CS-340 Introduction to Computer Networking

Lecture 15: Encryption and Anonymity

Steve Tarzia

Last Lecture: Ethernet Link Layer

- Link layer handles error detection and correction: **Parity**, **Checksum**, and Cyclic Redandancy Check (**CRC**).
- Ethernet adds **MAC addresses** to identify src/dst on a shared link.
  - **ARP** uses Ethernet broadcast to find *IP address* → *MAC address* mapping
- **DHCP** requests are sent by Ethernet broadcast (to FF:FF:FF:FF:FF:FF)
- Old Ethernet **hubs** broadcasted data to all ports.
- Ethernet **switches** learn/remember which MAC addresses are reachable on each port and relay traffic only to the appropriate ports.
  - Reduce broadcast traffic and eliminate collisions.
- **VLANs** create multiple isolated LANs/subnets on one switch.
- **Data Centers** & **Supercomputers** demand fast local networks.

Network Security Overview

Goals

How would you define "network security?"

STOP and THINK

- **Confidentiality:**
  Keep message private/secret.
- **Reliability:**
  Deliver message to the intended recipient. Don't drop it!
- **Integrity:**
  Deliver message without alteration.
- **Authentication:**
  Verify the identity of the endpoint I'm communicating with.
- **Anonymity:**
  Conceal relationship between two Internet hosts. Conceal who I am talking to.

Challenges

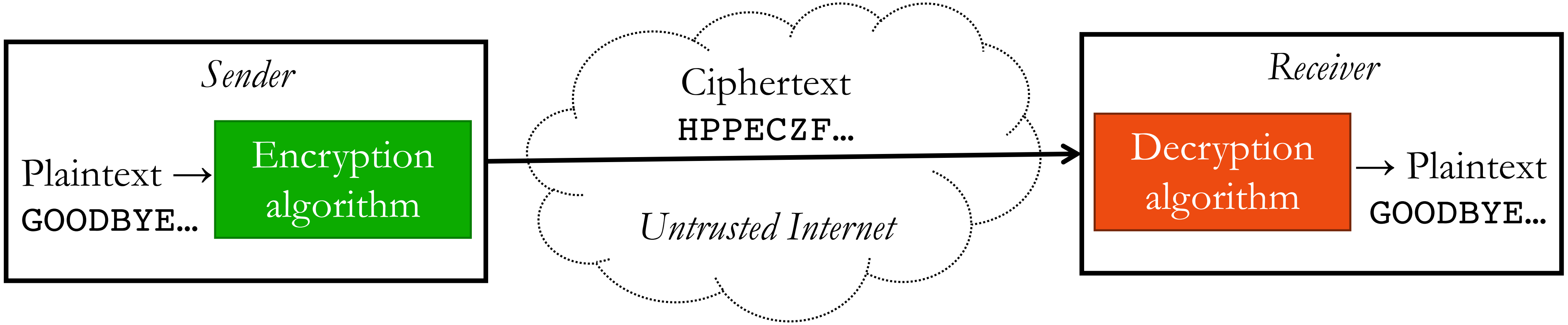
Why is the Internet difficult to secure?

STOP and THINK

- Internet messages travel through untrusted, *3rd-party links and routers.*
  - Routers must see all bits of the packet to copy it to the next hop.
- Local networks use *shared media.*
  - Nearby Ethernet or WiFi hosts can often see your packets.
- *DNS* may be *poisoned*, sending you to the wrong IP address.
- Individual attackers, corporations, and governments all wish to violate your network security.
- IP packets must be *addressed* to reach destination.

Encryption provides confidentiality

- **Plaintext** or **cleartext** is the message you wish to send.
  - For example, `"GOODBYE AND THX FOR ALL THE FISH"`
- **Ciphertext** is the encrypted version of the message, which should have no meaning to an intruder who observes the message before delivery.
  - For example, `"HPPECZF BOE UIY GPS BMM UIF GJTI"`
- Sender uses an encryption algorithm to produce ciphertext from plaintext.
- Receiver uses a decryption algorithm to recover plaintext from ciphertext.
- A **cipher** is a pair of compatible encryption & decryption algorithms.



Strong encryption

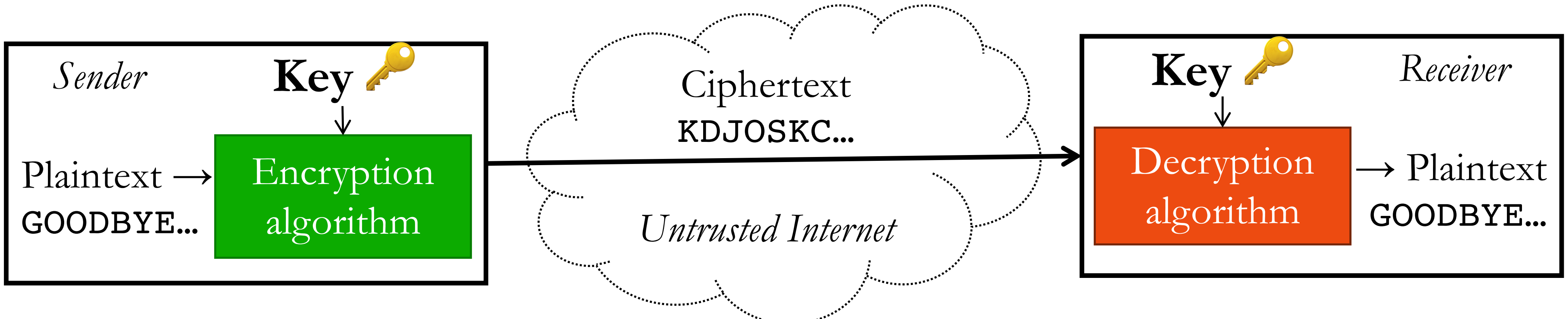
- Our example used a very simple, weak cipher.

  Plaintext: `"GOODBYE AND THX FOR ALL THE FISH"`
  Ciphertext: `"HPPECZF BOE UIY GPS BMM UIF GJTI"`

  - Just replace each letter with the next letter in the alphabet (A→B, B→C, ...)
- Decryption algorithm must be known only to the receiver
- Decryption algorithm must be different for each connection, otherwise the receiver would be able to decode senders messages to other destinations.
- Details of Cryptography are beyond the scope of this course, but we'll see how *cryptographic primitives* can be used to secure networks.
  - CS-396 Intro to Cryptography covers this in depth.

Parameterized encryption algorithms

- In practice, cryptographic algorithms are standardized, but use a unique **erncryption key** for each connection. (Also called the **session key**.)
- The key alters the algorithm's behavior and changes the output.
- Allows a single, standard algorithm to be used for many connections.
  - Just choose a different parameter value (key value) for each connection.
- Eliminates the need to invent a new new, secure encryption scheme for each new connection, just choose a **random number** to serve as the key.
  - Receiver must know the exact key used by the sender in order to decrypt.

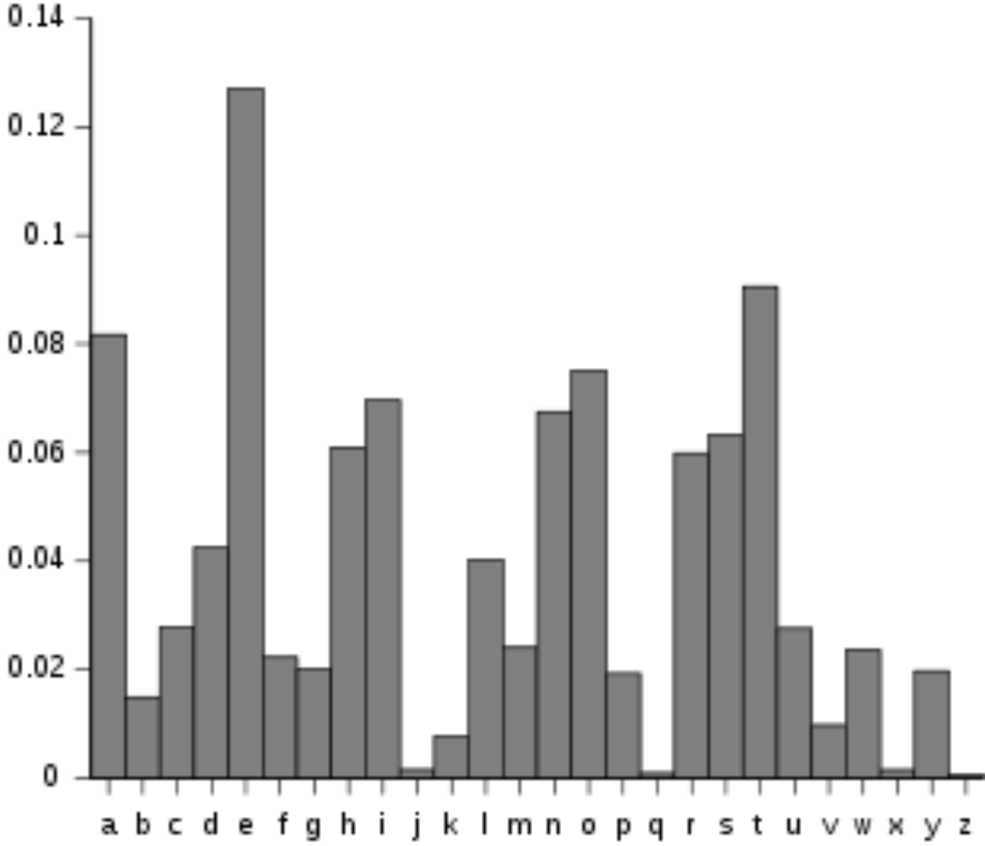


Caesar Cipher is a simple parameterized cipher

- It's a generalization of the first cipher we saw.
  - Replace each letter A-Z with the letter that is **k** places later in the alphabet (wrapping around when you reach the end).
  - It's a *family* of 26 related ciphers. The parameter **k** is the encryption key.
- With key k=1: `"GOODBYE"` → `"HPPECZF"`
- With key k=13: `"GOODBYE"` → `"TBBQOLR"`
- Receiver must know the key to decode correctly.
  - Using the wrong key, k=1, `"TBBQOLR"` → `"SAAPNKQ"`
- Try it out: https://cryptii.com/pipes/caesar-cipher

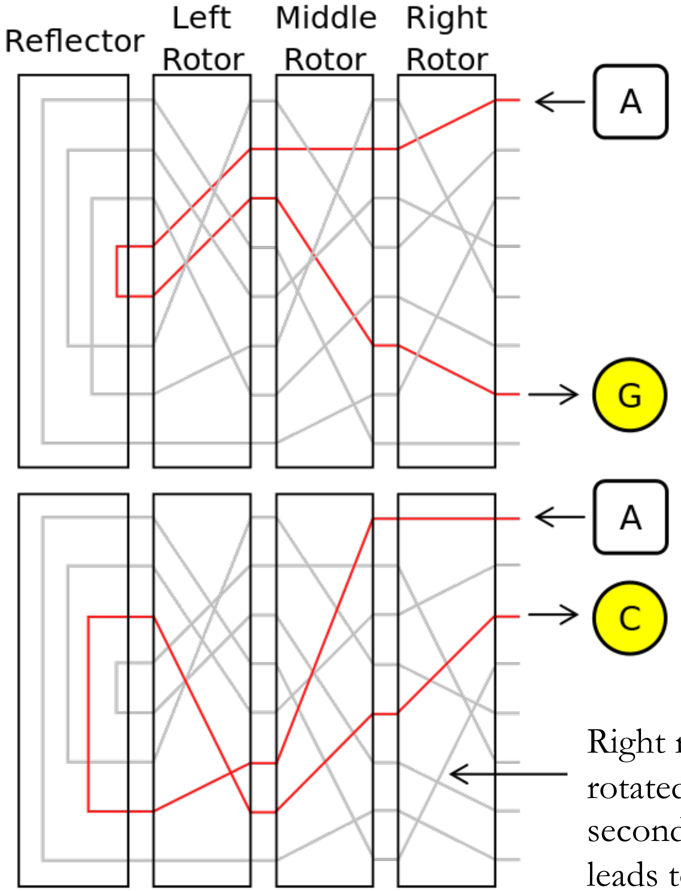
Breaking weak encryption

- Caesar Cipher has just 26 possible keys, so we can do a **brute force attack**:
  - Try decrypting with all possible keys and stop when you get something that looks like valid information (eg., contains English).
- Can also do a **statistical attack**:
  - Letter frequencies in English are well known.
  - Letter pair frequencies, etc., can also be used.
  - Any cipher that just substitutes one letter for another is susceptible.



Modern cryptography began before computers

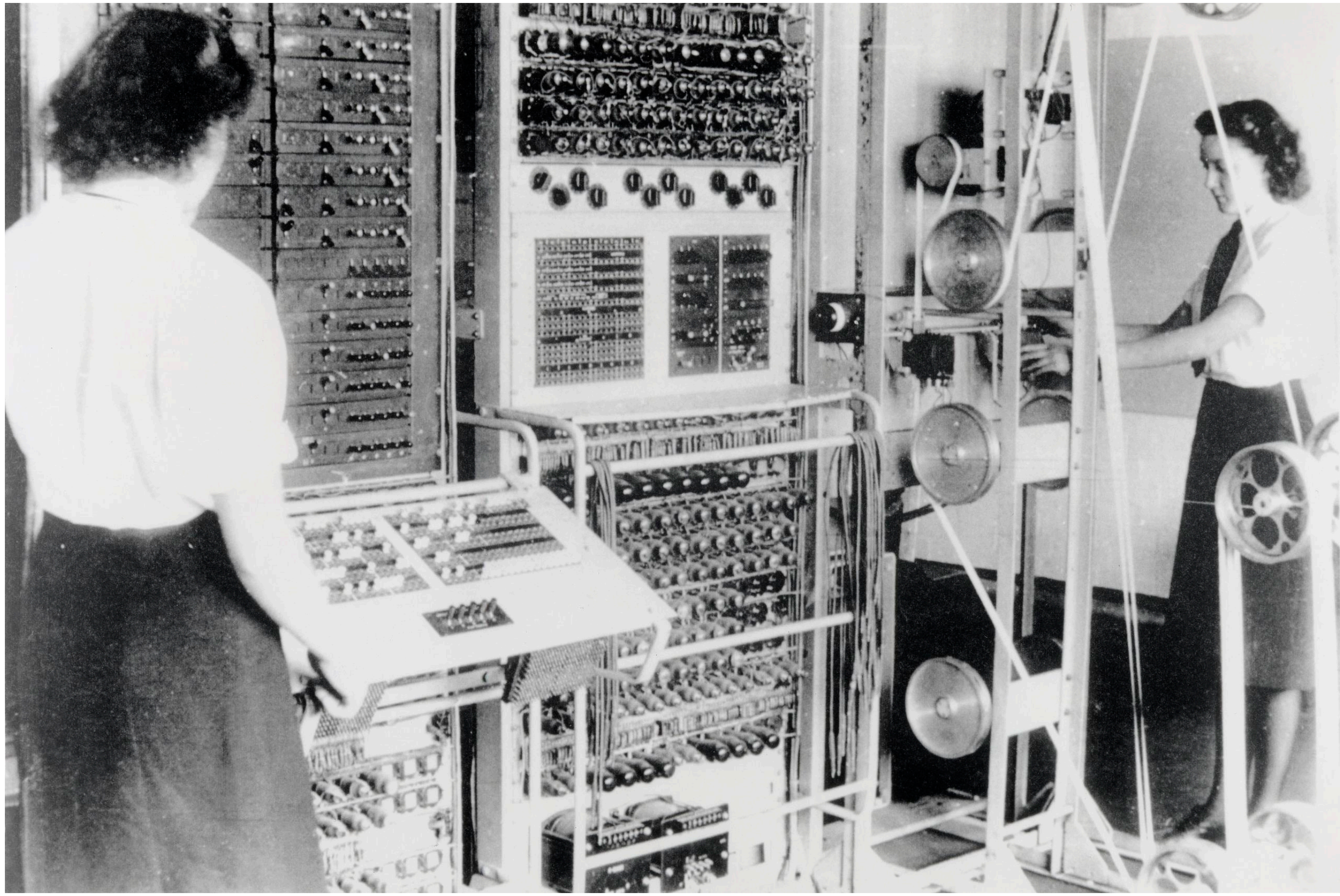
- **Enigma** machine: a German electro-mechanical *rotor cipher*
  - Invented in ~1920
  - Heavily used in WWII.
- Three **Rotors** scrambled each character, and changed configuration after each character.
- Configuration of wires connected to the **plugboard** changed the output. This configuration was a **key**.



Right rotor rotated, so second "A" leads to a different output.

British **Ultra** project decoded many messages in WWII

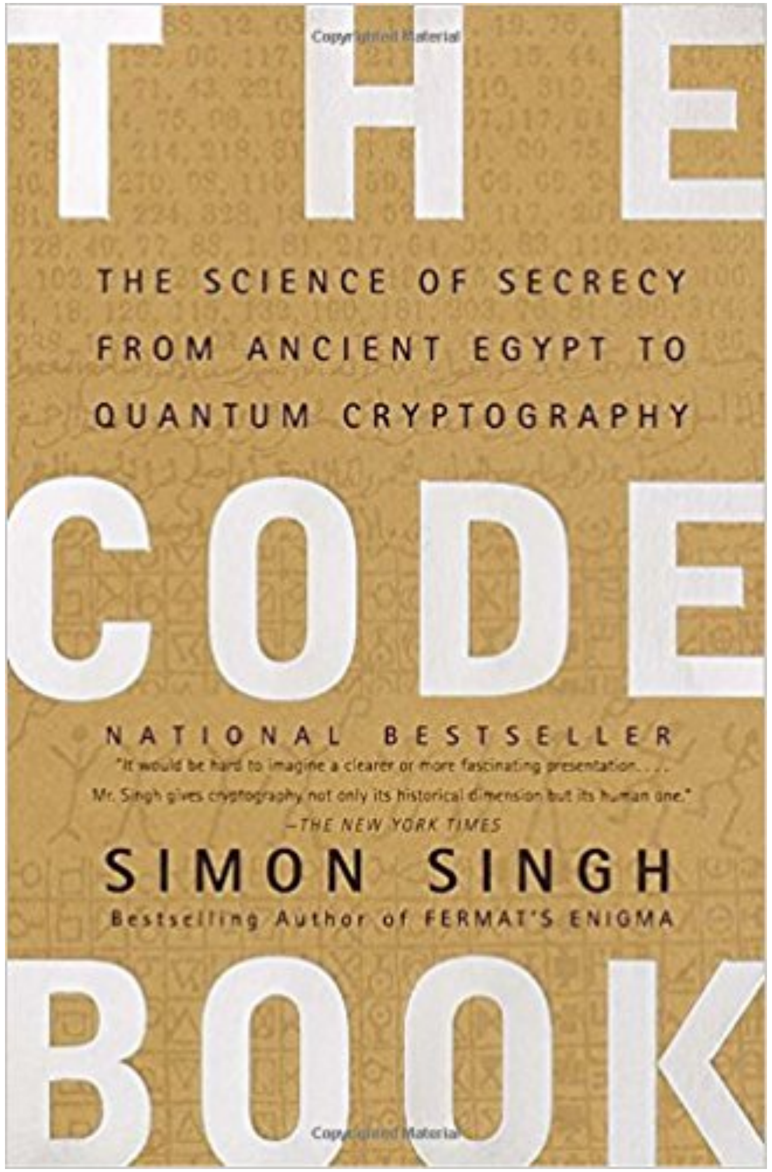
- Alan Turing and the team at Bletchley Park used statistics and cryptanalysis to decode German messages.
- German messages often repeated certain phrases at the start, making decoding easier.
- Built two computers for decoding:
  - Bombe (electromechanical), and Collosus (vacuum tubes)
- Somewhat innaccurately portrayed in “The Imitation Game” 2014 film.



“Collosus” computer

Further reading on the history of ciphers

- The Code Book, by Simon Singh

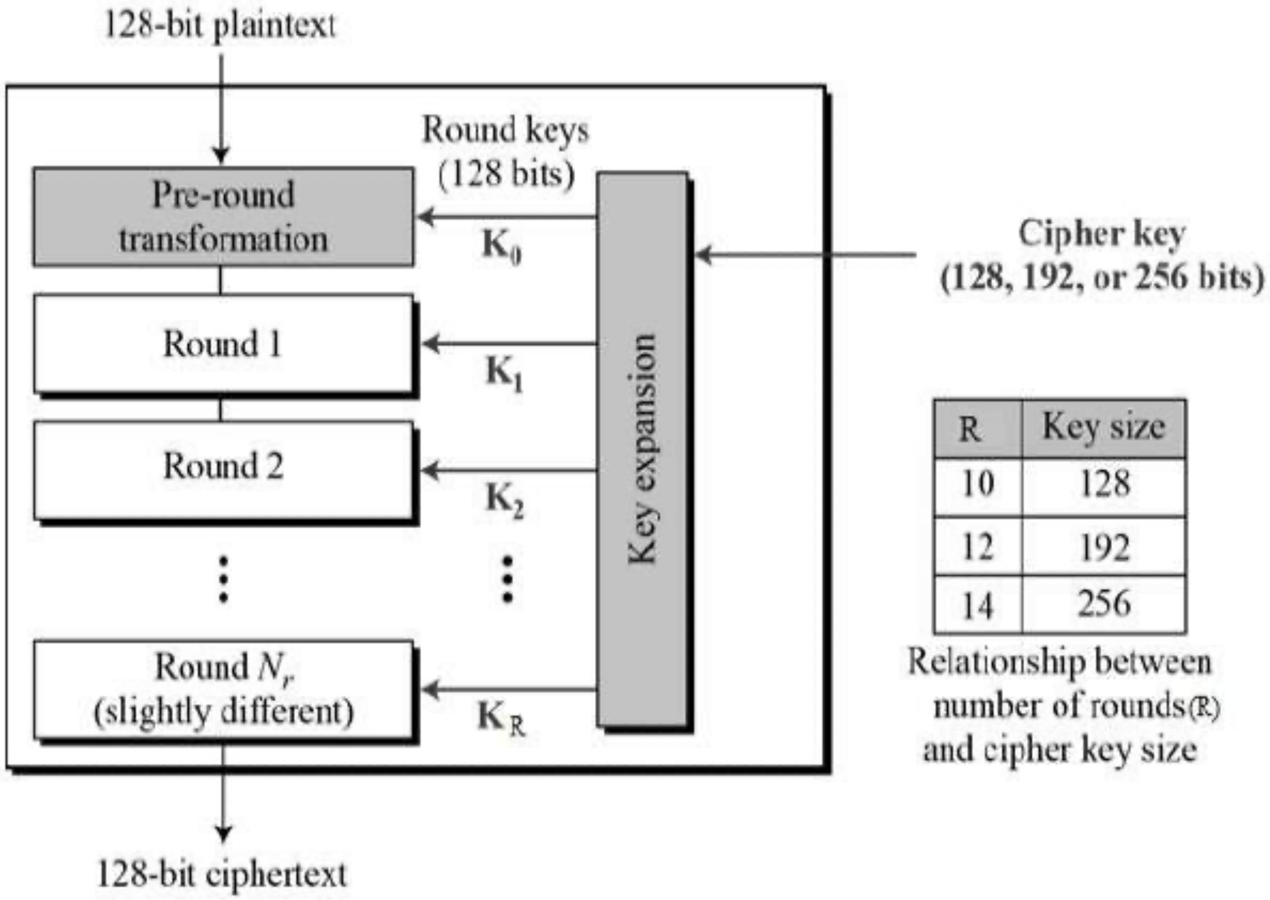


Strong encryption with AES

- Advanced Encryption Standard (AES) uses the Rijndael cipher.
- AES is used to encrypt most HTTPS encrypted web traffic.
- Key length can be 128, 192, or 256 bits
- Using a larger key is more secure, but slower.
- Brute force attack of AES-128 would likely require decrypting the message $2^{127} = 2\times10^{38}$ times. This would take ~[one billion *billion* years]
- In reality, AES has some flaws that allow an attacker to narrow down the search space, but it's still impractical to break (as far as we know).
- AES is a **symmetric-key** encryption algorithm:
  - The same key is used to encrypt and decrypt.

AES characteristics

- Operates on a fixed-size **block** of plaintext.
- Organizes data into a matrix.
- Performs many rounds of these transformations:
  - Replace bytes using a lookup table (a non-linear operation)
  - Last 3 rows are shifted *k* steps.
  - Combine 4 bytes in each column
  - XOR the "round key" to each byte



R	Key size
10	128
12	192
14	256

Relationship between number of rounds (R) and cipher key size

Key distribution

- **Symmetric-key encryption** (including AES) requires that the two participants have a **shared secret key**.
- Sender can generate the key with a good random number generator.
- But how can we send the key to the receiver?

  - Simple solution is to use a **pre-shared key**:
    - Must somehow *meet* ahead of time to agree on a key.
    - Both participants type-in the same passphrase/key before communicating.
  - But, most secure communication over the Internet is between "strangers."
  - Cannot send the key as plaintext over the network – attacker would see it!

- In practice, **public key cryptography** is used to share keys.
  - Does not require pre-sharing the key. Can interact entirely over the network.

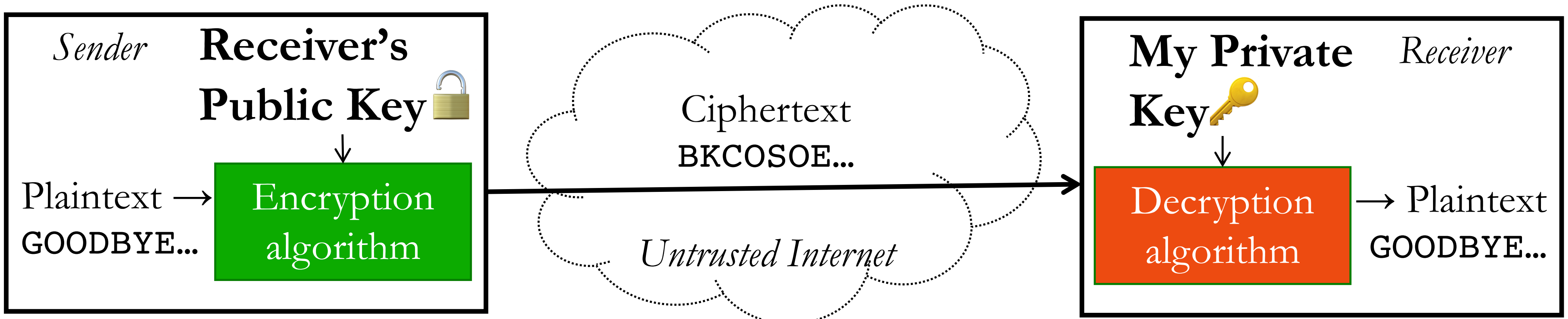
Public Key Cryptography

- Public Key cipher is **asymmetric** because it uses **two** related keys:
  - **Public key** is used for encryption.
  - **Private key** is used for decryption.

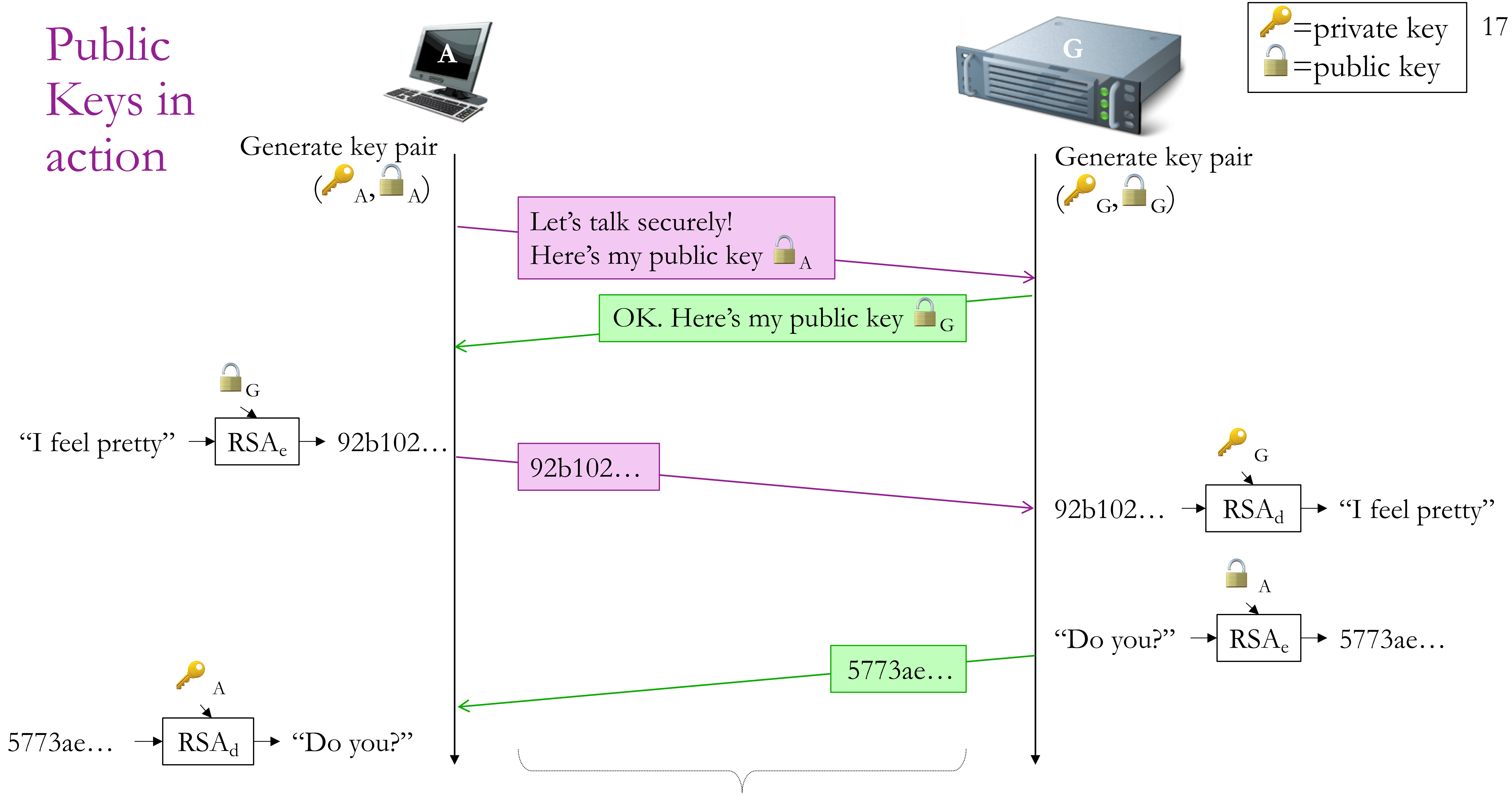
Mathematically related. Must be generated simultaneously.

- Each party generates a (public, private) **key pair**
  - Keep your private key hidden, and share the public key openly.
- To send you an encrypted message, I just need your public key.
- **RSA** (invented in the 1970s) uses factoring of large prime integers.
- **Elliptic Curve Cryptography** (ECC) came into use around 2005.

Both these are used on today's Internet.



Public Keys in action

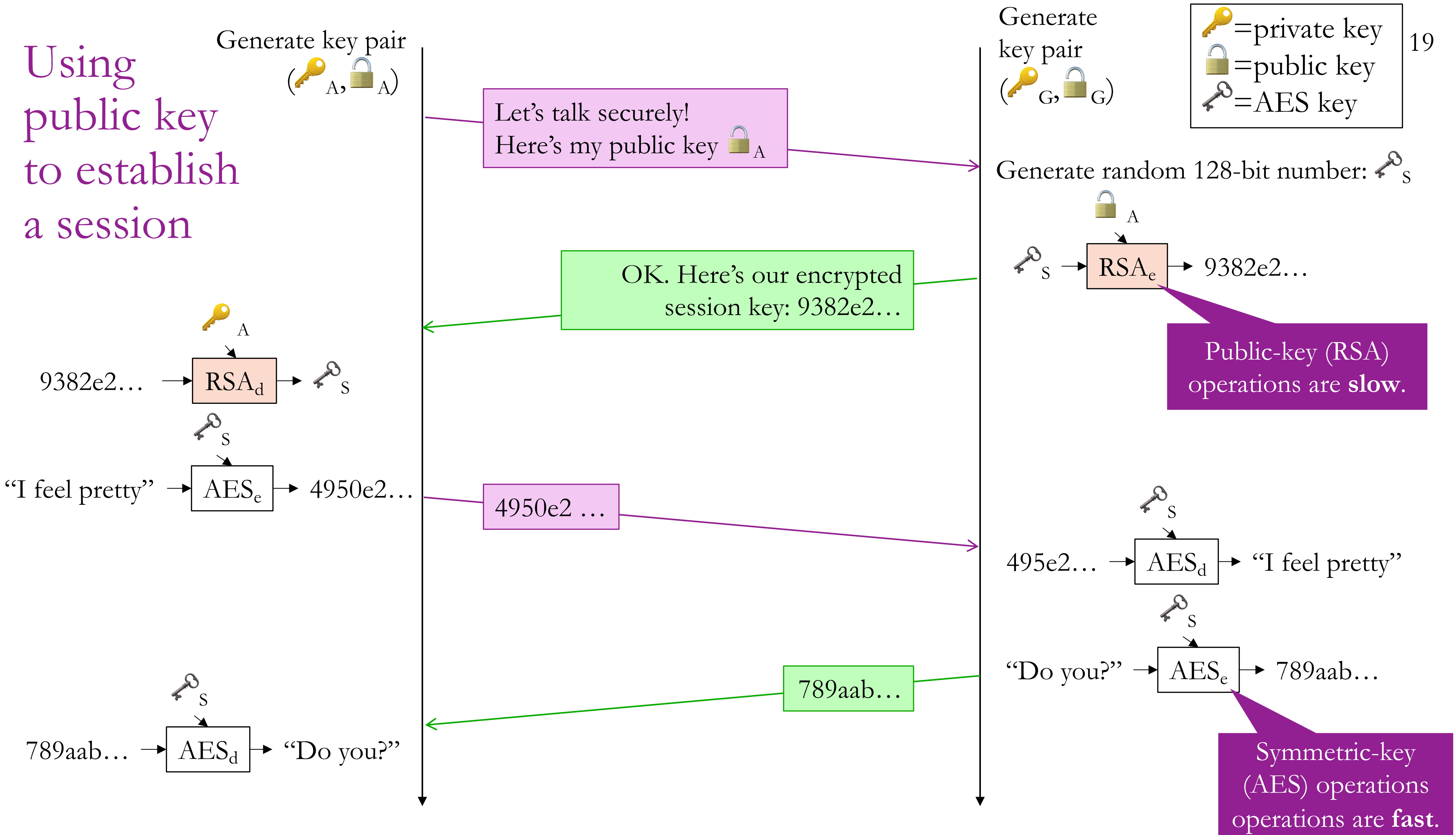


Network observer only sees public keys and encrypted messages. A & G are never seen.

Combining asymmetric and symmetric ciphers

- Public key algorithms like RSA are much more computationally expensive than symmetric ciphers like AES.
- In practice, we use public key cryptography just to exchange a shared key (a session key), then use AES for the remaining messages.
- This allows us to use an efficient symmetric cipher (AES) **without** having *pre-shared* a key.

Using public key to establish a session



RSA Public Key Cryptography: basic principles

- RSA uses **modular exponentiation** for encryption and decryption.
- Number theory tells us that we can find three large integers: *n, e, d* such that for any integer $m < n$:

$$(m^e)^d \bmod n = m$$

- ***m*** is the **message** to encrypt, viewed as an integer.
- ***e*** is the **encryption exponent**. The pair (e, n) is the public key.
    - To encrypt a message to ciphertext ***c*****,** raise the plaintext to the e^{th} power (mod n):

$$m^e \bmod n = c$$

- ***d*** is the **decryption exponent**. The pair (d, n) is the private key.
    - To decrypt a message, raise the ciphertext to the d^{th} power (mod n):

$$c^d \bmod n = (m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = m$$

More about RSA

- It turns out that there are many sets of (e, d, n) that satisfy

$$(m^e)^d \bmod n = m, \quad \textit{for all} \quad m > n$$

- These form different public-private key pairs for RSA encryption
- We can't go into the details here (generating (e, d, n), proofs…).
    - Each key pair can be derived from a pair of large large prime numbers.
- Key idea is that modular exponentiation of huge numbers is efficient, but factoring large prime numbers (to reverse it) takes exponential time.
    - Even knowing **e**, **n**, and **m**, it's extremely difficult to find **d**.

- Notice that RSA public and private keys can be ***reversed***:

$$(m^e)^d \bmod n = m^{ed} \bmod n = (m^{\mathbf{d}})^{\mathbf{e}} \bmod n = m$$

    - If private key is used for encryption, then the public key will decrypt it!
    - Later, we'll use RSA keys in this reverse way for digital signatures.

- Good RSA demo: https://www.cryptool.org/en/cto-highlights/rsa-step-by-step

Intermission

Anonymity on the Internet

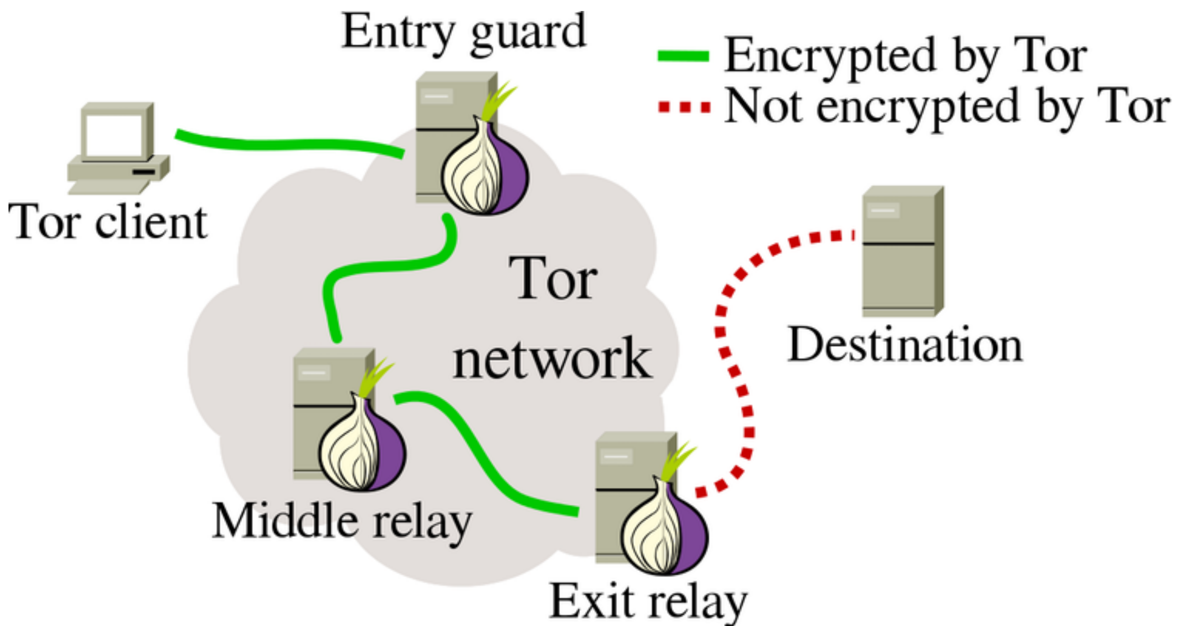
- Even with encryption, routers know **who** you are communicating with.
- Every IP packet reveals its source and destination IP address.
- This allows governments and corporations to monitor and block access to certain websites or services.
- **Anonymity** can be a goal of network security.

Onion routing obscures the communication path

- **Onion Routing** creates an **overlay network** at the application layer.
  - **Mix routing** idea described in 1981 by [David Chaum]
  - Further developed in the 1990s at the US Naval Research Lab.
  - Implemented in the [Tor Browser] in ~2002--2006.
  - Used by the [Dark Web], starting in ~2011.
- Sending a message from A to B involves a chain of many TCP connections to hide the original source and final destination of the traffic.
- Public-key encryption lets a relay forward traffic one hop without knowing the final destination.

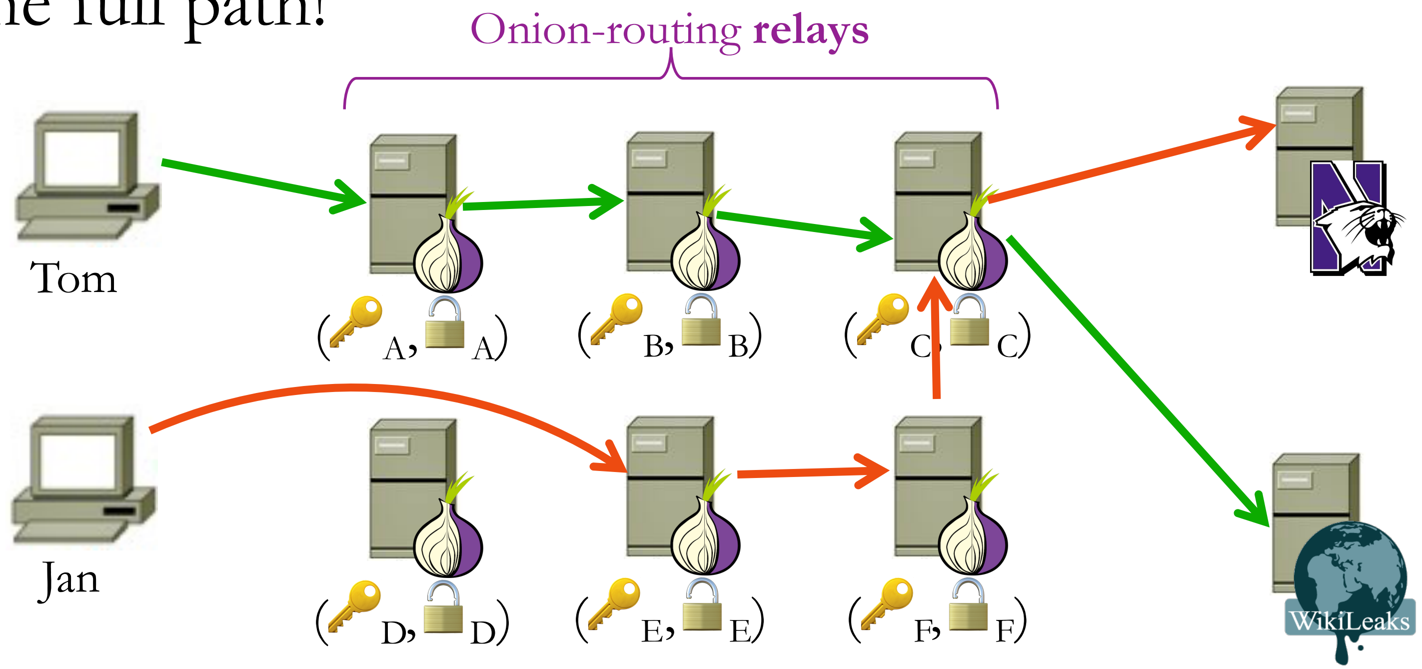
Assumption:

- Attacker may control one or a few Internet hosts but not all.



Onion routing involves a random choice of several **relays**

- Relays are listed in a public directory for anyone to see.
  - Each relay publishes its IP address and public key.
- Listens for encrypted messages sent over TCP connections.
  - Will accept traffic from anyone and will deliver it wherever instructed.
- Client chooses the full path to be taken by the message.
  - No other node knows the full path!



Building an anonymized, onion-routed message

- Message is encapsulated in several **layers of encryption**, one for each hop in the onion network.
- Sender 1.2.3.4 does the following:
  - Constructs a message to be delivered to 5.6.7.8:80. Message includes some encrypted return address info.
  - Randomly choose several relays: A,B,C.
  - Get their public keys: $_A$, $_B$, $_C$
  - Add onion layers to the message, by:
    - Adding a header with the relay X address
    - Encrypting the message with $_X$

To: Relay A

Content encrypted by $_A$:

To: Relay B

Content encrypted by $_B$:

To: Relay C

Content encrypted by $_C$:

To: 5.6.7.8:80

Return-address: [???]

```
GET / HTTP/1.1
Host: wikileaks.org
```

Encrypted layers of the onion

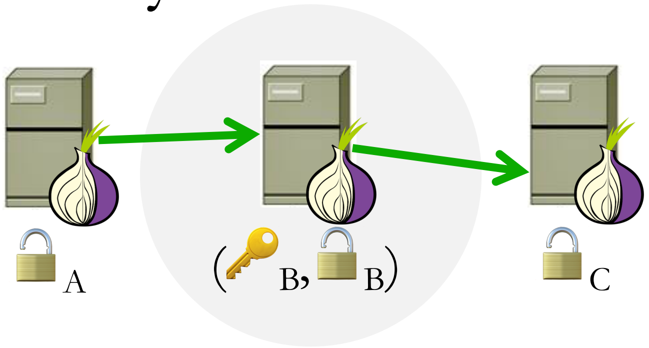
To: Relay A

Content encrypted by 🔓$_A$:

(need private key 🔑$_A$ to decrypt)

Each relay has a limited view of the transaction

Consider Relay B:



B knows

- Who is contacting it (other relays).
- Who it is forwarding to (other relays).
- Nothing else!

B gets this from Relay A:

To: Relay B

Content encrypted by B:

(need private key B to decrypt)

B unwraps one layer of the onion.
Decrypts using B to this:

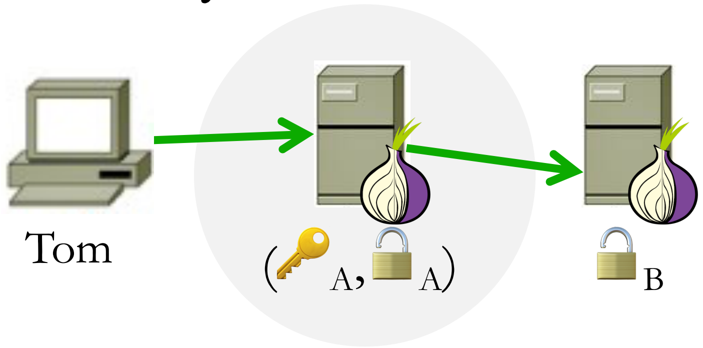
To: Relay C

Content encrypted by C:

(need private key C to decrypt)

Entry node

Consider Relay A:



Acting as the **entry node** in this transaction, Relay A knows:

- The IP address of a Tor participant (Tom). This is *slightly* sensitive.
- Who it is forwarding to (another relay).
- Nothing about the final destination.

A gets this from Tom:

To: Relay A

Content encrypted by $_A$:

(need private key $_A$ to decrypt)

A unwraps one layer of the onion.
Decrypts using $_A$ to this:

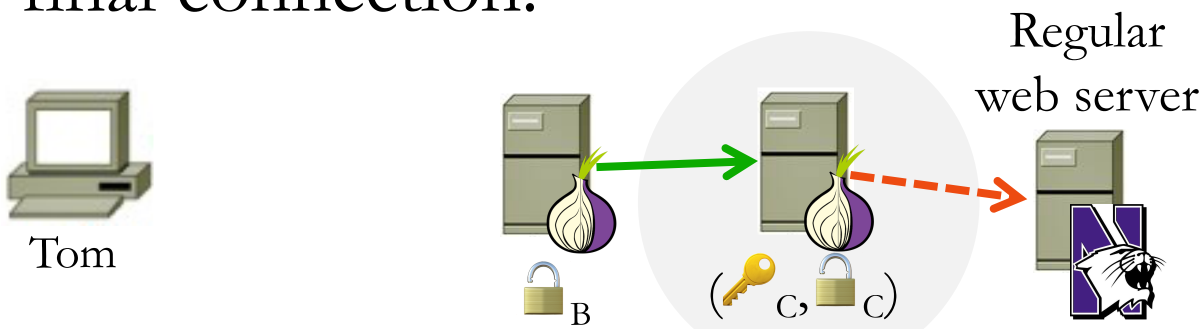
To: Relay B

Content encrypted by $_B$:

(need private key $_B$ to decrypt)

Exit node

Tor is designed to be compatible with any web server, and exit nodes make the final connection:



Exit node C knows:

- The destination address. Only *slightly* sensitive, because don't know sender.
- The request contents.
- Something about the return address…

What must the exit node know to return a response to the client?

C gets this from Relay B:

To: Relay C

Content encrypted by C:

(need private key C to decrypt)

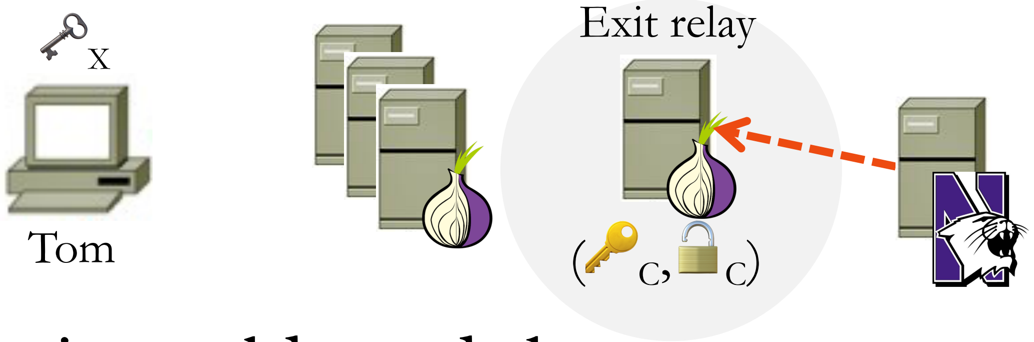
C unwraps one layer of the onion.
Decrypts using C to this:

To: 8.9.9.2:80

Return-address: [???]

```
GET / HTTP/1.1
Host: northwestern.edu
```

Hiding the return address



- If the exit relay knew the client address, it could send the response back in the normal onion-routing way, but that would reveal too much.
  - The relationship between source and destination must be kept secret all!
- **Solution** (described by Chaum): client request includes:
  - an **onion-encrypted return path**, chosen and encrypted by the *client*, and
  - a **one-time-use encryption key** for the response message.
- Request seen by exit node looks like this:
- Why is a new key generated?

To: 8.9.9.2:80
Return-key: X
Return-path: (D, *remainder of path encrypted by* D**)**

```
GET / HTTP/1.1
Host: northwestern.edu
```

Onion-routed request vs. response

To: Relay A

Content encrypted by A:

To: Relay B

Content encrypted by B:

To: Relay C

Content encrypted by C:

To: 8.9.9.2:80

Return-key: X

Return-path: (D, *remainder of path encrypted by* D **)**

```
GET / HTTP/1.1
Host: northwestern.edu
```

Encrypted return path is copied from request

Response constructed by Exit Relay C:

To: Relay D

Content encrypted by D:

To: Relay E

Content encrypted by E:

To: Relay F

Content encrypted by F:

Deliver-To: 1.2.3.4:10001

Final relay will know destination, but not source

Content encrypted by X:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2093

<html> …
```

Anonymous services

- Tor focuses on making web browsing (clients) anonymous.
- I2P takes the idea further by allowing anonymous web hosting (and other services).
  - Hostnames map to onion-encrypted paths (instead of to IP addresses).
  - Doesn't use DNS, but another distributed database on the P2P relay network.
- Cryptocurrencies (eg., Bitcoin) enable anonymous e-commerce.
- This is the Dark Web.

Web-based onion services in February 2016

Category	% of total	% of active
Violence	0.3	0.6
Arms	0.8	1.5
Illicit Social	1.2	2.4
Hacking	1.8	3.5
Illicit links	2.3	4.3
Illicit pornography	2.3	4.5
Extremism	2.7	5.1
Illicit Other	3.8	7.3
Illicit Finance	6.3	12
Illicit Drugs	8.1	15.5
Non-illicit+Unknown	22.6	43.2
Illicit total	29.7	56.8
Inactive	47.7	
Active	52.3	

Tradeoffs in a free society

- Censorship and surveillance are the favorite tools of oppressors.
  - 2% of the East German population worked full-time in domestic surveillance.
  - 15% of their population acted as informants to the secret police (Stasi).
  - This was done even with old technologies – a person had to listen in real time.
- Democratic societies are not immune to these threats.
  - U.S. National Security Agency has an estimated $10B annual budget and ~35k employees. Roughly the size of Facebook's staff. What are they all doing?
    - Snowden's 2013 whistleblowing revealed that NSA has access to (eg.,) Google cloud data, phone records (including GPS), and was listening in on U.S. phone conversations.
- On the other hand, anonymity removes consequences.

Anonymity online + the ability to do **harm** online = **chaos**.

  - Is it a good thing to enable child pornography, hitmen-for hire, etc?

Recap

- Network security goals are:
  - **Confidentiality, Reliability, Integrity, Authentication & Anonymity**
- Routers and other participants on the network cannot be trusted.
- **AES** is a the standard **symmetric-key** encryption algorithm. Must somehow establish a shared session key, used by both parties.
- Public Key cryptography (**RSA**, ECC) uses a pair of related keys.
  - **Public key** is openly advertised and is used for encryption
  - **Private key** is secret and is used for decryption.
- Onion-routing/mix networks create routing **overlays** on the Internet.
  - Sender encrypts data many times. Relays decrypt one layer each.
  - This enables **anonymous** web browsing and even anonymous services.