Transforming a message so only certain recipients can read it



What is Applied Cryptography?

- Providing extra data to show who wrote the message



Symbol Substitution

Tomb of Khnumhotep II

Letter Substitution in The Kama Sutra (500 BCE?)

Plain	a	ā	i	ī	u	ū	ṛ	ṝ	ḷ	ḹ	e	ai	o	au	ṃ	ḫ	ñ	ś	ṣ	s	i	r	l	u
Cipher	kh	g	gh	ṅ	ch	j	jh	ñ	ṭh	ḍ	ḍh	ṇ	th	d	dh	n	ph	b	bh	m	y	r	l	v

LETTER SUBSTITUTION

Mlecchita-vikalpa

Don’t lose the codebook!

- If the enemy gets the codebook, you need a new codebook!
  - Modern phrase: “Security through Obscurity”
  - It’s more common than you think
- Auguste Kerckhoffs’s Assumption (1883):
  - *It’s all about the key*
  - Assume the attacker knows everything else
  - Plan on the enemy getting the codebook

Codebooks Get Keys

1870

TELEGRAPHIC

CODE,

TO ENSURE

SECRESY

IN THE

Transmission of Telegrams.

BY

ROBERT SLATER,

SECRETARY OF THE

Société du Cable Transatlantique Français, Limited.

(French Atlantic Telegraph Company).

Keyed Codebook

EXAMPLE I.

The Queen is the supreme power in the Realm.

Add any number below 25000 (say, for instance,) 5555 to the numbers opposite to those words it is desired to transmit. Where the result exceeds 25000, deduct that number, or, in other words, commence the alphabet again.

Word to be transmitted.	No. in Vocabulary.	Plus 5555.	Representing in Vocabulary.
The	22313	27868	Bounteous
Queen	18095	23650	wedge
is	12370	17925	purifying
the	22313	27868	bounteous
supreme	21953	27508	biography
power	17056	22611	transparent
in	11426	16981	posed
the	22313	27868	bounteous
Realm	18419	23974	yoke

The message being transmitted:—

Bounteous wedge purifying bounteous biography transparent posed bounteous yoke,

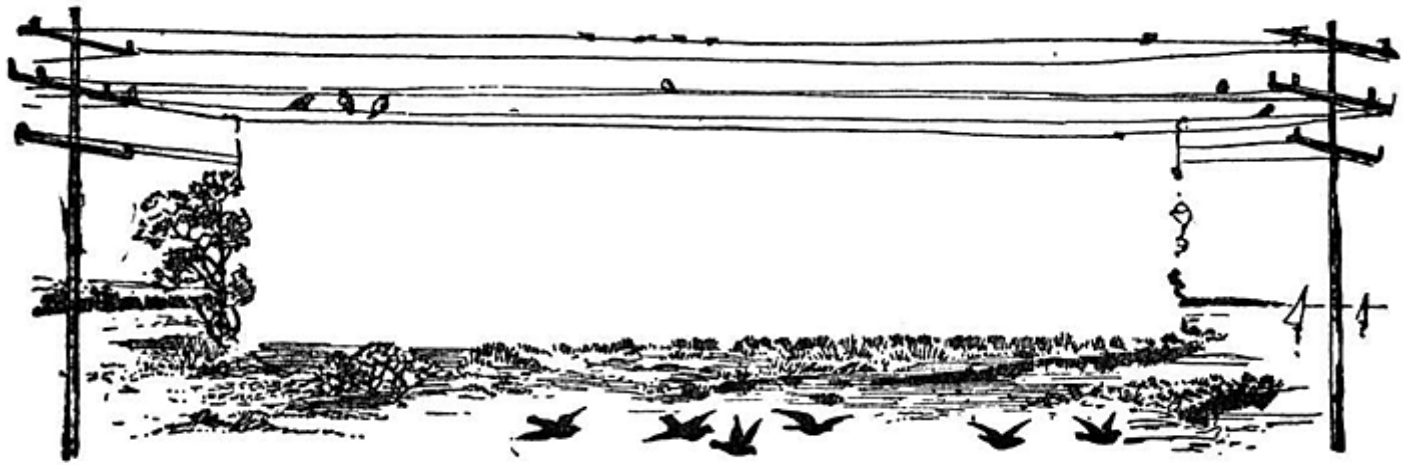
Modern encryption algorithms are fancy codebooks

- Instead of words or letters, they operate on bits or bytes

S_5		Middle 4 bits of input														
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Encryption

- AES, the “Advanced Encryption Standard” world’s most popular
- Other encryption algorithms: DES, 3DES, Blowfish
- Takes 128 bits of message, 128, 192, or 256 bits of key
- Outputs 128 bits of ciphertext



Key Lengths

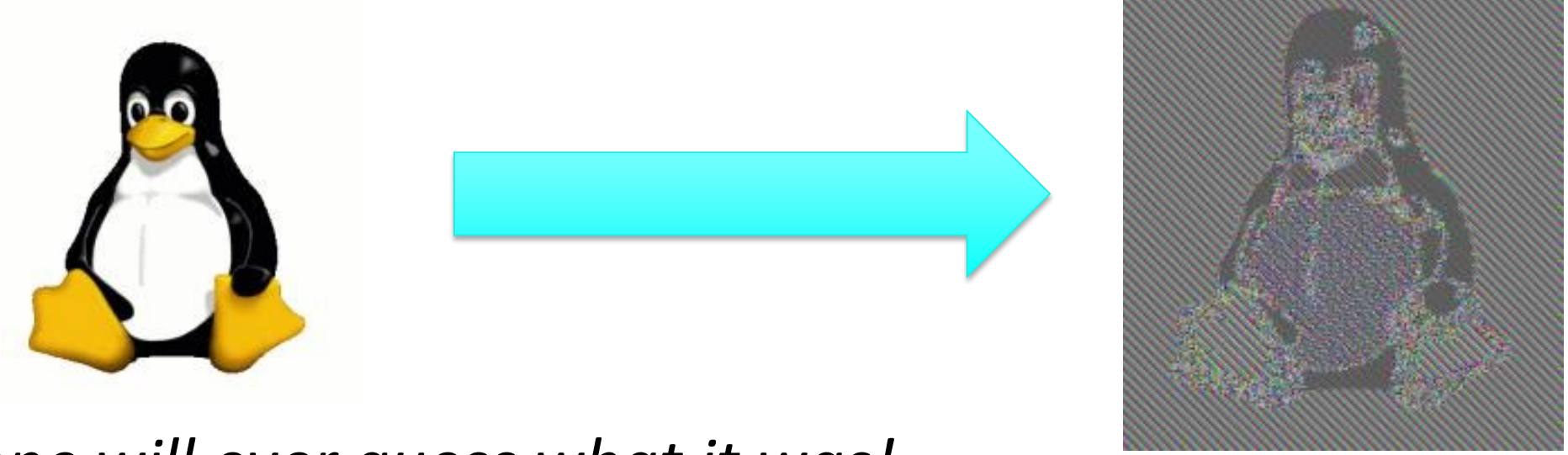
- 128 bits is (probably) enough for anyone
- All the Bitcoin miners in the world do < 56 bits/sec
  - 71 bits to go: 10^{13} years, or 10,000 aeons (=1 trillion years)
- But they'll get faster
- As far as we know, quantum computers will buy you another 21 bits/sec
  - 51 bits to go: 71 million years

128 bits ought to be enough for anybody

- Maybe you demand more!
- Feeding 128 bits of message at a time

128 bits ought to be enough for anybody

- Maybe you demand more!
- Feeding 128 bits of message at a time



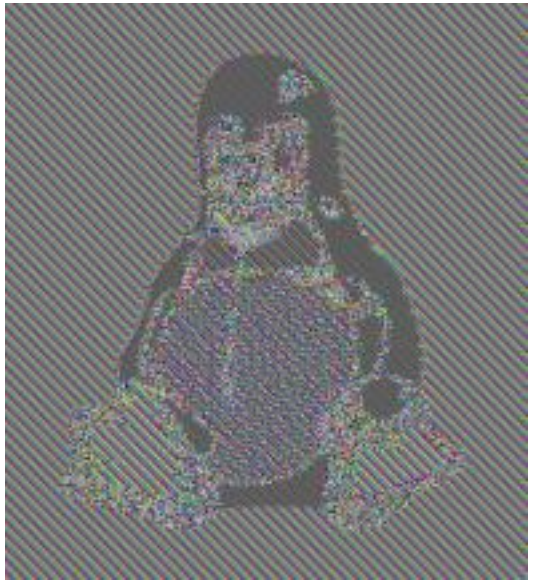
- *No one will ever guess what it was!*

Modes of Operation

- Simply feeding 128 bits of message at a time is called “Electronic Codebook”
  - Same key, same 128 bits in = same 128 bits out
- Modern cipher modes don’t have this problem
- When you hear “AES-GCM,” they mean:
  - The AES encryption algorithm
  - Galois Counter Mode to extend

Pardon the Interruption

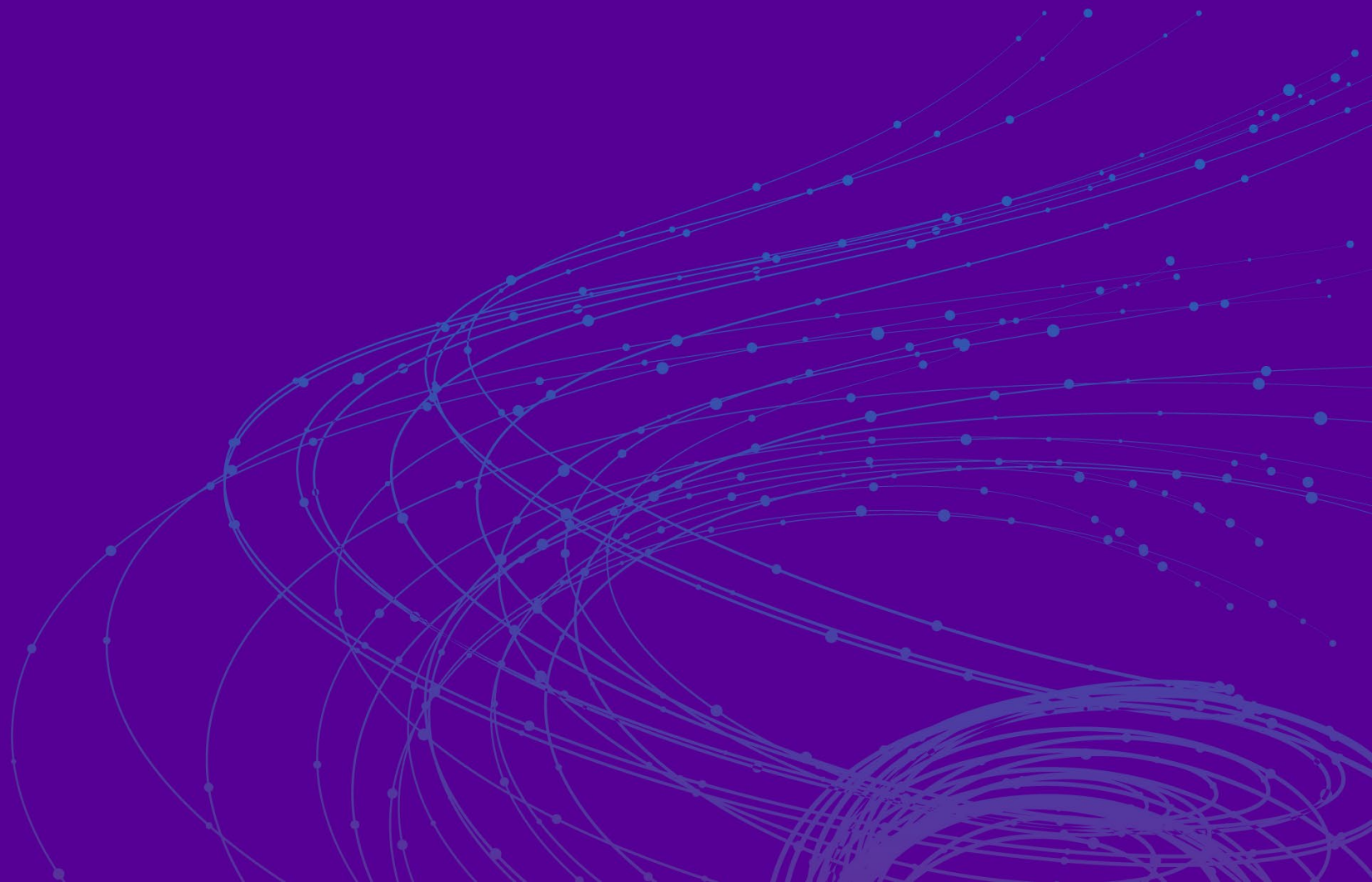
Encryption Security



- Ideally, encryption is a magical black box
  - Chooses a random replacement
- Without the key an attacker can't…
  - **Distinguish** between real and bogus replacements
  - *Even if you give her the file/message*
  - Both just look like random garbage without the key
- You can distinguish ECB mode, but you can't distinguish GCM*
  - *Unless the file/message is > 64 GB

RSAConference2019

Public Key

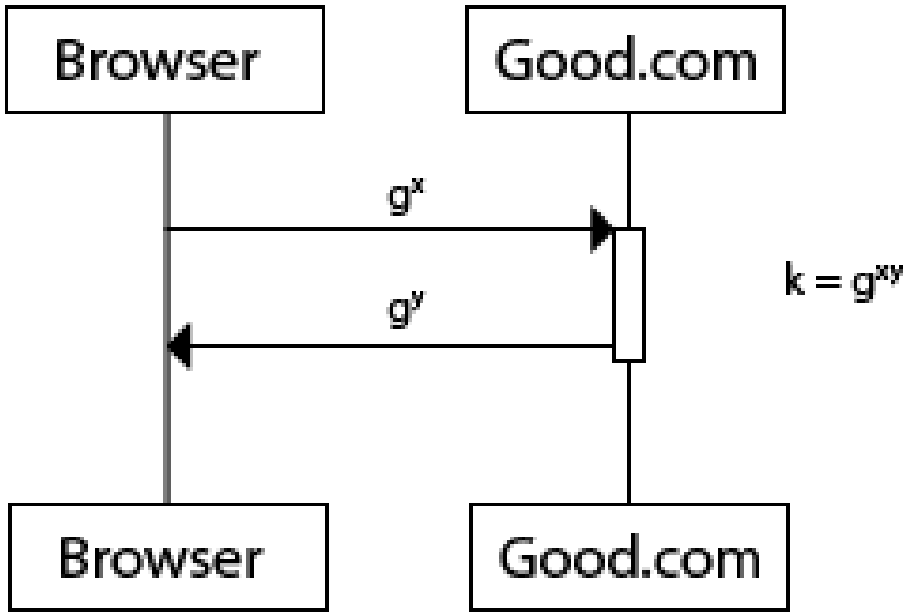


Public-Key

- There is public-key encryption in the world
  - Most famously, the Rivest-Shamir-Adleman algorithm
  - The RSA algorithm that spawned a company and a conference
- It's really a different animal
  - Not "better" or "next-gen" encryption
  - A different tool for a different job
- Public-Key is more about the possible protocols
- So, "public-key encryption" is a misleading phrase
  - You don't really encrypt your data with a public key

Key Agreement

- First, we have been talking a lot about keys
- Begging the question: how do you get these keys from client to server?



Diffie-Hellman

- *y* is Server's private key
- It computes the (long) number g^y
  - Operations use modular arithmetic
  - Just like in 1870!
- It can send g^y to the browser
  - Finding *y* given g^y is hard
- Browser does k = $(g^y)^x$
- Server does k = $(g^x)^y$

EXAMPLE I.

The Queen is the supreme power in the Realm.

Add any number below 25000 (say, for instance,) 5555 to the numbers opposite to those words it is desired to transmit. Where the result exceeds 25000, deduct that number, or, in other words, commence the alphabet again.

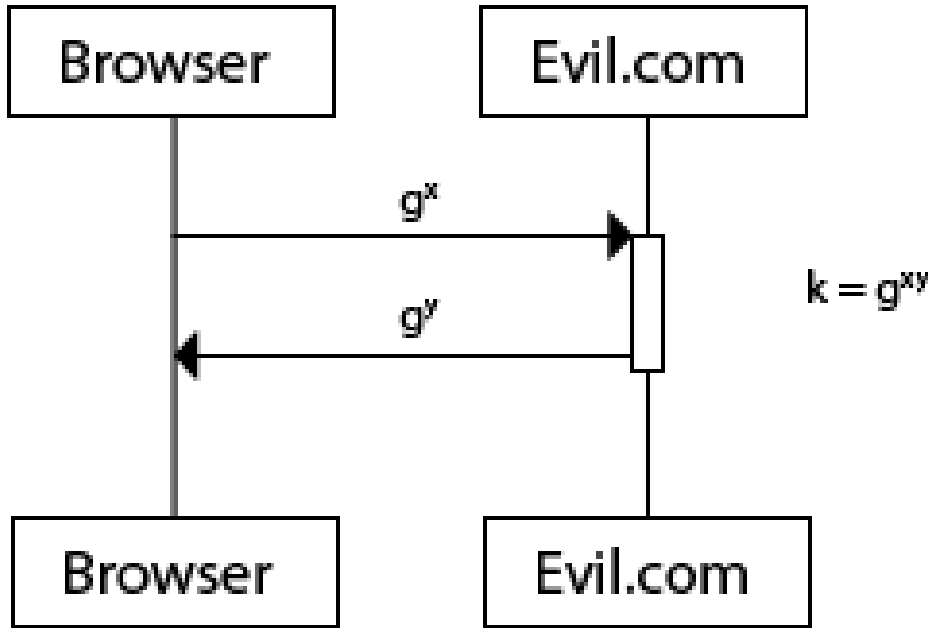
Word to be transmitted.	No. in Vocabulary.	Plus 5555.	Representing in Vocabulary.
The	22313	27868	Bounteous
Queen	18095	23650	wedge
is	12370	17925	purifying
the	22313	27868	bounteous
supreme	21953	27508	biography
power	17056	22611	transparent
in	11426	16981	posed
the	22313	27868	bounteous
Realm	18419	23974	yoke

The message being transmitted:—

Bounteous wedge purifying bounteous biography transparent posed bounteous yoke,

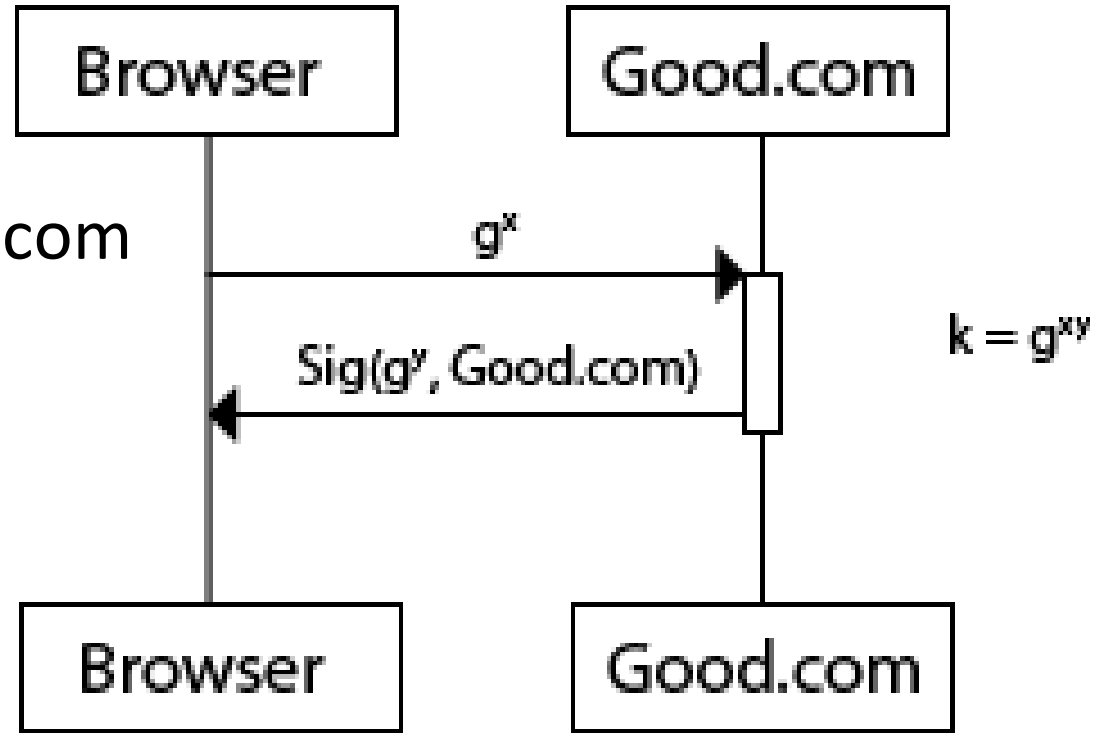
Yes, but…

- How do I know g^y came from the server and not evil.com?
  - It's great that keys can be public, but how do I know which is which?
- Digital signature algorithm like RSA
  - Takes file/message and *private* key
  - Produces a 256-byte signature



Signing Keys

- So we have another key distribution problem
- Solution: have someone vouch for Server's public key and DNS name
  - This is a certificate
  - Links public key and Good.com
- Evil.com can't modify
  - Signature won't match



Transport Layer Security (Apply it Now!)

- TLS neatly packages all this for you
  - And more besides, like freshness
  - Another reason not to reinvent the wheel
  - Don’t protect real data with your own algorithm
- Some subtlety here again: TLS 1.0 is weak (with all SSL versions)
  - Now, or when you get home, go to ssllabs.com and “test your server”
  - It will scan your company’s site looking for old versions, broken algorithms, etc.
  - If you get a bad grade, you have homework!
- Ask about your company’s plans for TLS 1.3
  - Chrome and Firefox have it, but not yet Microsoft Edge
  - iOS 12.2 has it by default

Hashing

- An algorithm like SHA-256 is like *yet another codebook*
- You give it a file/message, it gives you a short fingerprint
  - No key, but looks random anyway
- Used as an ingredient in digital signatures
- MD5, SHA-0, SHA-1 are now broken (like TLS)
  - Not for use in digital signatures or certificates
  - Still have some (cautious) uses, but you really don't want to explain it to customers

Why Hashing?

- Why do we need *yet another codebook?*
  - A different magical black box
- You can undo encryption with the key
- You can’t undo a good hash
  - Or another file/message with same fingerprint
  - …without lots of computation
- This is called Hash Inversion

Bad News

The real world is short on magic
Our magical black boxes have limits
Our humans do, too

Passwords

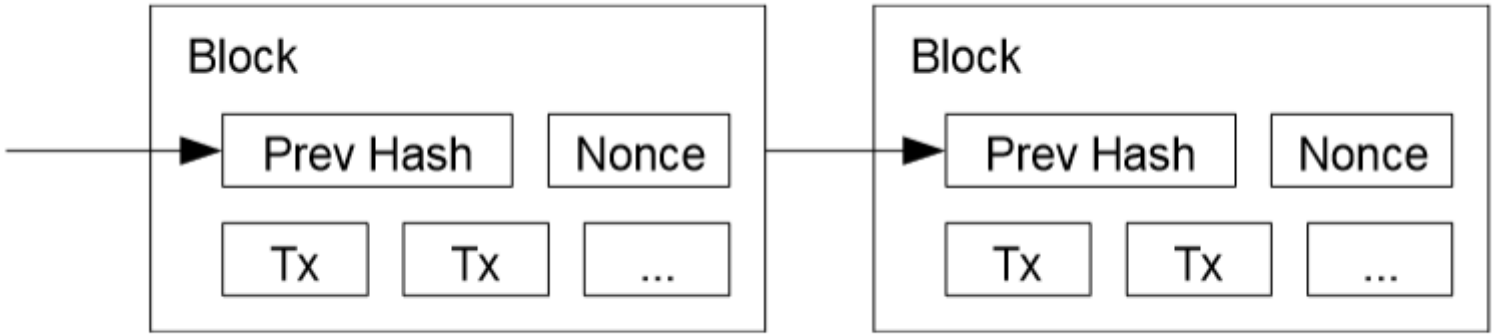
- Our black boxes actually run on real machines
- Servers get hacked and leak their username/password lists
  - Apply it Now: go to haveibeenpwned.com and see how many breaches you're in
- Countermeasure: Let's have the Server be able to check the password, but not hold it directly

Password Hashing

- Not “Password Encryption”
  - No direct undo
- Humans pick bad passwords, easy to guess
  - Fingerprints leak
  - Doesn’t require algorithm break
- Attackers have enough time to try your passwords
  - Compare 12-character random string with what users typically pick
- Apply it Now: get and use a password manager (LastPass, 1Password, RoboForm, …)

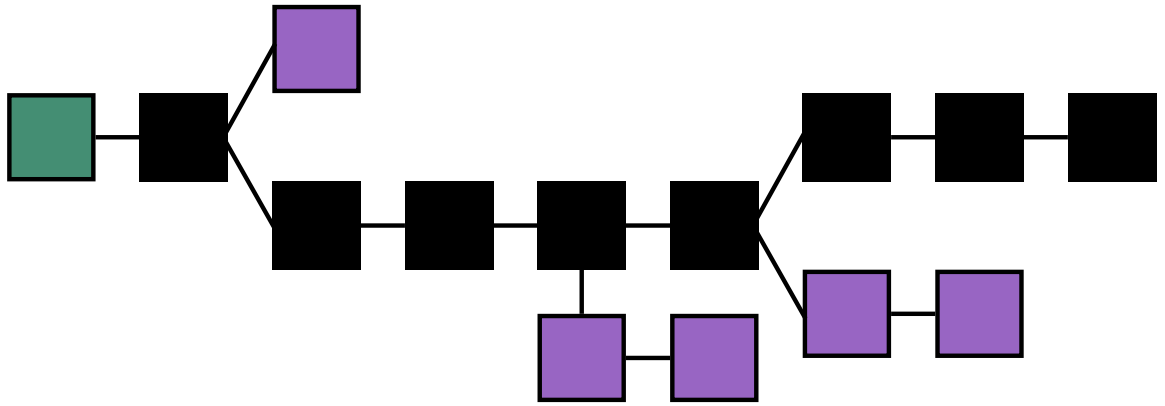
Bitcoin Mining

- Hash Inversion is impractical
  - But we can make it easier in special cases
- Search for a message whose hash starts with a number of 0s
  - Number of 0s controls the amount of work – *how much hashing*
  - The message has a particular structure



Bitcoin Mining

- Any new transaction block must “chain” to the previous
  - And start with the number of 0s
- To modify the previous block, you must *find a new nonce*
  - Redoing past work, racing the other miners
  - The longest chain is the correct chain



Conclusion – Apply It Now!

- Use TLS, don't roll your own
  - Check the health of your organization's certs
  - TLS 1.3 will become more common in the coming year
- Use a password manager
  - Check haveibeenpwned.com
- We're not even scratching the surface
  - Post-quantum public key anyone?
  - RSA-3072 takes fewer qubits than ECC-256