CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 1 - Basics of cryptography

Spring 2022

Outline

1. Basics of cryptography
2. Secret-key encryption
3. Public-key encryption
4. Integrity
5. Authentication

Cryptology

- Cryptology is a science that studies:
  - Cryptography ("secret writing"): *Making secret messages*
    - Turning plaintext (an ordinary readable message) into ciphertext (secret messages that are "hard" to read)
  - Cryptanalysis: *Breaking secret messages*
    - Recovering the plaintext from the ciphertext
- The point of cryptography is to send secure messages over an insecure medium (like the Internet)
- Cryptanalysis studies cryptographic systems to look for weaknesses or leaks of information

The scope of these lectures

- The goal of the cryptography unit in this course is to show you what cryptographic tools exist, and information about using these tools in a secure manner
- We won't be showing you details of how the tools work
  - For that, see CO 487, chapter 2 of van Oorschot's text book, or chapter 2.3 of Pfleeger's textbook

Dramatis personae

When talking about cryptographic schemes, we often use a standard cast of characters:

- Alice, Bob, Carol, Dave
  - People (usually honest) who wish to communicate
- Eve
  - A passive eavesdropper, who can listen to any transmitted messages
- Mallory
  - An active Man-In-The-Middle, who can listen to, and modify, insert, or delete, transmitted messages
- Trent
  - A Trusted Third Party
- ... many more ...
  - Peggy (prover), Victor (verifier), etc.

Building blocks

Cryptography contains three major types of components

- Confidentiality components
  - Preventing Eve from reading Alice's messages
- Integrity components
  - Preventing Mallory from modifying Alice's messages without being detected
- Authenticity components
  - Preventing Mallory from impersonating Alice

Often remembered as CIA.

Kerckhoffs' principle

Shannon's maxim: one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them.

- So don't use secret encryption methods
  - Then what do we do?
- Have a large class of encryption methods, instead
  - Hopefully, they are all equally strong
- Make the class public information
- Use a secret key to specify which one you're using
- It's easy to change the key; it's usually just a smallish number

Kerckhoffs's principle: a cryptosystem should be secure, even if everything about the system, except the key, is public knowledge

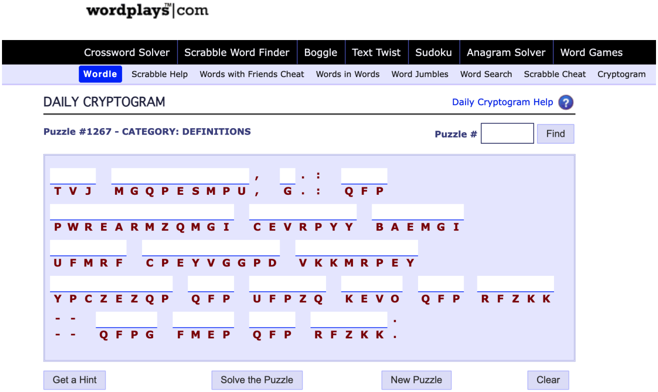
Kerckhoffs' principle

Kerckhoffs' principle has a number of implications:

- The system is *at most* as secure as the number of keys
- Eve can just try them all, until she finds the right one
- A strong cryptosystem is one where that's the best Eve can do
  - With weaker systems, there are shortcuts to finding the key

- Example: newspaper cryptogram has 403,291,461,126,605,635,584,000,000 possible keys
- But you don't try them all; it's way easier than that!

Daily cryptogram



DAILY CRYPTOGRAM

Puzzle #1267 - CATEGORY: DEFINITIONS

TVJ MGQPESMPU, G.: QFP

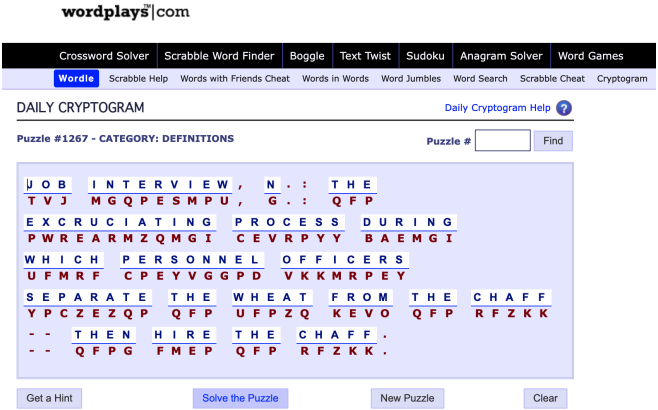
PWREARMZQMGI CEVRPYY BAEMGI

UFMRF CPEYVGGPD VKKMRPEY

YPCZEZQP QFP UFPZQ KEVO QFP RFZKK

-- QFPG FMEP QFP RFZKK.

Daily cryptogram



DAILY CRYPTOGRAM

Daily Cryptogram Help

Puzzle #1267 - CATEGORY: DEFINITIONS

Puzzle # Find

JOB INTERVIEW, N.: THE
TVJ MGQPESMPU, G.: QFP

EXCRUCIATING PROCESS DURING
PWREARMZQMGI CEVRPYY BAEMGI

WHICH PERSONNEL OFFICERS
UFMRF CPEYVGGPD VKKMRPEY

SEPARATE THE WHEAT FROM THE CHAFF
YPCZEZQP QFP UFPZQ KEVO QFP RFZKK

-- THEN HIRE THE CHAFF.
-- QFPG FMEP QFP RFZKK.

Strong cryptosystems

What information do we assume the attacker (Eve) has when she's trying to break our system?

- She may:
  - Know the algorithm (the public class of encryption methods)
  - Know a number (maybe a large number) of corresponding plaintext/ciphertext pairs
  - Have access to an encryption and/or decryption oracle

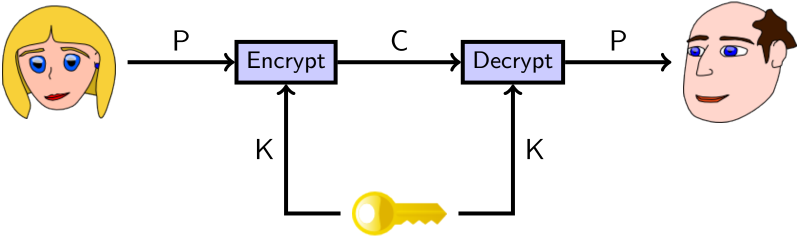
And we still want to prevent Eve from learning the key!

Outline

1. Basics of cryptography
2. Secret-key encryption
3. Public-key encryption
4. Integrity
5. Authentication

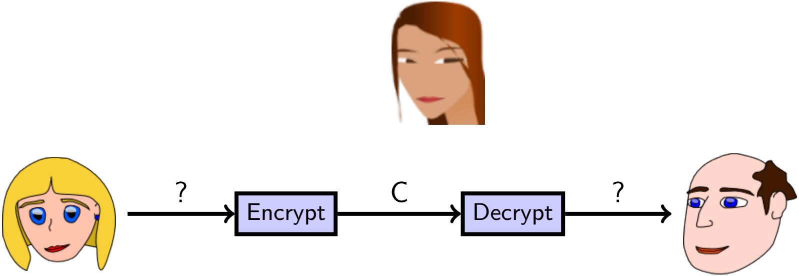
Secret-key encryption

- Secret-key encryption is the simplest form of cryptography
- Used for thousands of years
- Also called symmetric encryption
- The key Alice uses to encrypt the message is the same as the key Bob uses to decrypt it
- $D_k(E_k(m)) = m$



Secret-key encryption

- Eve, not knowing the key, should not be able to recover the plaintext



Perfect secret-key encryption

Is it possible to make a completely unbreakable cryptosystem?

- Yes: the One-Time Pad

- It's also very simple:
  - The key is a truly random bitstring of the same length as the message
  - The "Encrypt" and "Decrypt" functions are both XOR

One-time pad

- It's very hard to use one-time pad correctly
  - The key must be truly random, not pseudorandom
  - The key must never be used more than once!
    - A "two-time pad" is insecure!
- Q: Why does "try every key" not work here?
- Q: How do you share the secret keys?
- Used in the Washington / Moscow hotline for many years

Computational security

In contrast to the "perfect" security property of one-time pad, most cryptosystems have "computational" security

- This means that it's certain they can be broken, given *enough* work by Eve
- How much is "enough"?

- At worst, Eve tries every key
  - How long that takes depends on how long the keys are
  - But it only takes this long if there are no "shortcuts"!

Some data points

- One modern computer can try about 17 million keys per second
- A medium-sized company or research lab may have 100 computers
- The BOINC project (the largest computing grid in the world) has the computation power of about 300,000 computers

Berkeley Open Infrastructure
for Network Computing

40-bit crypto

This was the US legal export limit for a long time

$2^{40} = 1{,}099{,}511{,}627{,}776$ possible keys

- One computer: 18 hours
- One lab: 11 minutes
- BOINC: 200 ms

56-bit crypto

This was the US government standard (DES) for a long time

$2^{56} = $ 72,057,594,037,927,936 possible keys

- One computer: 134 years
- One lab: 16 months
- BOINC: 4 hours

128-bit crypto

This is the modern standard

$2^{128} =$ 340,282,366,920,938,463,463,374,607,431,768,211,456

- One computer: 635 thousand million million million years
- One lab: 6 thousand million million million years
- BOINC: 2 million million million years

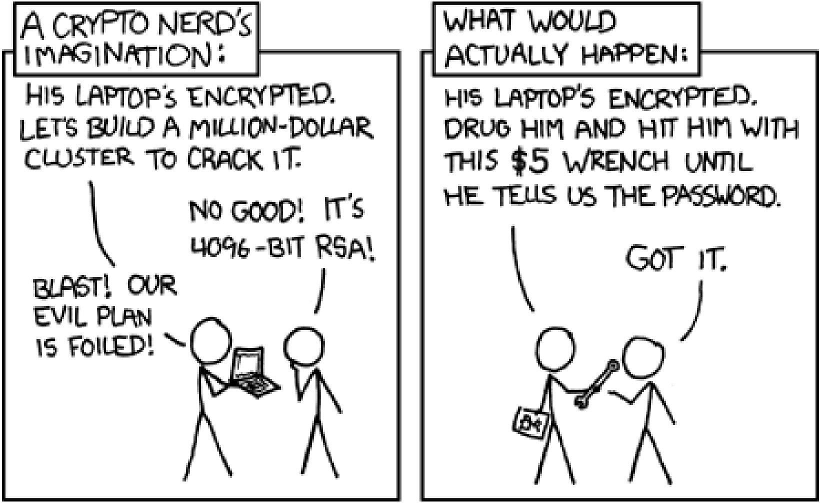
The Universe is $\approx$13 thousand million years old

Well, we cheated a bit

This isn't really true, since computers get faster over time
Moore's law: computing speed doubles every 18 months

- A better strategy for breaking 128-bit crypto is just to wait until computers get 2^{88} times faster, then break it on one computer in just 18 hours.
- How long do we need to wait? 132 years.
- If we believe Moore's law will keep on working, we'll be able to break 128-bit crypto in 132 years (and 18 hours) :-)
  - Q: Do we believe this?
- How about quantum computers? e.g., Grover's algorithm
  - reduces the search space from 2^{128} to 2^{64}
  - requires around 3,000 logical qubits (we have 127 qubits now)

An even better strategy


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

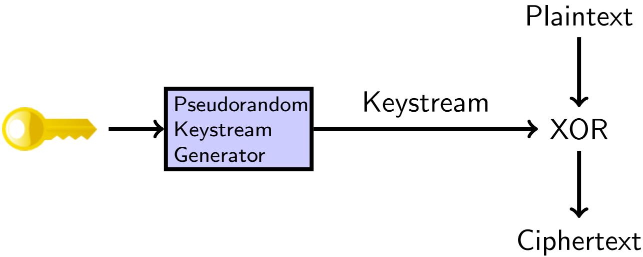
Types of secret-key cryptosystems

Secret-key cryptosystems come in two major classes

- Stream ciphers
- Block ciphers

Stream ciphers

- A stream cipher is what you get if you take the One-Time Pad, but use a pseudorandom keystream instead of a truly random one



- RC4 was the most common stream cipher on the Internet but deprecated. ChaCha is increasingly popular (Chrome and Android), and SNOW3G is mostly used in mobile phone networks.

Stream ciphers

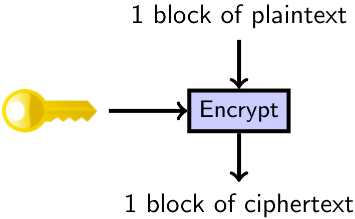
- Stream ciphers can be very fast
  - This is useful if you need to send a lot of data securely
- But they can be tricky to use correctly!
  - What happens if you use the same key to encrypt two messages?
    - $C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2$
  - How would you solve this problem without requiring a new shared secret key for each message?
    - $K' = K \parallel nonce$
    - Where have we seen this technique before?
- WEP, PPTP are great examples of how not to use stream ciphers. (The insecurity of the WEP algorithm is first identified by Prof. Goldberg in our school, and we'll cover it in the next lecture.)

Block ciphers

- Stream ciphers operate on the message one bit at a time
  - What happens in a stream cipher if you change just one bit of the plaintext?
- An alternative design is block ciphers
  - Block ciphers operate on the message one block at a time
  - Blocks are usually 64 or 128 bits long
- AES is the block cipher everyone should use today
  - Unless you have a really, really good reason
  - Native AES support on Intel chips since Westmere (2010)

Modes of operation

- Block ciphers work like this:

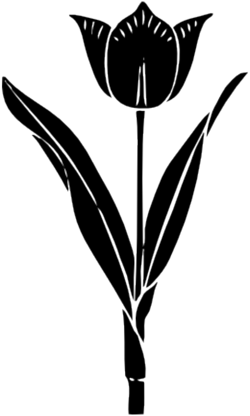


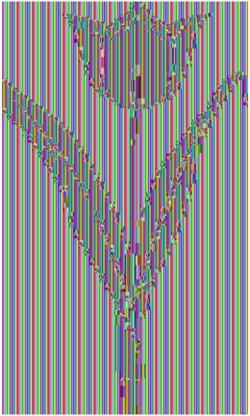
- But what happens when the plaintext is larger than one block?
  - The choice of what to do with multiple blocks is called the mode of operation of the block cipher

Modes of operation

The simplest thing to do is just to encrypt each successive block separately.

- This is called Electronic Code Book (ECB) mode
- But if there are repeated blocks in the plaintext, you'll see the same repeating patterns in the ciphertext:

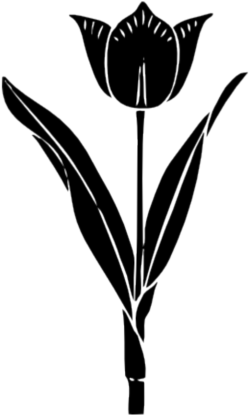


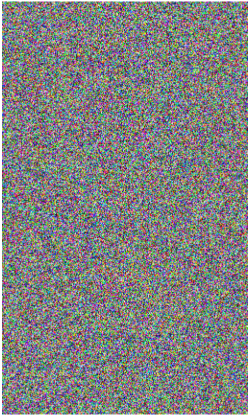


Modes of operation

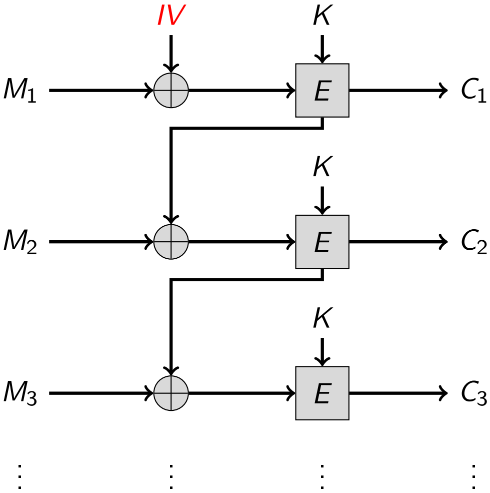
There are much better modes of operation to choose from, Common ones include Cipher Block Chaining (CBC), Counter (CTR), and Galois Counter (GCM) modes

- Patterns in the plaintext are no longer exposed because these modes involves some kind of "feedback" among different blocks





Cipher Block Chaining (CBC) encryption process



Initialization vector (IV)

Without the IV, what will happen if we encrypt the same message twice with the same key?

- $C_1 = E_K(P),\ C_2 = E_K(P) \implies C_1 = C_2$

Solutions?

- Option 1: change the K
  - $C_1 = E_{K_1}(P),\ C_2 = E_{K_2}(P)\,,\ K_1 \neq K_2 \implies C_1 \neq C_2$
- Option 2: "change" the P
  - $C_1 = E_K(P \parallel IV_1),\ C_2 = E_K(P \parallel IV_2)\,,\ IV_1 \neq IV_2 \implies C_1 \neq C_2$
- Then why Option 2 is preferred?
  - Because we can send IV in the clear!

An initialization vector might also be called as a nonce (number used once) or a salt.

Key exchange

How do Alice and Bob share the secret key?

- Meet in person
- Diplomatic courier
- ...
- In general this is very hard

Or, we invent new technology...

Outline

1. Basics of cryptography
2. Secret-key encryption
3. Public-key encryption
4. Integrity
5. Authentication

Public-key cryptography

- Invented (in public) in the 1970's
- Also called asymmetric cryptography
  - Allows Alice to send a secret message to Bob without any prearranged shared secret!
  - In secret-key cryptography, the same key encrypts the message and also decrypts it
  - In public-key cryptography, there's one key for encryption, and a different key for decryption!
- Some common examples:
  - RSA, ElGamal, ECC, NTRU, McEliece

Public-key cryptography

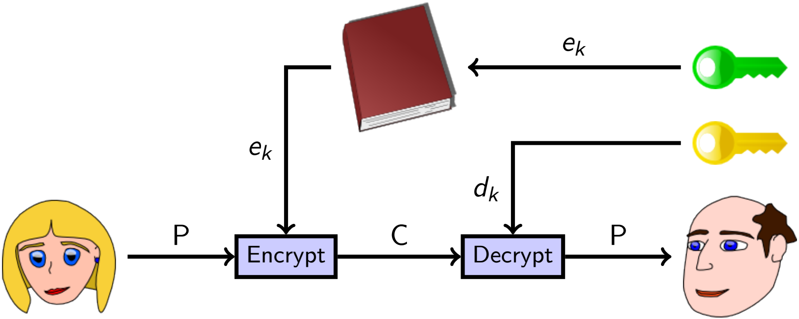
How does it work?

1. Bob creates a key pair (e_k, d_k)
2. Bob gives everyone a copy of his public encryption key e_k
3. Alice uses it to encrypt a message, and sends the encrypted message to Bob
4. Bob uses his private decryption key d_k to decrypt the message
    - Eve can't decrypt it; she only has the encryption key e_k
    - Neither can Alice!
    - It must be hard to derive d_k from e_k

So with this, Alice just needs to know Bob's public key in order to send him secret messages

- These public keys can be published in a directory somewhere

Public-key cryptography



Public key sizes

- Recall that if there are no shortcuts, Eve would have to try 2^{128} things in order to read a message encrypted with a 128-bit symmetric key.
- Unfortunately, all of the public-key methods we know do have shortcuts. For example:
  - Eve could read a message encrypted with a 128-bit RSA key with just 2^{33} work, which is easy!
    - In RSA, $n = pq$; n is public; factoring n reveals the key
    - 2^{33} is the “work factor” to factor a 128-bit integer n
    - Quantum computers can factor even faster, see Shor’s algorithm
  - If we want Eve to have to do 2^{128} work, we need to use a much longer public key

Hybrid cryptography

In addition to having longer keys, public-key cryptography takes a long time to calculate (as compared to secret-key cryptography)

- Using public-key to encrypt large messages would be too slow, so we take a hybrid approach:
  - Pick a random 128-bit key K for a secret-key cryptosystem
  - Encrypt the large message with the key K (e.g., using AES)
  - Encrypt the key K using a public-key cryptosystem
  - Send both the encrypted message and the encrypted key to Bob
- This hybrid approach is used for almost every cryptography application on the Internet today

Is that all there is?

It seems we've got this "sending secret messages" thing down pat. What else is there to do?

- Even if we're safe from Eve reading our messages, there's still the matter of Mallory
- It turns out that even if our messages are encrypted, Mallory can sometimes modify them in transit!
- Mallory won't necessarily know what the message says, but can still change it in an undetectable way
  - e.g. bit-flipping attack on stream ciphers
- This is counterintuitive, and often forgotten

How do we make sure that Bob gets the same message Alice sent?

Outline

1. Basics of cryptography
2. Secret-key encryption
3. Public-key encryption
4. Integrity
5. Authentication

Integrity components

How do we tell if a message has changed in transit?

- Simplest answer: use a checksum
- For example, add up all the bytes of a message
- The last digits of serial numbers (credit card, ISBN, etc.) are usually checksums
- Alice computes the checksum of the message, and sticks it at the end before encrypting it to Bob.
- When Bob receives the message and checksum, he verifies that the checksum is correct

This doesn't work!

- With most checksum methods, Mallory can easily change the message in such a way that the checksum stays the same
- We need a "cryptographic" checksum
- It should be hard for Mallory to find a second message with the same checksum as any given one

Cryptographic hash functions

- A hash function h takes an arbitrary length string x and computes a fixed length string $y = h(x)$ called a message digest
  - Common examples: MD5, SHA-1, SHA-2, SHA-3 (AKA Keccak, from 2012 on)
- Hash functions should have three properties:
  - Preimage-resistance:
    - Given y, it's hard to find x such that $h(x) = y$ i.e., a "preimage" of x
  - Second preimage-resistance:
    - Given x, it's hard to find $x' \neq x$ such that $h(x) = h(x')$ i.e., a "second preimage" of $h(x)$
  - Collision-resistance:
    - It's hard to find any two distinct values x, x' such that $h(x) = h(x')$ i.e., a "collision"

What is “hard”?

- Collisions are always easier to find than preimages or second preimages due to the well-known birthday paradox
  - If there are 2^n digests, we need to try an average $2^{n/2}$ messages to find 2 with the same digest
- For SHA-1, for example, it takes 2^{160} work to find a preimage or second preimage, and 2^{80} work to find a collision using a brute-force search
  - However, there are faster ways than brute force to find collisions in SHA-1 or MD5

The birthday paradox

- If there are n people in a room, what is the probability that at least two people have the same birthday?
  - For n = 2: P(2) = 1 - $\frac{364}{365}$
  - For n = 3: P(3) = 1 - $\frac{364}{365} \times \frac{363}{365}$
  - For n people: P(n) = 1 - $\frac{364}{365} \times \frac{363}{365} \times \ldots \times \frac{365-n-1}{365}$
- With 22 people in the room, there is better than 50% chance that two people will have a common birthday
- With 40 people in the room, there is almost 90% chance that two people will have a common birthday

Cryptographic hash functions

You can't just send an unencrypted message and its cryptographic hash to get integrity assurance

- Even if you don't care about confidentiality!

Mallory can change the message and just compute the new message digest herself

Cryptographic hash functions

- Hash functions provide integrity guarantees only when there is a secure way of sending the message digest
  - For example, Bob can publish a hash of his public key (i.e., a message digest) on his business card
  - Putting the whole key on there would be too big
  - But Alice can download Bob's key from the Internet, hash it herself, and verify that the result matches the message digest on Bob's card
- What if there's no external channel to be had?
  - For example, you're using the Internet to communicate

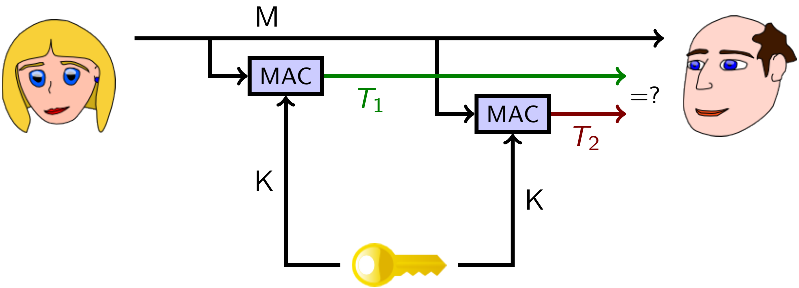
Outline

1. Basics of cryptography
2. Secret-key encryption
3. Public-key encryption
4. Integrity
5. Authentication

Message authentication codes (MAC)

- We do the same trick as for encryption: have a large class of hash functions, and use a shared secret key to pick the "correct" one
- Only those who know the secret key can generate, or even check, the computed hash value (sometimes called a tag)
- These "keyed hash functions" are usually called Message Authentication Codes, or MACs
- Common examples:
  - SHA-1-HMAC, SHA-256-HMAC, CBC-MAC

Message authentication codes (MAC)



Combining ciphers and MACs

In practice we often need both confidentiality and message integrity

- There are multiple strategies to combine a cipher and a MAC when processing a message
  - Encrypt-then-MAC, MAC-then-Encrypt, Encrypt-and-MAC
- Encrypt-then-MAC is the recommended strategy
- Ideally your crypto library already provides an authenticated encryption mode that securely combines the two operations so you don't have to worry about getting it right
  - E.g., GCM, CCM (used in WPA2, see later), or OCB mode

Repudiation

Suppose Alice and Bob share a MAC key K, and Bob receives a message M along with a valid tag T that was computed using K

- Then Bob can be assured that Alice is the one who sent the message M, and that it hasn't been modified since she sent it!
- This is like a "signature" on the message
- But it's not quite the same!
- Bob can't show M and the tag T to Carol to prove Alice sent the message M

Repudiation

- Alice can just claim that Bob made up the message M, and calculated the tag T himself
- This is called repudiation; and we sometimes want to avoid it
- Some interactions should be repudiable
  - Private conversations
- Some interactions should be non-repudiable
  - Electronic commerce

Digital signatures

For non-repudiation, what we want is a true digital signature, with the following properties:

If Bob receives a message with Alice's digital signature on it, then:

- it must be Alice, and not an impersonator, who sent the message (like a MAC)
- the message has not been altered after it was sent (like a MAC),
- Bob can prove these facts to a third party (additional property not satisfied by a MAC).

How do we arrange this?

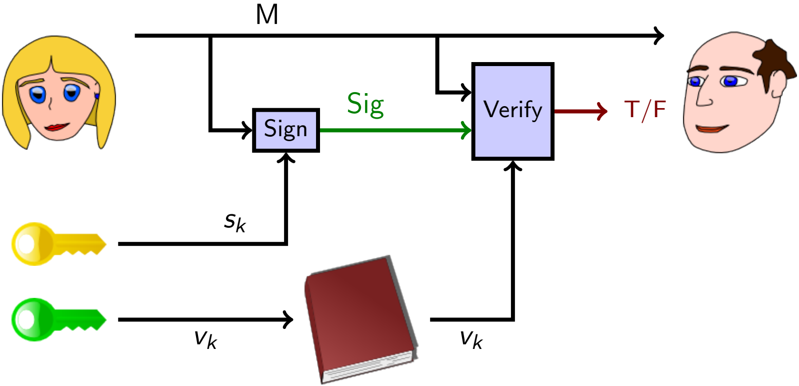
- Use similar techniques to public-key cryptography

Making digital signatures

- Remember public-key cryptosystems:
  - Separate keys for encryption and decryption
  - Give everyone a copy of the encryption key
  - The decryption key is private

- To make a digital signature:
  - Alice signs the message with her private signature key (s_k)
- To verify Alice's signature:
  - Bob verifies the message with Alice's public verification key (v_k)
  - If it verifies correctly, the signature is valid

Making digital signatures



Hybrid signatures

- Just like encryption in public-key cryptosystems, signing large messages is slow
- We can also hybridize signatures to make them faster:
  - Alice sends the (unsigned) message, and also a signature on a hash of the message
  - The hash is much smaller than the message, so it is faster to sign and verify
- Remember that authenticity and confidentiality are separate; if you want both, you need to do both

Combining public-key encryption and digital signatures

- Alice has two different key pairs:
  - an (encryption, decryption) key pair (e_k^A, d_k^A)
  - a (signature, verification) key pair (s_k^A, v_k^A)
- So does Bob: (e_k^B, d_k^B) and (s_k^B, v_k^B)
- Alice uses e_k^B to encrypt a message destined for Bob: $C = E_{e_k^B}(M)$
- She uses s_k^A to sign the ciphertext: $T = Sign_{s_k^A}(C)$
- Bob uses v_k^A to check the signature: $Verify_{v_k^A}(C, T)$, if verified, C is authentic
- He uses d_k^B to decrypt the ciphertext: $M = D_{d_k^B}(C)$
- Similarly for reverse direction

Putting it all together

- We have all these blocks; now what?
- Put them together into protocols
- This is HARD. Just because your pieces all work, doesn't mean what you build out of them will; you have to *use* the pieces correctly: see a counterexample here.
- Common mistakes include:
  - Using the same stream cipher key for two messages
  - Assuming encryption also provides integrity
  - Falling for replay attacks or reaction attacks
  - *LOTS* more!

CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 2 - Cryptography use cases

Spring 2022

Outline

1. Overview
2. WEP: a failure case
3. IPSec
4. TLS
5. WireGuard

Security controls using cryptography

- Q: In what situations might it be appropriate to use cryptography as a security control?
- A: In situations where trust cannot be assumed.

Use cases in program and OS security

- Apps can be installed only if digitally signed by the vendor (BlackBerry) or upgraded only if signed by the original developer (Android)
- OS allows execution of programs only if signed (iOS)
- OS allows loading of certified device drivers only (Windows)
- Secure boot: OS components booted only if correctly signed

Encrypted code

There is research into processors that executes encrypted code only

- The processor will decrypt instructions before executing them
- The decryption key is processor-dependent
- Malware won't be able to spread without knowing a processor's encryption key

Downsides?

Encrypted data

A common technique that aims to protect data in the storage media when the laptop gets lost/stolen, which can be performed either on hardware or by software.

- It often does not protect data against other users who legitimately use laptop
- Or somebody installing malware on laptop
- Or somebody (maybe physically) extracting the decryption key from the laptop's memory

Network security and privacy

Entities you can only communicate with over a network are inherently less trustworthy (e.g., they may not be who they claim to be). This makes networking a primary scenario for cryptography.

This is a separation of concern, and in particular, "*separating the security of the medium from the security of the message*"

Network security and privacy

Cryptography is used at every layer of the network stack for both security and privacy applications:

- Link
  - WEP, WPA, WPA2
- Network
  - VPN, IPsec
- Transport
  - TLS / SSL, Tor
- Application
  - ssh, Mixminion, PGP, OTR, Signal (next class!)

Outline

1 Overview

2 WEP: a failure case

3 IPSec

4 TLS

5 WireGuard

Wired Equivalent Privacy (WEP)

The Wired Equivalent Privacy (WEP) protocol is a link-layer security protocol that aims to *make wireless communication links just as secure as wired links*.

In particular, WEP was intended to enforce three security goals

- Data Confidentiality
  - Prevent an adversary from learning the contents of the wireless traffic
- Data Integrity
  - Prevent an adversary from modifying the wireless traffic or fabricating traffic that looks legitimate
- Access Control
  - Prevent an adversary from using your wireless infrastructure

Unfortunately, none of these is actually enforced!

WEP description

- The sender and receiver share a secret k (either 40 or 104 bits)
- In order to transmit a message M:
  - Compute a checksum $c(M)$
    - this does not depend on k
  - Pick an IV v and generate a keystream $K = RC4(v, k)$
  - Ciphertext $C = K \oplus \langle M \parallel c(M) \rangle$
  - Transmit v and C over the wireless link
- Upon receipt of v and C:
  - Use the received v and the shared s for $K = RC4(v, k)$
  - Decrypt as $K \oplus C = K \oplus K \oplus \langle M' \parallel c' \rangle = M' \parallel c'$
  - Check to see if $c' = c(M')$
  - If it is, accept M' as the message transmitted

Problem 1: key reuse

Keystream is derived as: $K = RC4(v, k)$

- IV (v) is too short: only 3 bytes = 24 bits.
- Secret (k) is rarely changed!

Key-stream gets re-used after 2^{24} iterations → two-time pad.

WEP checksum calculation

The checksum algorithm in WEP is CRC32, which has two important (and undesirable) properties:

- It is independent of k and v
- It is linear: $c(M \oplus D) = c(M) \oplus c(D)$
- Why is linearity a pessimal property for your integrity mechanism to have when used in conjunction with a stream cipher?

Problem 2: integrity breach

If Eve knows C and v in $C = RC4(v, k) \oplus \langle M \parallel c(M)\rangle$

... and Eve wants to modify the plaintext M into $M' = M \oplus \delta$,

... then, all Eve needs to do is

- Calculate $C' = C \oplus \langle \delta \parallel c(\delta)\rangle$
- Send (C', v) instead of (C, v)

Problem 3: packet injection

- What if the adversary wants to inject a new message F onto a WEP-protected network?

- All she needs is a single plaintext/ciphertext pair
- This gives her a value of v and the corresponding keystream $RC4(v, k)$
- Then $C' = \langle F \parallel c(F) \rangle \oplus RC4(v, k)$, and she transmits v, C'
- C' is in fact a correct encryption of F, so the message must be accepted

WEP authentication protocol

- How did the adversary get that single plaintext/ciphertext pair required for the attack on the previous slide?
    - Problem 3: It turns out the authentication protocol gives it to the adversary for free!

- This is a major disaster in the design!
- The authentication protocol (described on the next slide) is supposed to prove that a certain client knows the shared secret k
- But if I watch you prove it, I can turn around and execute the protocol myself!

WEP authentication protocol

- Here's the authentication protocol:
  - The access point sends a challenge string to the client
  - The client sends back the challenge, WEP-encrypted with the shared secret *k*
  - The wireless access point checks if the challenge is correctly encrypted, and if so, accepts the client
- So the adversary has just seen both the plaintext and the ciphertext of the challenge
- Problem number 4: this is enough not only to inject packets (as in the previous attack), but also to execute the authentication protocol itself!

More problems with WEP

- Somewhat surprisingly, the ability to modify and inject packets leads to ways in which Eve can trick the AP to decrypt packets! Check Prof. Goldberg's talk for more details.
- Note that none of the attacks so far use the fact that the stream cipher was RC4. it turns out that when RC4 is used with similar keys, the output keystream has a subtle weakness, which lead the recovery of either a 104-bit or 40-bit WEP key in under 60 seconds, most of the time. Check this paper for more details.

Replacing WEP

Wi-fi Protected Access (WPA) was rolled out as a short-term patch to WEP while formal standards for a replacement protocol (IEEE 802.11i, later called WPA2) were being developed

- Replaces CRC-32 with a real MAC
- IV is 48 bits
- Key is changed frequently (TKIP)
- Ability to use a 802.1x authentication server
  - But maintains a less-secure PSK (Pre-Shared Key) mode for home users
- Ability to run on most older WEP hardware

Replacing WEP

The 802.11i standard was finalized in 2004, and the result (called WPA2) has been required for products calling themselves "Wi-fi" since 2006

- Replaces the RC4 and MAC algorithms in WPA ith the CCM authenticated encryption mode (using AES)
- Considered strong, except in PSK mode
  - Dictionary attacks still possible (avoided in WPA3 (2018))

Outline

1. Overview
2. WEP: a failure case
3. IPSec
4. TLS
5. WireGuard

Network layer security: purpose

Suppose every link in our network had strong link-layer security. Why would this not be enough?

- Source, destination IPs may not share the same link. Network layer threats such as IP spoofing still exist.
- We need end-to-end security across networks, i.e., securing network layer packets from one host to another so that routers or other hosts in the middle cannot modify or read the packet payload (they still need to read packet metadata for routing)

The IP Security suite (IPSec) extends the Internet Protocol (IP) to provide confidentiality and integrity of packets transmitted across the network. IPSec enables various architectures of Virtual Private Networks (VPNs) which is the foundation in network-layer security.

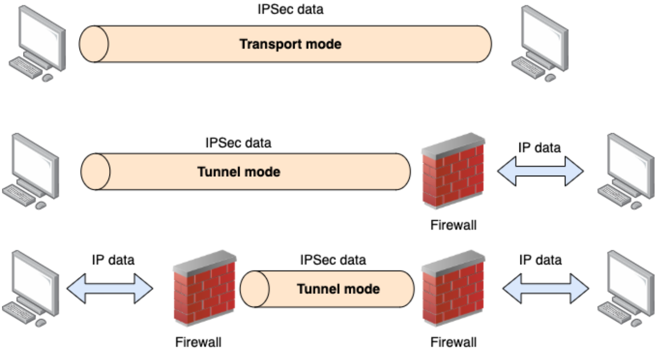
Internet Key Exchange (IKE)

The source and destination IP addresses agree on a shared symmetric key via the IKE process, which internally uses the Diffie-Hellman protocol:

- Alice chooses prime p at random and finds a generator g
- Alice chooses $X \leftarrow_{\mathrm{R}} \{0, 1, \ldots, p-2\}$ and sends $A = g^X \pmod{p}$ to Bob, together with p and g
- Bob chooses $Y \leftarrow_{\mathrm{R}} \{0, 1, \ldots, p-2\}$ and sends $B = g^Y \pmod{p}$ to Alice
- Alice and Bob both compute $s = g^{XY} \pmod{p}$
    - Alice does that by computing $B^X \pmod{p}$
    - Bob does that by computing $A^Y \pmod{p}$
- Now they share a common secret s which can be used to derive a symmetric key

Modes of operation

- IPSec has two main modes of operation:
  - Transport mode: uses the original IP header
  - Tunnel mode: encapsulates the original IP header



IPSec Headers

Authentication Header (AH) – RFC4302

- Offers integrity and data source authentication
  - Authenticates payload and parts of IP header that do not get modified during transfer, e.g., source IP address
- Offers protection against replay attacks
  - Via extended sequence numbers

Encapsulated Security Payload (ESP) – RFC4303

- Offers confidentiality
  - IP data is encrypted during transmission
- Offers authentication functionality similar to AH
  - But authenticity checks only focus on the IP payload
- Applies padding and generates dummy traffic
  - Makes traffic analysis harder (more on this on an upcoming lecture!)

Authentication Header (AH)

```
      0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |  Ver. |  IHL  |    DHCP   |ECN|         Total Length          |
| I +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| P |        Identification         |Flags|    Fragment Offset      |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| H | Time to Live  |   Protocol    |       Header Checksum         |
| E +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| A |                      Source IP Address                        |
| D +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| E |                   Destination IP Address                      |
| R +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                 Optional Fields (variable)                    |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |  Next Header  |  Payload Len  |          RESERVED             |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| A |                Security Parameters Index (SPI)                |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| H |                    Sequence Number Field                      |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |          ** Integrity Check Value-ICV (variable) **           |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| P |                 Packet Payload (variable)                     |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Encapsulating Security Payload (ESP)

```
      0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |  Ver. |  IHL  |    DHCP   |ECN|         Total Length          |
| I +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| P |        Identification         |Flags|    Fragment Offset      |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| H | Time to Live  |   Protocol    |       Header Checksum         |
| E +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| A |                      Source IP Address                        |
| D +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| E |                   Destination IP Address                      |
| R +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                 Optional Fields (variable)                    |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                Security Parameters Index (SPI)                |
| E +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                    Sequence Number Field                      |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S |                Payload Data + Padding (variable)              |
|   |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                               |  Pad Length   |  Next Header  |
| P +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |          ** Integrity Check Value-ICV (variable) **           |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPSec packets' format

A regular IP packet in the form of $\langle$ H $\parallel$ P $\rangle$ can be transformed into an IPSec packet depending on the mode of operation:

	AH	**ESP**
Transport	H $\parallel$ AH $\parallel$ P $\hookrightarrow$ Int. of H and P	H $\parallel$ ESP-H $\parallel$ $\langle$ P $\rangle_k$ $\parallel$ ESP-T $\hookrightarrow$ Int. and Conf. of P only
Tunnel	H' $\parallel$ AH $\parallel$ $\langle$ H $\parallel$ P $\rangle$ $\hookrightarrow$ Int. of H and P	H' $\parallel$ ESP-H $\parallel$ $\langle$ H $\parallel$ P $\rangle_k$ $\parallel$ ESP-T $\hookrightarrow$ Int. and Conf. of H and P

The Tunnel-ESP combination (also known as an IP-in-IP tunneling) is often used to implement Virtual Private Networks (VPNs)

IPSec deployment challenges

- Needs to be included in the kernel's network stack.
- There may be legitimate reasons to modify some IP header fields; IPSec breaks networking functionalities that require such changes.
  - with AH, you cannot replace a private address for a public one at a NAT box.
  - with ESP, it depends
    - In transport usually does not work due to TCP and UDP checksums
    - In tunnel mode it is fine

Outline

1. Overview
2. WEP: a failure case
3. IPSec
4. TLS
5. WireGuard

Transport-layer security and privacy

- Network-layer security mechanisms arrange to send *individual* IP packets securely from one network to another
- Transport-layer security mechanisms transform arbitrary TCP connections to add security and privacy
- The main transport-layer security mechanism:
  - TLS (formerly known as SSL)
- The main transport-layer privacy mechanism:
  - Tor — will be covered in the lecture on PETs

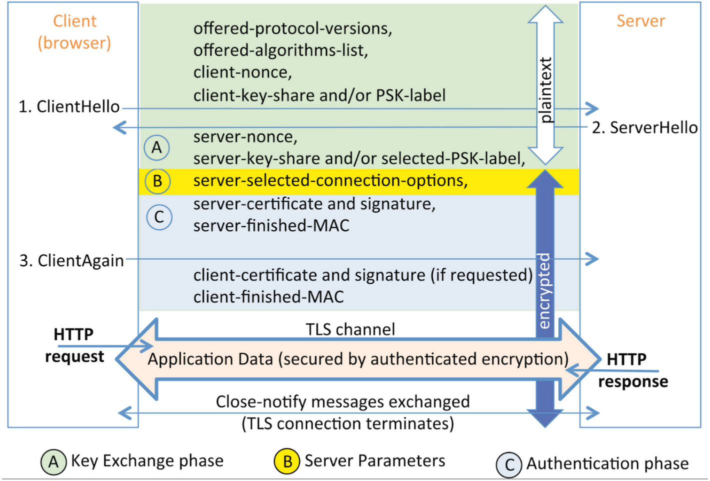
TLS / SSL

- In the mid-1990s, Netscape invented a protocol called Secure Sockets Layer (SSL) meant for protecting HTTP (web) connections
  - The protocol, however, was general, and could be used to protect any TCP-based connection
  - HTTP + SSL = HTTPS
- Historical note: there was a competing protocol called S-HTTP. But Netscape and Microsoft both chose HTTPS, so that's the protocol everyone else followed
- SSL went through a few revisions, and was eventually standardized into the protocol known as TLS (Transport Layer Security, imaginatively enough)

TLS at a high level: RFC8446

- Client connects to server, indicates it wants to speak TLS, with
  - Client key-share under ECDHE
  - The list of ciphersuites it knows
- Server sends its certificate to client, which contains:
  - Server key-share under ECDHE
  - Its host name
  - Its verification key
  - Some other administrative information
  - A signature from a Certificate Authority (CA)
- Both client and server derives the same session key K (which is hard for Eve to derive) based on the two key shares
- Server also chooses which ciphersuite to use
- All remaining traffic will be encrypted and authenticated under K

TLS connection establishment


Client
(browser)
Server
1. ClientHello
offered-protocol-versions,
offered-algorithms-list,
client-nonce,
client-key-share and/or PSK-label
plaintext
2. ServerHello
A
server-nonce,
server-key-share and/or selected-PSK-label,
B
server-selected-connection-options,
C
server-certificate and signature,
server-finished-MAC
3. ClientAgain
client-certificate and signature (if requested)
client-finished-MAC
encrypted
HTTP
request
TLS channel
Application Data (secured by authenticated encryption)
HTTP
response
Close-notify messages exchanged
(TLS connection terminates)
A Key Exchange phase
B Server Parameters
C Authentication phase

Security properties provided by TLS

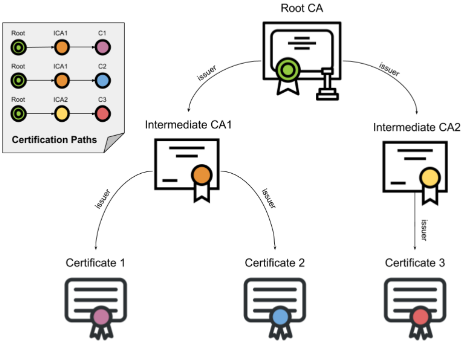
- Server authentication
- Message integrity
- Message confidentiality
- Client authentication (optional)

Why is client authentication mostly optional?

Certificate Authorities (CAs) in TLS

A certificate authority acts as a trusted third-party that:

- Issues digital certificates
- Certifies the ownership of a public key by the named subject of the certificate
- Manages certificate revocation lists (CRLs)



What can go wrong with TLS?

It is possible to man-in-the-middle TLS:

- An adversary can compromise a CA to plant fake certificates
  - e.g., DigiNotar's fake *.google.com certificates used by an ISP in Iran
- An adversary can install a custom CA on users' devices, allowing them to sign certificates that clients will accept for any site
  - e.g., in 2019, Kazakhstan's ISPs mandated the installation of a root certificate issued by the government
- Companies may think it is an excellent idea
  - e.g., Lenovo's Superfish or Sennheiser HeadSetup root certificates
    - for advertisement and communication purposes, respectively

SSL-based VPNs

- We can use SSL/TLS to create secure site-to-site tunnels
  - Similarly to IPSec
- A more flexible “user-space VPN”
  - In contrast to IPSec, it does not require kernel-level access
  - Virtual network interfaces are used instead
- Several solutions available:
  - e.g., OpenVPN, Cisco AnyConnect

Outline

1. Overview
2. WEP: a failure case
3. IPSec
4. TLS
5. WireGuard

Issues with existing VPNs

IPSec:

- Is complex, hard to audit, and prone to misconfigurations
  - “IPSec is too complex to be secure” (Schneier and Ferguson, ’99)
  - Big book of IPSec RFCs: Internet security architecture (Loshin, ’99)
- Does not prevent you from making bad choices
  - Supports all ciphers, including obsolete ones and NULL

SSL VPNs:

- Also on the complex side
  - Full TLS stack implementation
- Tend to be slow
  - Penalty of running over TCP
  - Must copy packets in and out of userspace
- Also does not prevent you from making bad choices

WireGuard

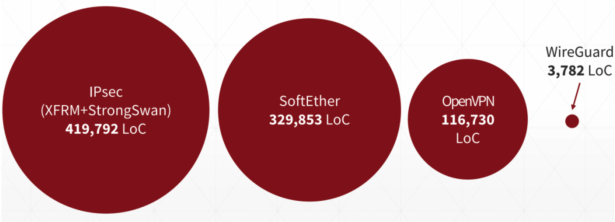
- New (and simpler) VPN design built from the ground-up
- Offers a kernel and a user-space implementation
- Faster than IPSec and TLS-based VPN solutions

WIREGUARD

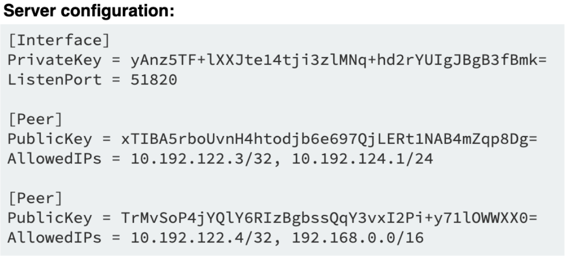
FAST, MODERN, SECURE VPN TUNNEL

Lightweight and secure

- Easy to configure
  - But no PKI, keys are distributed manually
- Easy to audit
  - 4,000 LoCs vs IPSec's 400,000 LoCs
- Hard to get it wrong
  - Single cipher suite



Cryptokey Routing

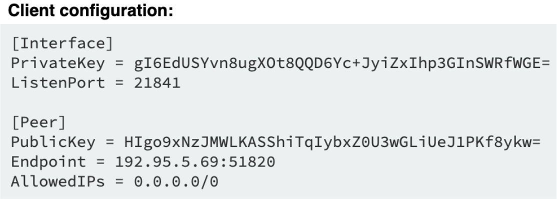


Server configuration:

```
[Interface]
PrivateKey = yAnz5TF+lXXJte14tji3zlMNq+hd2rYUIgJBgB3fBmk=
ListenPort = 51820

[Peer]
PublicKey = xTIBA5rboUvnH4htodjb6e697QjLERt1NAB4mZqp8Dg=
AllowedIPs = 10.192.122.3/32, 10.192.124.1/24

[Peer]
PublicKey = TrMvSoP4jYQlY6RIzBgbssQqY3vxI2Pi+y71lOWWXX0=
AllowedIPs = 10.192.122.4/32, 192.168.0.0/16
```



Client configuration:

```
[Interface]
PrivateKey = gI6EdUSYvn8ugXOt8QQD6Yc+JyiZxIhp3GInSWRfWGE=
ListenPort = 21841

[Peer]
PublicKey = HIgo9xNzJMWLKASShiTqIybxZ0U3wGLiUeJ1PKf8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0
```

- When sending: allowed IPs behaves like a routing table
- When receiving: allowed IPs behave like an access control list

Are we safe now?

Network layer encryption:

- Implemented between network sources and destinations
  - Not necessarilly end-to-end - Many times applied to portions of a network
- Makes extensive use of encapsulation
  - e.g., encapsulation of IP packets in IPSec

Can we push encryption closer to the source and decryption closer to the destination?

CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 3 - Application layer security

Spring 2022

Outline

1. Application layer security
2. SSH
3. PGP
4. OTR
5. Signal

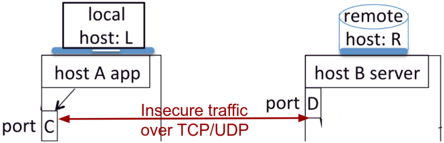
Application layer security

- TLS can provide for encryption at the TCP socket level
  - "End-to-end" in the sense of a network connection
  - Is this good enough? Hint: one application may involve multiple TCP connections
- Many applications would like true end-to-end security
  - Human-to-human would be best, but those last 50 cm are really hard!
  - We usually content ourselves with desktop-to-desktop
- We'll look at three particular applications:
  - SSH, PGP, and instant messaging

Outline

1. Application layer security
2. SSH
3. PGP
4. OTR
5. Signal

Insecure application traffic pre-SSH



- Suppose that you want to connect to a remote machine
    - You may think “Oh ok, let me use Telnet”
- Think again...
    - All data exchanged through Telnet is in plain text!

Enter secure remote login (ssh)

- You're already familiar with this tool for securely logging in to a remote computer (the ugster machines)
- Usual usage (simplified):
  - Client connects to server
  - Server sends its verification key
    - The client should verify that this is the correct key
  - Client and server run a key agreement protocol to establish session keys, server signs its messages
    - All communication from here on in is encrypted and MAC-ed with the session keys
  - *Client authenticates to server*
  - Server accepts authentication, login proceeds

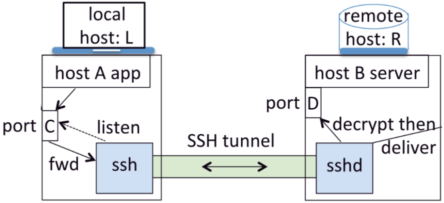
User authentication with ssh

There are two main ways to authenticate with ssh:

- Send a password over the encrypted channel
  - The server needs to know (a hash of) your password
- Sign a random challenge with your private signature key
  - The server needs to know your public verification key

Which is better? Why?

SSH port forwarding



SSH allows for tunneling:

- The client machine can create a mapping between a local TCP port and a port in the remote machine
    - e.g., localhost:IMAP to mail.myorg.ca:IMAP
- The client SSH and the server SSHd operate as a secure relay
    - Allows the client to interact with server applications via SSH

Outline

1. Application layer security
2. SSH
3. PGP
4. OTR
5. Signal

Pretty Good Privacy

- The first popular implementation of public-key cryptography.
- Originally made by Phil Zimmermann in 1991
  - He got in a lot of trouble for it, since cryptography was highly controlled at the time.
  - But that's a whole 'nother story. :-)
- Today, there are many (more-or-less) compatible programs
  - GNU Privacy Guard (gpg), Hushmail, etc.

Pretty Good Privacy

- What does it do?
  - Its primary use is to protect the contents of email messages
- How does it work?
  - Uses public-key cryptography to provide:
    - Encryption of email messages (using hybrid encryption)
    - Digital signatures on email messages (hash-then-sign)

Recall on public-key cryptography

- In order to use public-key encryption and digital signatures, Alice and Bob must each have:
  - A public encryption key
  - A private decryption key
  - A private signature key
  - A public verification key

Sending a message

- To send a message to Bob, Alice will:
  - Write a message
  - Sign it with her own signature key
  - Encrypt both the message and the signature with Bob's public encryption key
- Bob receives this, and:
  - Decrypts it using his private decryption key to yield the message and the signature
  - Uses Alice's verification key to check the signature

Back to PGP

PGP's main functions:

- Create these four kinds of keys
  - encryption, decryption, signature, verification
- Encrypt messages using someone else's encryption key
- Decrypt messages using your own decryption key
- Sign messages using your own signature key
- Verify signatures using someone else's verification key
- Sign other people's keys using your own signature key

Obtaining keys

Earlier, we said that Alice needs to get an authentic copy of Bob's public key in order to send him an encrypted message.

How does Alice do this?

- Certificate authorities (CAs)?

What if we don't involve CAs?

- Bob could put a copy of his public key on his webpage
  - Is this good enough?

Verifying public keys

- If Alice knows Bob personally, she could:
  - Download the key from Bob's web page
  - Phone up Bob, and verify she's got the right key
  - Problem: keys are big and unwieldy!

```
mQGiBDi5qEURBADitpDzvvzW+9lj/zYgK78G3D76hvvvIT6gpTIlwg6WIJNLKJat
01yNpMIYNvpwi7EUd/lSNl6t1/A022p7s7bDbE4T5NJda0IOAgWeOZ/plIJC4+o2
tD2RNuSkwDQcxzm8KUNZOJla4LvgRkm/oUubxyeY5omus7hcfNrBOwjC1wCg4Jnt
m7s3eNfMu72Cv+6FzBgFog8EANirkNdC1Q8oSMDihWj1ogiWbBz4s6HMxzAaqNf/
rCJ9qoK5SLFeoB/r5ksRWty9QKV4VdhhCIy1U2B9tSTlEPYXJHQPZ3mwCxUnJpGD
8UgFM5uKXaEq2pwpArTm367k0tTpMQgXAN2HwiZv//ahQXH4ov30kBBVL5VFxMUL
UJ+yA/4r5HLTpP2SbbqtPWdeW7uDwhe2dTqffAGuf0kuCpHwCTAHr83ivXzT/7OM
```

Fingerprints

- Luckily, there's a better way!
- A fingerprint is a cryptographic hash of a key
- This, of course, is *much shorter*:
  - B117 2656 DFF9 83C3 042B C699 EB5A 896A 2898 8BF5
- Remember: there's no (known) way to make two different keys that have the same fingerprint, provided that we use a collision-resistant hash function

Fingerprints

- So now we can try this:
  - Alice downloads Bob’s key from his webpage
  - Alice’s software calculates the fingerprint
  - Alice phones up Bob, and asks him to read his key’s actual fingerprint to her
  - If they match, Alice knows she’s got an authentic copy of Bob’s key
- That’s great for Alice, but what about Carol?
  - Carol might not know Bob
  - At least not well enough to phone him

Signing keys

- Once Alice has verified Bob's key, she uses her signature key to sign Bob's key
- This is effectively the same as Alice signing a message that says "I have verified that the key with fingerprint `B117 2656 DFF9 83C3 042B C699 EB5A 896A 2898 8BF5` really belongs to Bob"
- Bob can attach Alice's signature to the key on his webpage
  - If Bob wants, he can get many people to sign his key...

Can you see some potential issue with key signing?

Web of Trust

- Now Alice can act as an introducer for Bob
- If Carol doesn't know Bob, but does know Alice (and has already verified Alice's key, and trusts her to introduce other people):
  - she downloads Bob's key from his website
  - she sees Alice's signature on it
  - she is able to use Bob's key without having to check with Bob personally
- This is called the Web of Trust, and the PGP software handles it mostly automatically

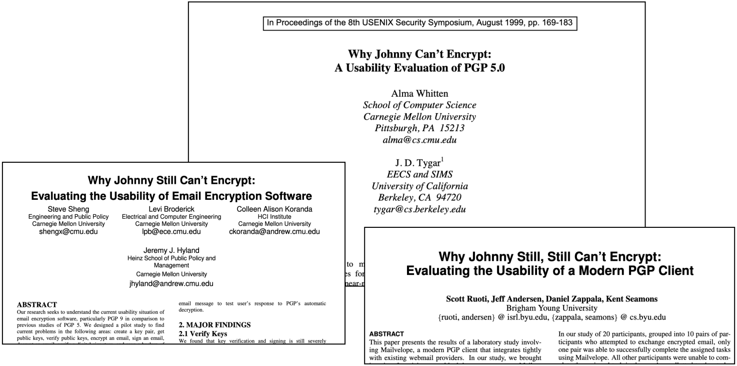
So, great!

So if Alice and Bob want to have a private conversation by email:

- They each create their sets of keys
- They exchange public encryption keys and verification keys
- They send signed and encrypted messages back and forth

Pretty Good, no?

Problem 1: Usability



In Proceedings of the 8th USENIX Security Symposium, August 1999, pp. 169-183

Why Johnny Can't Encrypt:
A Usability Evaluation of PGP 5.0

Alma Whitten
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alma@cs.cmu.edu

J. D. Tygar[1]
EECS and SIMS
University of California
Berkeley, CA 94720
tygar@cs.berkeley.edu

Why Johnny Still Can't Encrypt:
Evaluating the Usability of Email Encryption Software

Steve Sheng
Engineering and Public Policy
Carnegie Mellon University
shengx@cmu.edu

Levi Broderick
Electrical and Computer Engineering
Carnegie Mellon University
lpb@ece.cmu.edu

Colleen Alison Koranda
HCI Institute
Carnegie Mellon University
ckoranda@andrew.cmu.edu

Jeremy J. Hyland
Heinz School of Public Policy and Management
Carnegie Mellon University
jhyland@andrew.cmu.edu

ABSTRACT
Our research seeks to understand the current usability situation of email encryption software, particularly PGP 9 in comparison to previous studies of PGP 5. We designed a pilot study to find current problems in the following areas: create a key pair, get public keys, verify public keys, encrypt an email, sign an email,

email message to test user's response to PGP's automatic decryption.

2. MAJOR FINDINGS
2.1 Verify Keys
We found that key verification and signing is still severely

Why Johnny Still, Still Can't Encrypt:
Evaluating the Usability of a Modern PGP Client

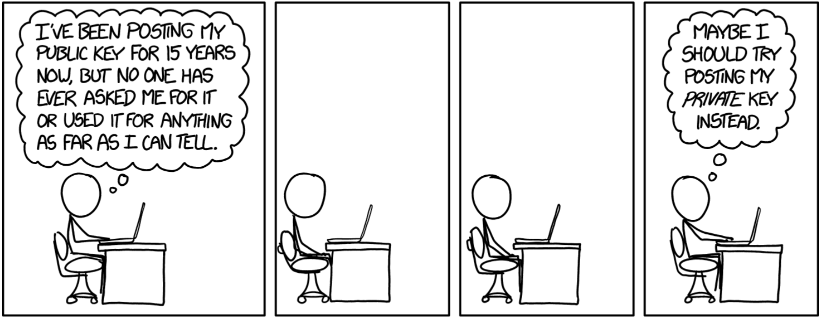
Scott Ruoti, Jeff Andersen, Daniel Zappala, Kent Seamons
Brigham Young University
{ruoti, andersen} @ isrl.byu.edu, {zappala, seamons} @ cs.byu.edu

ABSTRACT
This paper presents the results of a laboratory study involving Mailvelope, a modern PGP client that integrates tightly with existing webmail providers. In our study, we brought

In our study of 20 participants, grouped into 10 pairs of participants who attempted to exchange encrypted email, only one pair was able to successfully complete the assigned tasks using Mailvelope. All other participants were unable to com-

- Common mistakes:
  - Encrypt a message with the sender's public key
  - Send private key so that recipient can decrypt a message
- Oftentimes, study participants cannot send a PGP-encrypted e-mail after 45min

Problem 1: Usability


I'VE BEEN POSTING MY PUBLIC KEY FOR 15 YEARS NOW, BUT NO ONE HAS EVER ASKED ME FOR IT OR USED IT FOR ANYTHING AS FAR AS I CAN TELL.
MAYBE I SHOULD TRY POSTING MY *PRIVATE* KEY INSTEAD.

(Public Key)

Problem 2: Key compromise

- Suppose (encrypted) communications between Alice and Bob are recorded by the “bad guys”
  - criminals, competitors, etc
- Later, Bob’s computer is stolen by the same bad guys
- Or just broken into
  - Virus, trojan, etc

All of Bob’s key material is recovered

The bad guys can...

- Decrypt past messages
- Learn their content
- Learn that Alice sent them
- And have a mathematical proof they can show to anyone else!

How private is that?

What went wrong?

- Bob's computer got stolen?
- How many of you have never...
  - Left your laptop unattended?
  - Not installed the latest patches?
  - Run software with a remotely exploitable bug?
- What about your friends?

What really went wrong

- PGP creates lots of incriminating records:
  - Key material that decrypts data sent over the public Internet
  - Signatures with proofs of who said what
- Alice had better watch what she says!
  - Her privacy depends on Bob's actions

Outline

1. Application layer security
2. SSH
3. PGP
4. OTR
5. Signal

Casual conversations

- Alice and Bob talk in a room
- No one else can hear
    - Unless being recorded
- No one else knows what they say
    - Unless Alice or Bob tells them
- No one can prove what was said
    - Not even Alice or Bob

These conversations are “off-the-record” (OTR)

We like off-the-record conversations

- Legal support for having them
  - Illegal to record other people’s conversations without notification
- We can have them over the phone
  - Illegal to tap phone lines
- But what about over the Internet?

Cryptographic tools

We have the cryptographic tools to do OTR, but we need to have new perspectives on how to use these tools:

- We want perfect forward secrecy
- We want deniable authentication

Perfect forward secrecy

- Future key compromises should not reveal past communication
- Use secret-key encryption with a short-lived key (a session key)
- The session key is created by a modified Diffie-Hellman protocol
- Discard the session key after use
  - Securely erase it from memory (and everywhere possible)
- Use long-term keys only to authenticate the Diffie-Hellman protocol messages only

Future secrecy (a.k.a. post-compromise security)

- Past key compromises should not compromise the security of future sessions
  - What happens if new session keys are just hashes of the previous key?
  - e.g., derived through a session key ratchet:
    - K2 = H(K1), K3 = (H(H(K1))) = H(K2)
- So what can we do?
  - Regularly replace potentially compromised session keys with new key material

Deniable authentication

- Do not want digital signatures
  - Non-repudiation is great for signing contracts, but undesirable for private conversations
- But we do want authentication
  - We can't maintain privacy if attackers can impersonate our friends
- Use Message Authentication Codes (MAC)
  - We talked about these earlier

No third-party proofs

- Shared-key authentication
  - Alice and Bob have the same K
  - K is required to compute the MAC
  - How is Bob assured that Alice sent the message?
- Bob cannot prove that Alice generated the MAC
  - He could have done it, too
  - Anyone who can verify can also forge
- This gives Alice a measure of deniability

Using these techniques

Using these techniques, we can make our online conversations more like face-to-face “off-the-record” conversations.

But there is a wrinkle:

- These techniques require the parties to communicate interactively
- This makes them unsuitable for email
- But they’re still great for instant messaging!

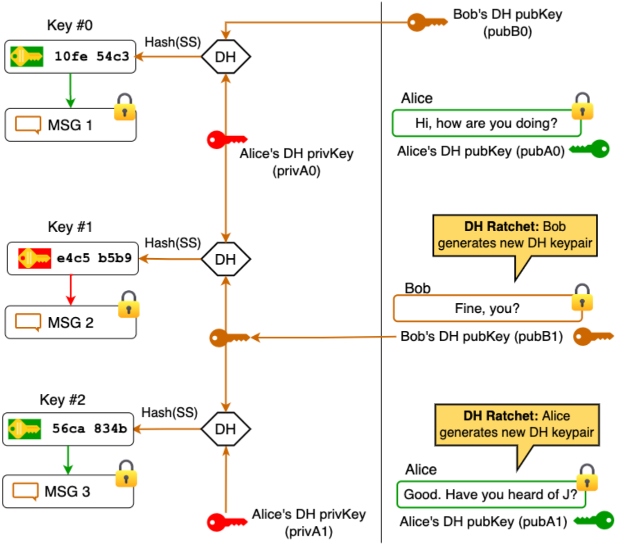
Off-the-Record Messaging

- Perfect Forward Secrecy
  - Shortly after Bob receives the message, it becomes unreadable to anyone, anywhere (provided the key is erased securely)
- Deniability
  - Although Bob is assured that the message came from Alice, he can't convince Carol of that fact
  - Also, Carol can create forged transcripts of conversations that are every bit as accurate as the real thing

A closer look at OTR's DH Ratchet

- Achieves perfect forward secrecy by making session keys roll forward.
  - Ratchet: a device that allows movement in a single direction
  - Diffie-Hellman key exchange ratchet
- DH keys "ping-pong":
  - Alice sends message 1 to Bob, encrypted with key 0
  - This message includes a DH value to create key 1
  - Bob decrypts message 1 and deletes key 0
  - Bob messages back Alice with message 2, encrypted with key 1
  - ...

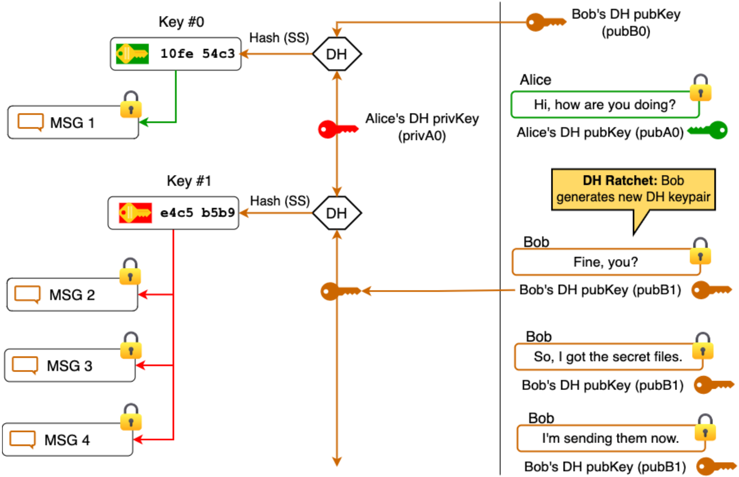
OTR's DH Ratchet (visually)



Issues with OTR's DH Ratchet

- Session keys only roll forward with interactive replies.
- What happens if Alice sends multiple messages but Bob takes a long time to respond?

Issues with OTR’s DH Ratchet



- Multiple messages get encrypted with the same key!
- Forward secrecy is only partially provided

Outline

1. Application layer security
2. SSH
3. PGP
4. OTR
5. Signal

Signal Protocol

- Signal is an app for iOS, Android, and Chrome
  - Original protocol based on OTR and used for encrypted SMS (e.g., Google Messages)
- The Signal Protocol is now used by other apps like WhatsApp
  - Also optionally in Facebook Messenger and Skype
  - Why on Earth would you like to always keep your conversations private, right? :-)

Signal

Signal Protocol

- Provides perfect forward secrecy
  - Similar to OTR, uses a “ratchet” technique to constantly rotate session keys
- Provides future secrecy (or “post-compromise security”)
  - A leak of past or long-term keys will be healed by introducing new DH ratchet keys
- Provides improved deniability
  - Uses “Triple Diffie-Hellman” deniable authenticated key exchange
- Supports out-of-order message delivery
  - Users can store per-message keys until late messages arrive

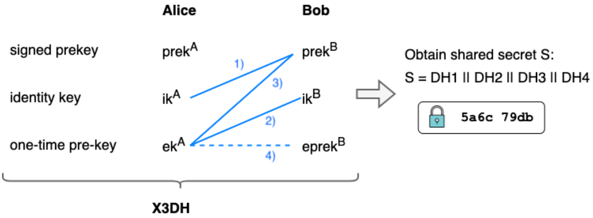
Double Ratchet (a.k.a. “Axolotl”)

- Signal combines assymetric and symmetric key ratchets
  - Establish a shared secret
  - Use a DH ratchet whenever parties take turns in exchanging messages
  - Use a symmetric key ratchet between consecutive messages
- Interesting properties:
  - Generates ephemeral per-message keys (forward and future secrecy)
  - Tolerates message loss/re-ordering
- Let’s take a closer look...

Registration Stage

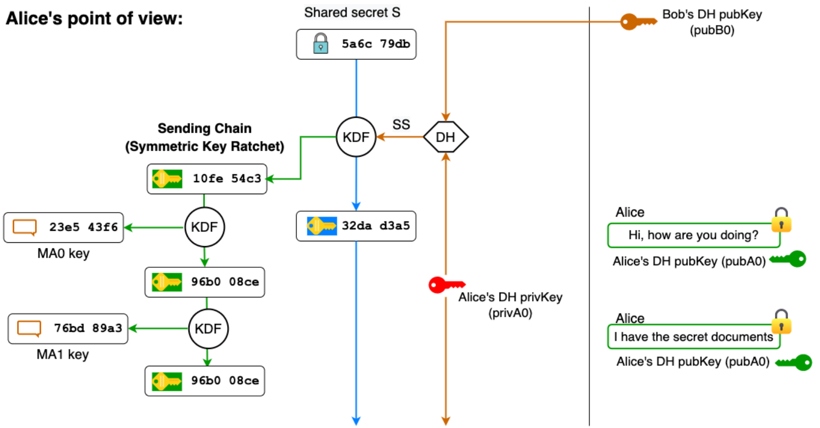
- Users generate a number of cryptographic keys and register themselves on a key distribution server
- Each user (e.g., Alice) generates the following DH private keys:
  - a long-term identity key (ik^A)
  - a medium-term signed prekey ($prek^A$)
  - multiple short-term “one-time” prekeys (ek^A)
- The public keys corresponding to these private keys are uploaded to the server
  - This is the “pre-key bundle”
  - What kinds of keys are these?
- For authentication, Alice and Bob should verify each other’s identity keys out-of-band

Extended Triple Diffie-Hellman (X3DH) Key Agreement



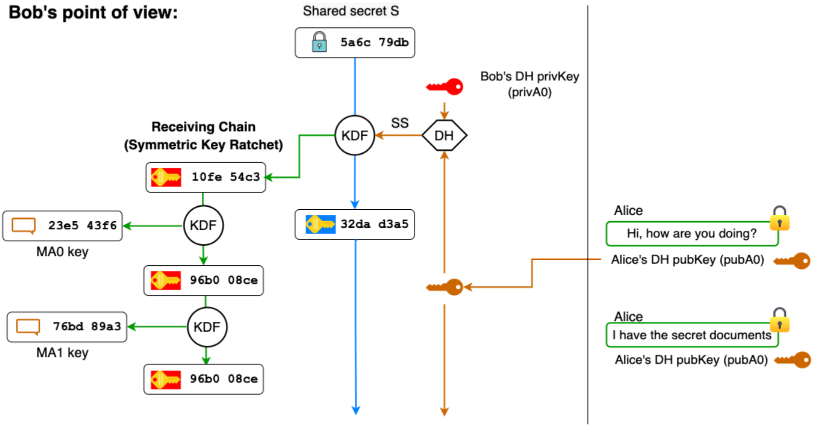
- X3DH outputs a master secret (S), used to:
  - Establish a common root key
  - Generate new ephemeral sending and receiving chain keys through the application of a key derivation function (KDF)
- X3DH is executed at first contact, device change, or app re-install since the identity key will change
- 1) and 2) offer mutual authentication (due to *ik*), while 3) and 4) provide forward secrecy (unique to this exchange)

Exchanging Messages (Alice to Bob)



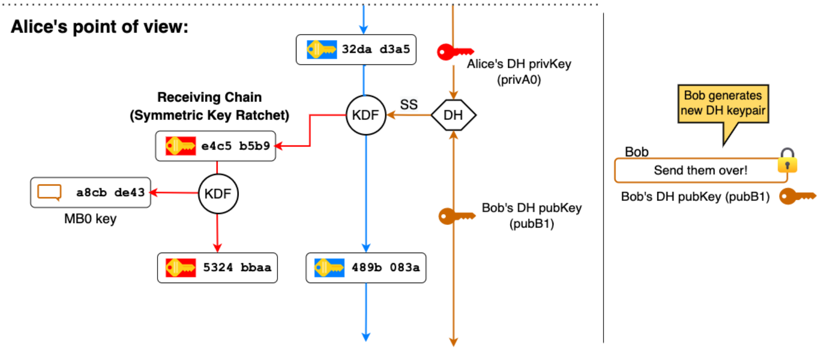
- The DH key exchange produces a shared secret (SS)
  - SS is used to derive a new root key and a sending chain key
- This new key is used to derive new chain keys and message keys

Exchanging Messages (Alice to Bob)



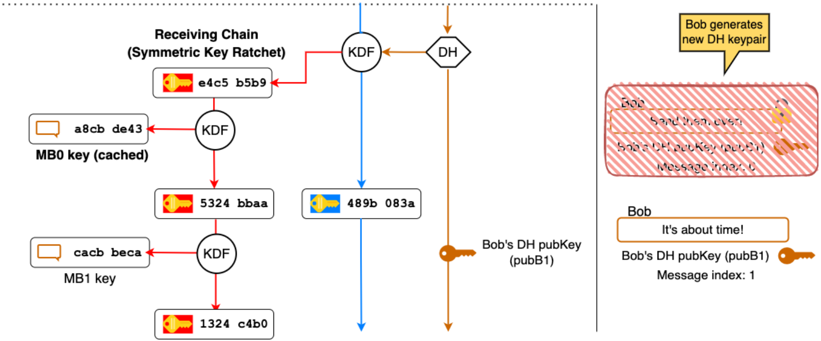
- Alice's DH public key (pubA0) allows Bob to derive the same shared secret
- Bob produces a mirrored version of Alice's sending chain

Exchanging Messages (Bob to Alice)



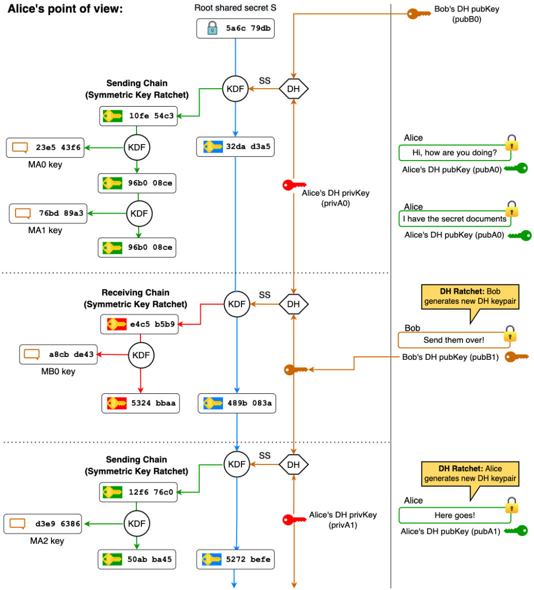
- Bob generates a new set of DH keys before replying to Alice
- Alice produces a mirrored version of Bob's sending chain

Exchanging Messages with losses/reorder (Bob to Alice)



- What if a message is lost along the way?
  - Alice can cache the MB0 key and advance the symmetric ratchet
  - Message MB0 can later be decrypted on arrival
- For how long should Alice cache her receiving chain's keys?

Exchanging Messages (the full picture)


Alice's point of view:
Root shared secret S
5a6c 79db
Bob's DH pubKey (pubB0)
Sending Chain (Symmetric Key Ratchet)
KDF
SS
DH
10fe 54c3
23e5 43f6
MA0 key
KDF
32da d3a5
96b0 08ce
Alice
Hi, how are you doing?
Alice's DH pubKey (pubA0)
Alice's DH privKey (privA0)
76bd 89a3
MA1 key
KDF
96b0 08ce
Alice
I have the secret documents
Alice's DH pubKey (pubA0)
Receiving Chain (Symmetric Key Ratchet)
KDF
SS
DH
DH Ratchet: Bob generates new DH keypair
e4c5 b5b9
a8cb de43
KDF
MB0 key
Bob
Send them over!
Bob's DH pubKey (pubB1)
5324 bbaa
489b 083a
Sending Chain (Symmetric Key Ratchet)
KDF
SS
DH
DH Ratchet: Alice generates new DH keypair
12f6 76c0
Alice
d3e9 6386
KDF
MA2 key
Here goes!
Alice's DH privKey (privA1)
Alice's DH pubKey (pubA1)
50ab ba45
5272 befe

Recap

End-to-end security at the application layer:

- Only the communicating end parties can decrypt and read exchanged messages
- Still not the case everywhere, but good progress is being made

CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 4 - Privacy-enhancing technologies (PETs)

Spring 2022

Outline

1. What is privacy?
2. Remailers
3. Tor
4. Private information retrieval (PIR)

Turtles, locks, bathrooms and more!

Are the many formal definitions and frameworks of privacy consistent with a layperson's understanding of privacy?

Paper on PoPETs 2018: Turtles, Locks, and Bathrooms: Understanding Mental Models of Privacy Through Illustration

Asked people of different ages in the US to draw a diagram on what privacy means to them, and here are a few illustrations:

Privacy as turtles

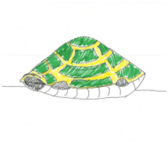


Fig. 63. "It's a turtle huddled up inside its shell." By John

PEARL OYSTERS HAVE SOMETHING VALUABLE
TO PROTECT - THE PEARL.
THEY CAN DO SO BY SIMPLY 'CLOSING THE LID'
IF ONLY SAFEGUARDING THE DATA IN MY
LAPTOP WERE THAT SIMPLE!

Fig. 62. "Pearl oysters have something valuable to protect - the pearl. They can do so by simply 'closing the lid.' If only safeguarding the data in my laptop were that simple!" By Sharon, age 25.

Fig. 40. "A shield that protects me." By HAP, age 24

[0] All pictures from the PoPETs'18 paper

Privacy as locks

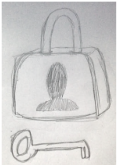


Fig. 10. "To me, privacy is fundamentally about feeling secure. Having the ability to control who has access to me, and to my information, makes me feel like I can control my privacy." By CJ, age 33

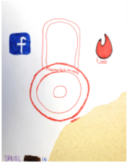


Fig. 11. By Daniel, age 16

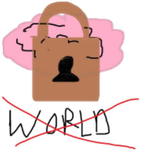


Fig. 33. "Privacy means that the thoughts in my brain are locked away. What I know does not have to go into the world, which I put an X over." By Thomas, age 19

[0] All pictures from the PoPETs'18 paper

Privacy as locks

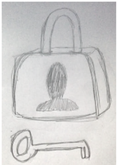


Fig. 10. "To me, privacy is fundamentally about feeling secure. Having the ability to control who has access to me, and to my information, makes me feel like I can control my privacy." By CJ, age 33

Fig. 11. By Daniel, age 16

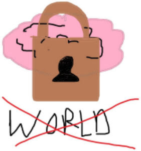


Fig. 33. "Privacy means that the thoughts in my brain are locked away. What I know does not have to go into the world, which I put an X over." By Thomas, age 19

- "Intellectual privacy is about needing to have protections from being watched and interfered with when we're making up our minds about the world – when we're reading, surfing the Web, talking on the phone, and sending e-mail to confidants." – Neil Richards

[0] All pictures from the PoPETs'18 paper

Privacy as bathrooms

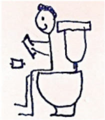


Fig. 23. "This is me enjoying my privacy. This is the only time during the day, were I am truly alone and nothing bothers me. No man no children no dogs." By Cindy, age 54

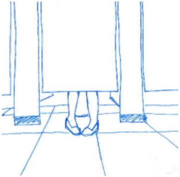


Fig. 38. By Rachel, age 20

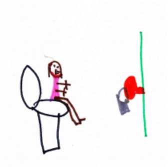


Fig. 24. "No one come in when I am in the bathroom!" By Sydney, age 7

[0] All pictures from the PoPETs'18 paper

Privacy as filters

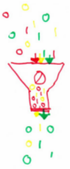


Fig. 45. "Green data (non-private) goes through; red does not (private data). Some yellow goes through (ambiguous)." By Ryan, age 36

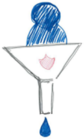


Fig. 58. "Privacy is to me the ability to filter and control the information relevant to you that you release into the world (and having some confidence in the ability of the status of such information as private)." By Isadora, age 20

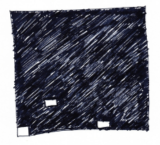


Fig. 74. "Privacy means my life is a black box, except for the items I choose to share with others." By Lauren, age 32

[0] All pictures from the PoPETs'18 paper

Privacy as controls

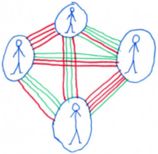


Fig. 46. 'Privacy is a network: I share what I want with whom I want and trust and what matches with those in the network, and don't share with those I don't want and trust to share with. Green = share. Red = don't." age 20s

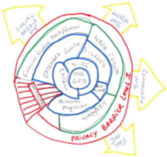


Fig. 47. "I give/receive based on my level of trust. Occasionally, I do not share with those I trust (i.e., my exception jail) as I do not trust what they will do with a specific piece of information. I accept that I must have a public persona." By Jim, age 51

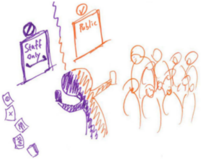


Fig. 55. "There are bright sides, and there are dark sides. Some of them we'd love to share; some we don't, and they are called 'privacy."' By Evan, age 21

[0] All pictures from the PoPETs'18 paper

Privacy as tools

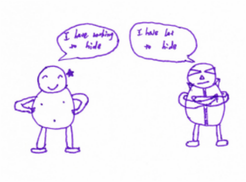


Fig. 54. By Lidong Wei

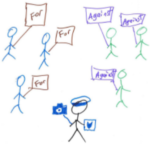


Fig. 30. "People should be able to express their views without surveillance & infiltration by the police." By anonymous, age "old"

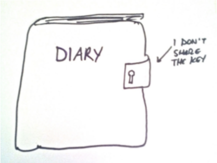


Fig. 28. "The picture is of a diary that has a mechanism to keep it shut. There is no key available within the drawing, so that no one can open it. If no one can open it, privacy remains intact." By Karen, age 43

[0] All pictures from the PoPETs'18 paper

What's the point?

- Several different aspects of privacy that people value.
- Privacy-enhancing technologies cover many of these aspects:
  - user controls and usability,
  - secure communication,
  - resisting censorship,
  - fairness and accountability,
  - ... and many more ...
- We'll only cover PETs that are related to two of these:
  - Anonymity: Privacy as masks, with topics like anti-surveillance, hiding identities, and Tor onion routing.
  - Data minimization: Related to privacy as filters. i.e., achieving a functionality, while minimizing the amount of data collected. The topic covered is Private Information Retrieval (PIR).

Outline

1. What is privacy?
2. Remailers
3. Tor
4. Private information retrieval (PIR)

Categorizing nymity

The goal of being anonymous: hiding identities

- Anonymity set: Set of possible candidates, known beforehand.
- We can place transactions (both online and offline) on a continuum according to the level of *nymity* they represent, that is, how they refine the anonymity set:
  - Verinymity: (Almost) unique information.
    - Government ID, SIN, credit card #, address
  - Persistent pseudonymity: a pseudonym or a “handle” that is used persistently by the same person
    - Posting blogs under a pseudonym, Twitter / Instagram usernames, etc
  - Linkable anonymity:
    - Prepaid phone cards, Loyalty cards
  - Unlinkable anonymity
    - Cash payments, Remailer, Tor (browser)

Nymity design decisions

- If you build a system at a certain level of nymity, it is *easy* to modify it to have a *higher* level of nymity, but *hard* to modify it to have a *lower* level.
- The lesson: design systems with a low level of nymity fundamentally; adding more is easy.

Anonymity for email: remailers

How to send and receive emails without revealing your own email address?

- Anonymous remailers
  - If “From” is hidden, then who do you reply to?

Type 0 remailers

In the 1990s, there were very simple ("type 0") remailing services, the best known being anon.penet.fi (1993–1996)

Here is how it worked:

- Send email to `anon.penet.fi`
- It is forwarded to your intended recipient
- Your "From" address is changed to `anon43567@anon.penet.fi` (but your original address is stored in a table)
- Replies to the anon address get mapped back to your real address and delivered to you

anon.penet.fi

This works, as long as:

- No one's watching the Internet connections to or from `anon.penet.fi`
- The operator of `anon.penet.fi`, the machine (hardware), and the software all remain trustworthy and uncompromised
- The mapping of anon addresses to real addresses is kept secret

Unfortunately, a lawsuit forced Julf (the operator) to turn over parts of the list, and he shut down the whole thing, since he could no longer legally protect it

Type I remailers

Cypherpunk (type I) remailers removed the central point of trust

- Messages are now sent through a "chain" of several remailers, with dozens to choose from
- Each step in the chain is encrypted to avoid observers following the messages through the chain
- Remailers also delay and reorder messages

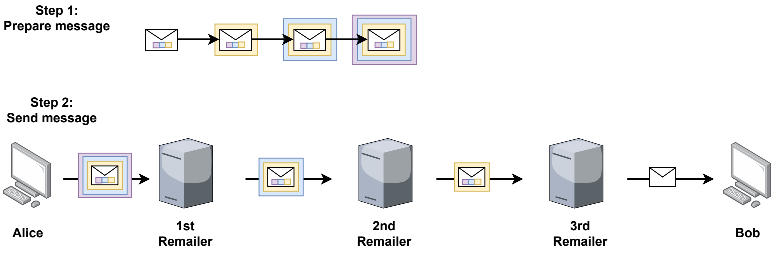
Support for pseudonymity is dropped: no replies!

Nym servers / pseudonymous remailers

How to do replies? (i.e., recovering pseudonymity)

- “nym servers” mapped pseudonyms to “reply blocks” that contained a nested encrypted chain of type I remailers.
- User A approaches a nym server with a chain of reply blocks for the nym server to relay back responses
- User A sends an anonymous mail B (via a chain of Type I remailers), including the chain of reply blocks
- User B responds to the nym server by attaching the response to the end of the reply blocks
- nym server relay the response back to user A by following the chain of reply blocks

Nym servers / pseudonymous remailers



Type II remailers

Mixmaster (type II) remailers appeared in the late 1990s

- Constant-length messages to avoid an observer watching “that big file” travel through the network
- Protections against replay attacks
- Improved message reordering

Requires a special email client to construct the message fragments

Type III remailers

Mixminion (type III) remailer appears in the 2000s

- Native (and much improved) support for pseudonymity
  - No longer reliant on type I reply blocks
  - Instead, relies on mix networks
- Improved protection against replay and key compromise attacks

But it's not very well deployed or mature, i.e., "you shouldn't trust Mixminion with your anonymity yet"

Protocol for send and receive

Consider a case that

- A wants to send a message (M) to B
- A expects B to reply
- A wants to remain anonymous to B for the whole process

For simplicity, assuming one mix hop (H) between A and B.

- A forms an untraceable return address $\langle S_1 \parallel A \rangle_{K_H}$
- A choose a one-time public key for B to encrypt the response K_x
- Both pieces and the message are encrypted with B's public key $P = \langle R_0 \parallel M \parallel \langle S_1 \parallel A \rangle_{K_H} \parallel K_x \rangle_{K_B}$
- A sends $\langle R_1 \parallel P \parallel B \rangle_{K_H}$ to hop which extracts P send it to B
- B sends response to hop: $\langle \langle S_0 \parallel X \rangle_{K_x} \parallel \langle S_1 \parallel A \rangle_{K_H} \rangle_{K_H}$
- Hop decrypts the response and extracts S_1 and A.
- Hop maintains a mapping of $S_1 \rightarrow A$ so it knows that the response needs to be relayed back to A.

The mix operation

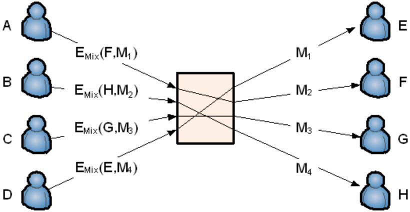


Figure: A white-box view

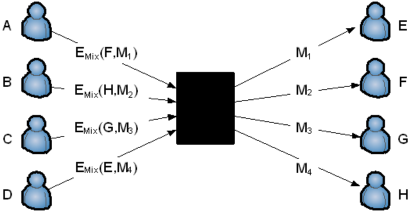


Figure: A black-box view

Outline

1. What is privacy?
2. Remailers
3. Tor
4. Private information retrieval (PIR)

Tor - purpose

Tor is a successful privacy enhancing technology that works at the transport layer with ≈2 million daily users

Why do we need Tor when we have TLS?

- TLS protects data.
- We also want to protect metadata about the communication: e.g., IP addresses, browser fingerprints.

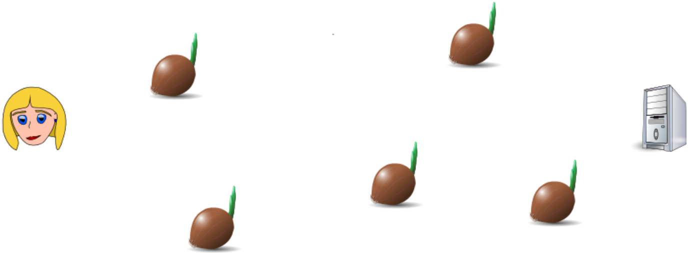
Tor is an anonymity network of nodes

- Scattered around the Internet are about 7,000 Tor nodes, also called Onion Routers

Tor makes internet browsing unlinkably anonymous. But Tor does not (and cannot) hide the existence of the transaction (website visit) altogether

How Tor works

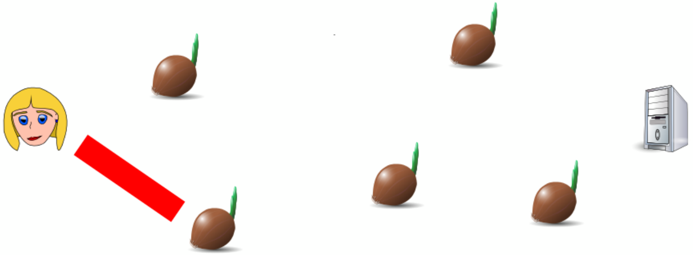
Alice wants to connect to a server without revealing her IP address



Alice has a global view of available Onion Routers

How Tor works

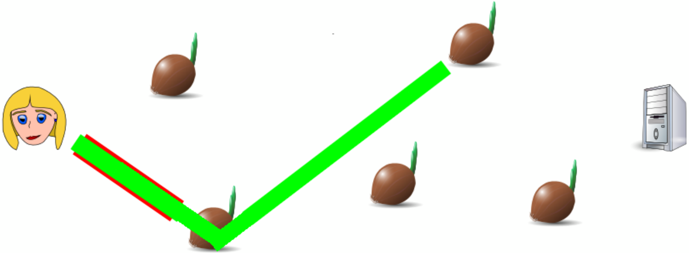
Alice picks one of the Tor nodes (n1) and uses public-key cryptography to establish an encrypted communication channel to it (much like TLS)



Result is a secret key K_1 shared by Alice and n1

How Tor works

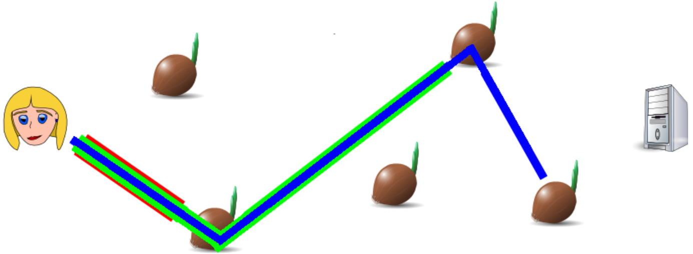
Alice tells n1 to contact a second node (n2), and establishes a new encrypted communication channel to n2, tunnelled within the previous one to n1



Result is a secret key K_2 shared between Alice and n2, which is unknown to n1

How Tor works

Alice tells n2 to contact a third node (n3), and establishes a new encrypted communication channel to n3, tunnelled within the previous one to n2

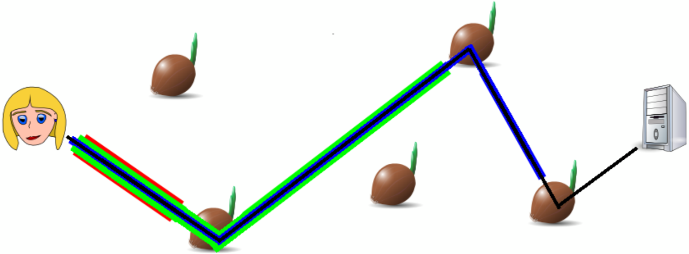


Result is a secret key K_3 shared between Alice and n3, which is unknown to n1 and n2

How Tor works

... And so on, for as many steps as she likes (usually 3) ...

Alice tells the last node (within the layers of tunnels) to connect to the website



Sending messages with Tor

- Alice now shares three secret keys:
  - $K1$ with n1
  - $K2$ with n2
  - $K3$ with n3
- When Alice wants to send a message M, she actually sends $E_{K1}(E_{K2}(E_{K3}(M)))$
- Node n1 uses $K1$ to decrypt the outer layer, and passes the result $E_{K2}(E_{K3}(M))$ to n2
- Node n2 uses $K2$ to decrypt the next layer, and passes the result $E_{K3}(M)$ to n3
- Node n3 uses $K3$ to decrypt the final layer, and passes the result M to the server

Replies in Tor

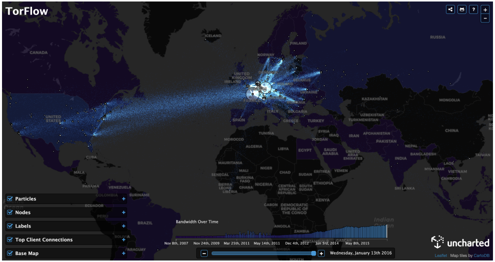
- When the website replies with message R, it will send it to n3
  - Why?
- Node n3 will encrypt R with $K3$ and send $E_{K3}(R)$ to n2
- Node n2 will encrypt that with $K2$ and send $E_{K2}(E_{K3}(R))$ to n1
- Node n1 will encrypt that with $K1$ and send $E_{K1}(E_{K2}(E_{K3}(R)))$ to Alice
- Alice will use $K1$, $K2$, and $K3$ to decrypt the layers of the reply and recover R

Who knows what?

- Notice that node n1 knows that Alice is using Tor, and that her next node is n2, but does not know which website Alice is visiting
- Node n3 knows some Tor user (with previous node n2) is visiting a particular website, but doesn't know who
- The website itself only knows that it got a connection from Tor node n3

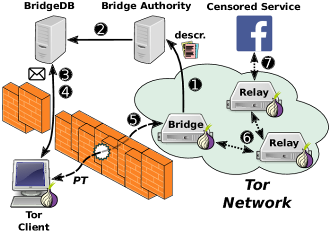
Global adversary and path selection

- What happens if an adversary can inspect all network links?
  - i.e., a global passive adversary
- How concentrated is the geographical distribution of Tor relays?
  - Path selection may help increase performance and anonymity



Tor and Internet censorship

- State-level adversaries can restrict connections to public Tor relays or otherwise attempt to fingerprint Tor traffic on the network
- Solution?
  - Distribute addresses of non-public Tor relays (bridges)
  - Modify Tor traffic to look like something else (pluggable transports)



[0] Picture from Matic et. al, NDSS'17 paper

Anonymity vs. pseudonymity

Tor provides for anonymity in TCP connections over the Internet, both unlinkably (long-term) and linkably (short-term)

What does this mean?

- There's no long-term identifier for a Tor user
- If a web server gets a connection from Tor today, and another one tomorrow, it won't be able to tell whether those are from the same person
- But two connections in quick succession from the same Tor node are more likely to come from the same person

Outline

1. What is privacy?
2. Remailers
3. Tor
4. Private information retrieval (PIR)

Motivation

Simple scenario:

- Netflix stores its' movies in a database
  1. The Shawshank Redemption
  2. The Godfather
  3. The Dark Knight
  4. 12 Angry Men
  5. ...
- You request movies by index, say 1, 4, 2, ...
- Netflix caches your selection and gradually builds a profile on your movie preferences
- But why? You has bought a Netflix license and so you should be able to access different movies

Definition

Goal: allow a user to query a database while hiding the identity of the data-items the user is after

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Non-private protocol

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Protocol:

- User: show me i
- Server: here is X_i

Analysis:

- No privacy!
- # of bits: 1 — very efficient

Trivially-private protocol

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Protocol:

- User: show me *ALL*
- Server: here is $\{X_1, X_2, ..., X_n\}$

Analysis:

- Complete privacy!
- # of bits: n — impractical

Sad news: if the server has unlimited computational power AND there is only a single copy of the database,
$\Longrightarrow$ n bits must be transferred!

“More” solutions?

- User asks for additional random indices
  - Drawback: balance information leak vs communication cost
- Anonymity
  - Note: this is in fact a different concern: it hides the identity of a user, not the fact that X_i is retrieved

Information-theoretic PIR (IT-PIR)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

An example 2-server IT-PIR protocol:

- User → Server 1: $Q_1 \subset_R \{1, 2, ..., n\} \wedge i \notin Q_1$
- Server 1 → User: $R_1 = \bigoplus_{k \in Q_1} X_k$
- User → Server 2: $Q_2 = Q_1 \cup \{i\}$
- Server 2 → User: $R_2 = \bigoplus_{k \in Q_2} X_k$
- User derive $X_i = R_1 \oplus R_2$

Analysis:

- Probabilistic-based privacy $(1/|Q_2|)$
- # of bits: 1 (× 2 servers) + inexpensive computation

Computational PIR

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: A single server with limited computational power

An example CPIR protocol:

- User chooses a large random number m
- User generates $n-1$ random quadratic residue (QR) mod m: $a_1, a_2, ..., a_{i-1}, a_{i+1}, ..., a_n$
- User generates a quadratic non-residue (QNR) mod m: b_i
- User → Server: $a_1, a_2, ..., a_{i-1}, b_i, a_{i+1}, ..., a_n$
- Server cannot distinguish between QRs and QNRs mod m, i.e., the request is just a series of random numbers: $u_1, u_2, ..., u_n$
- Server → User: $R = u_1^{X_1} \cdot u_2^{X_2} \cdot ... \cdot u_n^{X_n}$
- If R is a QR mod m, $X_i = 0$, else (R is a QNR mod m) $X_i = 1$

Comparison of CPIR and IT-PIR

CPIR

- Possible with a single server
- Server needs to perform intensive computations
- To break it, the server needs to solve a hard problem

IT-PIR

- Only possible with > 1 server.
- Server may need lightweight computations only
- To break it, the server needs to collude with other servers

CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 5 - Encrypted Traffic Analysis

Spring 2022

Outline

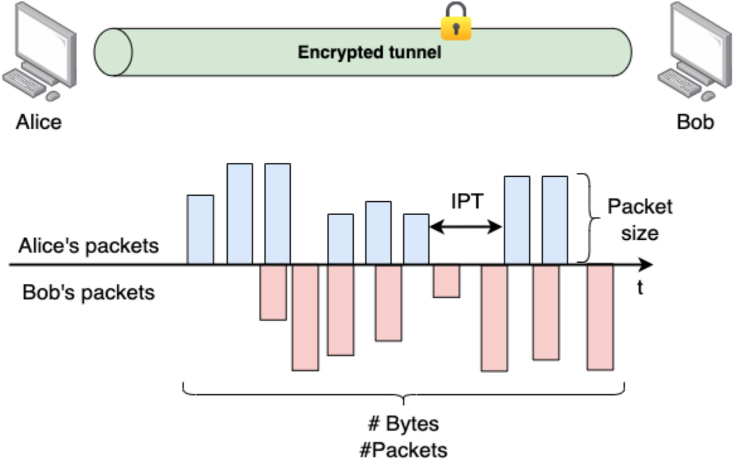
1. Overview
2. Network Analytics
3. Network Security
4. Privacy Breaches
5. Countermeasures

Encryption reduces visibility over network traffic

- TLS and other PETs significantly improved security and privacy for Internet users
  - Plaintext is no longer visible
  - Traffic monitoring capabilities are significantly reduced
- But one should not assume that traffic encryption provides absolute protection
  - e.g., against behavioural analysis
- There are strong incentives to “see” beyond encryption
  - Both for network adversaries and network administrators

Encrypted traffic analysis (ETA)

Let's take a look at an encrypted tunnel between Alice and Bob:



Network flows and metadata

What is a network flow?

- A flow is typically represented by a five-tuple
  - <Src. IP, Dest. IP, Src. port, Dest. port, Proto>
- One can extract additional metadata tied to a flow, based on:
  - Flow duration
  - Amount of packets exchanged
  - Packet sizes
  - Packet inter-arrival times
  - Payload byte entropy
  - And more...

What is this good for?

Encrypted traffic analysis (ETA) as a side channel

- Do you remember side channels from module 2?
  - Think of ETA as a sort of network side channel!
  - ETA can be used to infer sensitive information about encrypted traffic flows
- We'll look at three particular ETA applications for:
  - network analytics, network security, and privacy breaches
  - and also discuss potential countermeasures

Outline

1. Overview
2. Network Analytics
3. Network Security
4. Privacy Breaches
5. Countermeasures

Network Analytics

Traffic Engineering

- Prioritize application traffic (e.g., WhatsApp, Skype)
  - e.g., for non-neutral Internet ISPs
- Throttle selected protocols (e.g. BitTorrent)
  - e.g., for “traffic management” purposes

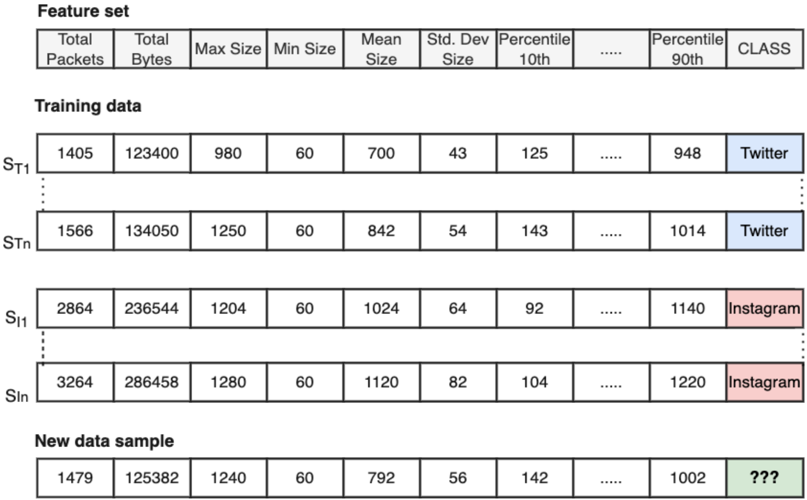
Quality-of-Service

- Derive quality metrics from encrypted flows
  - e.g. videoconferencing and video streaming QoE
  - e.g. websites’ page load time, speed index

Use case: Identification of mobile applications

- Mobile applications' traffic leaves a fingerprint
  - Network observers can understand which apps you are using
- Build a classifier based on summary statistics from each flow
  - Look at the packet size/timing distributions
  - Minimum, maximum, mean, standard deviation, variance, skew, kurtosis, percentiles, etc.
- May need to separate traffic bursts
  - Network packets occurring together within a threshold of time
  - Traffic bursts may encompass multiple flows

Let’s classify some apps!



Feature set

Total Packets	Total Bytes	Max Size	Min Size	Mean Size	Std. Dev Size	Percentile 10th	.....	Percentile 90th	CLASS

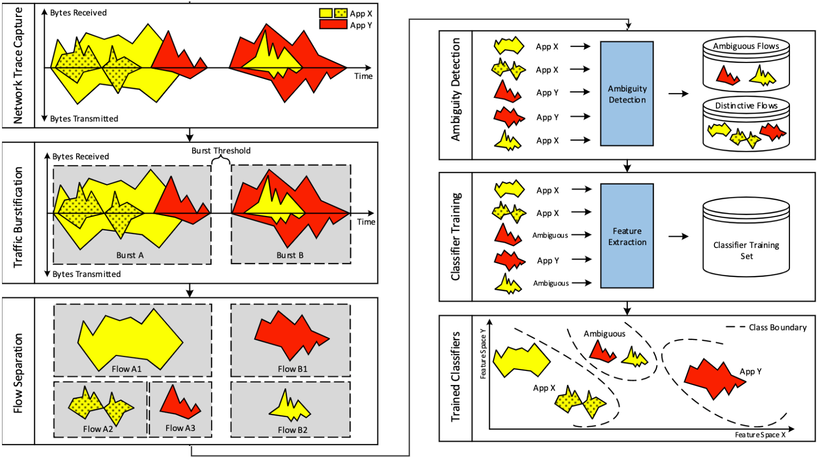
Training data

S_{T1}	1405	123400	980	60	700	43	125	.....	948	Twitter
S_{Tn}	1566	134050	1250	60	842	54	143	.....	1014	Twitter
S_{I1}	2864	236544	1204	60	1024	64	92	.....	1140	Instagram
S_{In}	3264	286458	1280	60	1120	82	104	.....	1220	Instagram

New data sample

1479	125382	1240	60	792	56	142	.....	1002	???

Use case: Identification of mobile applications



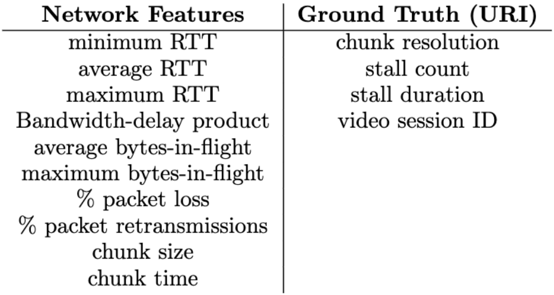
Taylor et al., IEEE TIFS '17

Use case: Measuring video QoE from encrypted traffic

- Majority of video traffic is delivered over adaptive bitrate
  - A video is encoded in multiple resolutions and split into chunks of variable length
  - Clients continuously fill a buffer of chunks, where ensuing chunks are based on network conditions
- DPI solutions can no longer be used to extract meaningful QoE metrics
  - e.g., initial delays, playback stalls frequency, resolution switch

Use case: Measuring video QoE from encrypted traffic

- Features extracted from encrypted traffic guide the models to detect quality impairments
- Able to detect stalls, average quality, and video quality adjustments



Network Features	Ground Truth (URI)
minimum RTT	chunk resolution
average RTT	stall count
maximum RTT	stall duration
Bandwidth-delay product	video session ID
average bytes-in-flight	
maximum bytes-in-flight	
% packet loss	
% packet retransmissions	
chunk size	
chunk time	

Dimopoulos et al., IMC ’16

Outline

1. Overview
2. Network Analytics
3. Network Security
4. Privacy Breaches
5. Countermeasures

Malware Detection

- Traditional network-based malware detection relies on unencrypted data
  - Heavy use of deep packet inspection
  - e.g., for signature-based detection over packet payloads
- No longer useful to detect virus spreading or data exfiltration
- Encrypted traffic analysis helps us to identify:
  - Malware communications towards C&C servers
  - Unusual network traffic patterns in the network

Any idea how?

Malware Detection

Malware classification:

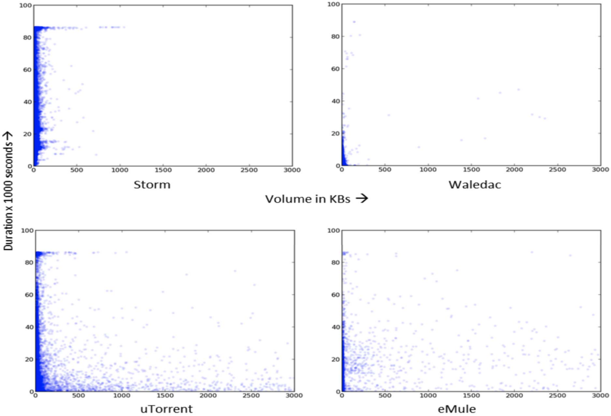
- Build a model out of legitimate / malicious network activity
- Leverage "fingerprints" of legitimate / malicious behaviour
- What if a new malware stream emerges?

Anomaly detection:

- Build a model for legitimate traffic and flag strange behavior
- Via one-class learning or clustering
- What if legitimate behavior changes over time?

Use case: P2P botnet detection

- Can we pinpoint interactions between bots and C&Cs?
    - Tend to be low-volume and long-standing vs. benign P2P apps



Narang et al., IEEE SPW '14

Use case: P2P botnet detection

Flows

- P2P applications (including botnets) randomize port numbers
- The usual flow definition leads to the generation of multiple flows out of what can be a continued interaction between two peers

Super-flows

- Aggregate multiple flows between two IPs into a super-flow
  - What if two IPs have benign and malicious flows between them?

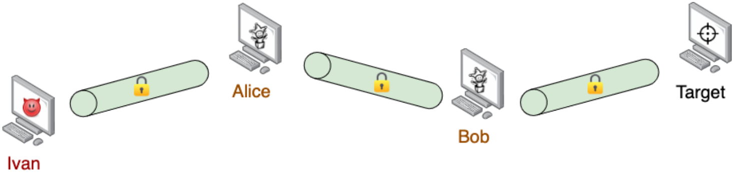
Use case: P2P botnet detection

Conversations

- Start by clustering flows:
  - Protocol, packets per second, avg. payload size
- Create conversations from flows placed within the same clusters
- Finally, classify conversations as malicious or benign based on:
  - Duration of the conversation
  - Number of packets exchanged
  - Volume of data exchanged
  - Median of packet inter-arrival times

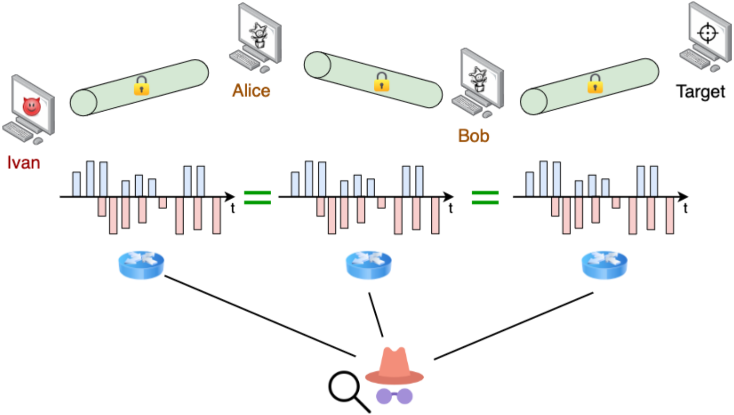
This approach was also shown effective for detecting previously unseen botnets!

Stepping-stones



- An attacker can hide its identity by using other machines as intermediaries (i.e., stepping-stones)
  - e.g., by hopping through compromised machines or by using Tor

Traffic correlation



Detection of stepping-stones

- Attempt to match (roughly) the same sequence of packets at different network vantage points

Difficulties in performing traffic correlation

In practice, flow observations will not be an exact match

- Due to network imperfections
  - Packet delays, jitter, loss
- Due to countermeasures
  - Chaff and delay injection at intermediate nodes, padding

- Traffic correlation algorithms must account for small differences between each flow observation

$$\delta_t(C,C') = \log\left(\prod_{k=1}^{K} |T_k(C',t) - T_k(C,t)|\right)$$

Staniford-Chen and Heberlein, IEEE S&P '95

Outline

1. Overview
2. Network Analytics
3. Network Security
4. Privacy Breaches
5. Countermeasures

Nefarious uses of encrypted traffic analysis

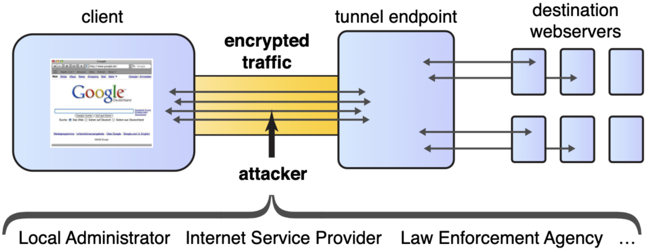
- One would assume that encryption is all that is needed to securely communicate over the Internet
- Unfortunately, encryption does not hide traffic patterns
- Traffic analysis can be weaponized to breach users' privacy

Metadata is not your data. Or is it?

(Dr. Evil dismissing the value of communication metadata)

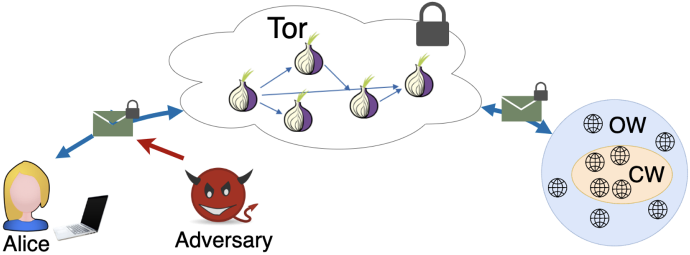
Website fingerprinting over VPNs

- VPNs are often advertised as the “holy-grail” of Internet security
- Passive adversaries can uncover which website is being visited
  - By building traffic fingerprints and using a classifier
- The attack can be launched in two settings:
  - Closed-world
  - Open-world



Herrmann et al., CCSW ’09

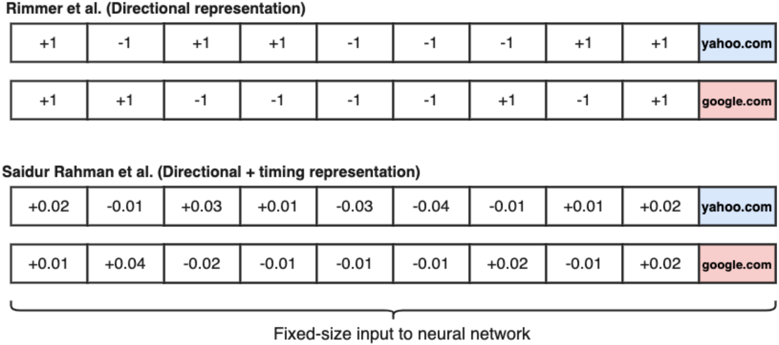
Website fingerprinting over Tor



- The Tor network can be seen as one “big VPN node”
  - Tor exchanges data in fixed-size cells
  - But packet direction and timing still leaks information

Website fingerprinting over Tor

- Lately, learned features based on different traffic representations have been used to launch website fingerprinting attacks on Tor
  - Directional representation Rimmer et al., NDSS ’18
  - Directional + timing representation Saidur Rahman et al., PoPETs ’20



Rimmer et al. (Directional representation)

+1	-1	+1	+1	-1	-1	-1	+1	+1	yahoo.com
+1	+1	-1	-1	-1	-1	+1	-1	+1	google.com

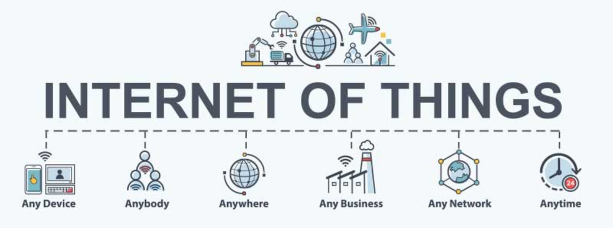
Saidur Rahman et al. (Directional + timing representation)

+0.02	-0.01	+0.03	+0.01	-0.03	-0.04	-0.01	+0.01	+0.02	yahoo.com
+0.01	+0.04	-0.02	-0.01	-0.01	-0.01	+0.02	-0.01	+0.02	google.com

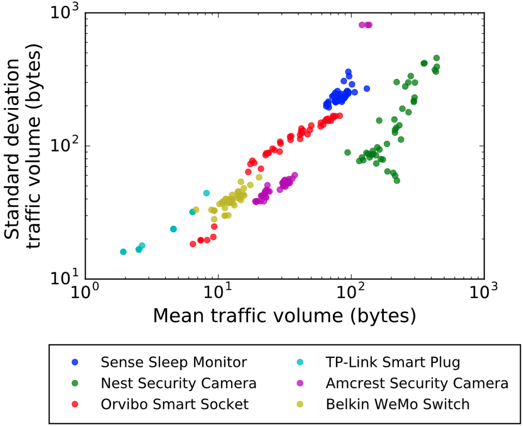
Fixed-size input to neural network

IoT device fingerprinting

- Passive network observers can potentially analyze IoT network traffic to infer sensitive details about users
  - Does this user have a blood monitor? A security camera? A sex toy?
- DNS queries associated with each encrypted flow often contain the device manufacturer name
  - We can still pinpoint the device without this information



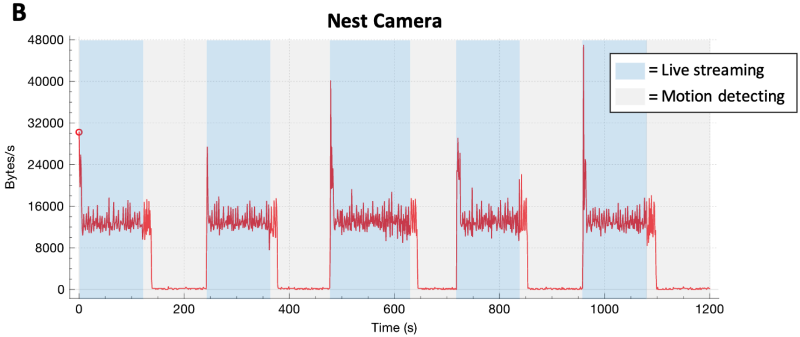
Distinguishing devices through traffic volume



Apthorpe et al., ConPro '17

- Rather simple volumetric features allow us to identify IoT devices
- Once a device is identified, one can also infer device state

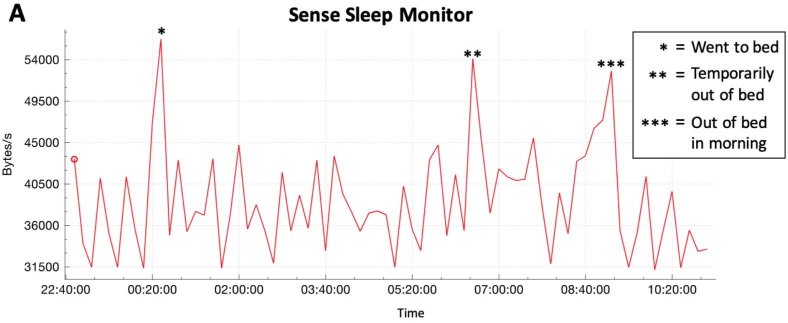
Motion sensor example - Nest indor security camera



Apthorpe et al., DAT '16

- Easy to discern when the camera picks up movement
- Easy to discern when nobody's home?

Sleep tracker example - Sense sleep monitor

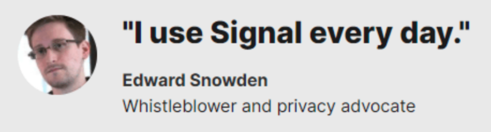


Apthorpe et al., DAT '16

- Easy to discern when a user goes to bed and wakes-up
- Easy to discern if a burglar should leave the crime scene?

Practical attacks against IM applications

- IM applications are extensively used to exchange potentially sensitive content securely
  - Remember OTR and Signal
  - Oftentimes used to exchange politically and socially sensitive content
- Governments and corporations may be interested in identifying participants of IM conversations
  - e.g., target whistleblowers or dissidents



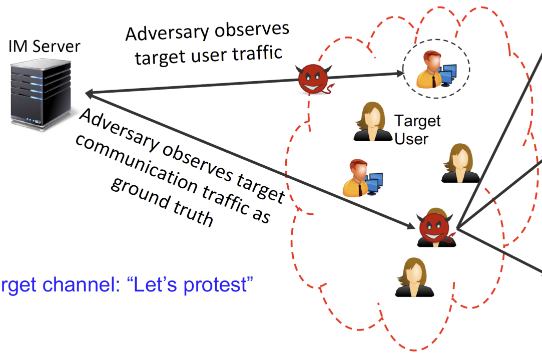
"I use Signal every day."

Edward Snowden

Whistleblower and privacy advocate

Adversary aims to uncover group membership

- How can the adversary set up the attack?


IM Server
Adversary observes target user traffic
Adversary observes target communication traffic as ground truth
Target User

Adversary joins the target channel as a member.

Adversary joins the target channel as an admin.

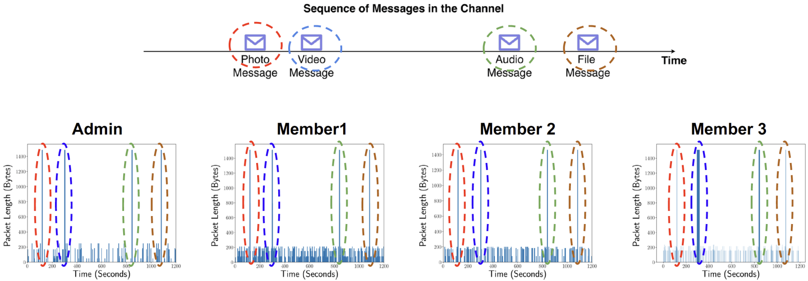
Adversary wiretaps an identified member/admin.

Target channel: “Let’s protest”

Bahramali et al., NDSS ’20

Looking for messaging events

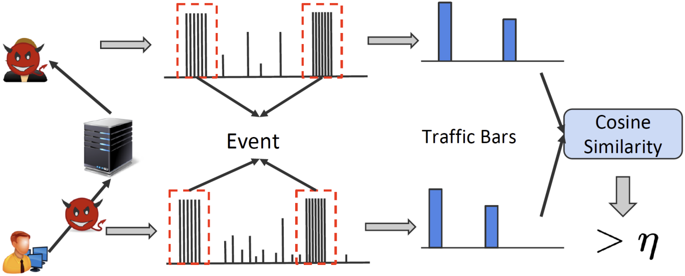
- Messaging events have different fingerprints



Bahramali et al., NDSS '20

Matching messaging events fingerprints

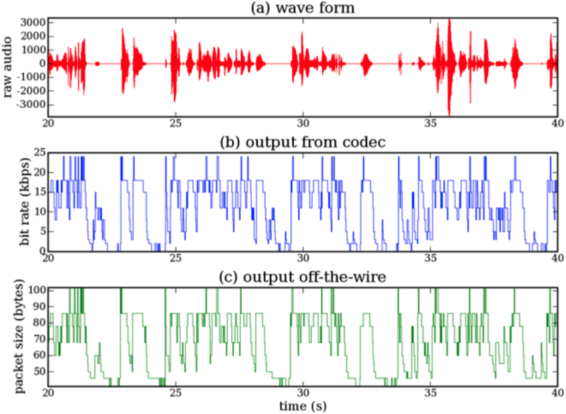
- Extract meaningful events and compare similarity
- Attack succeeded against Signal, Telegram, and WhatsApp!



Bahramali et al., NDSS '20

VoIP eavesdropping

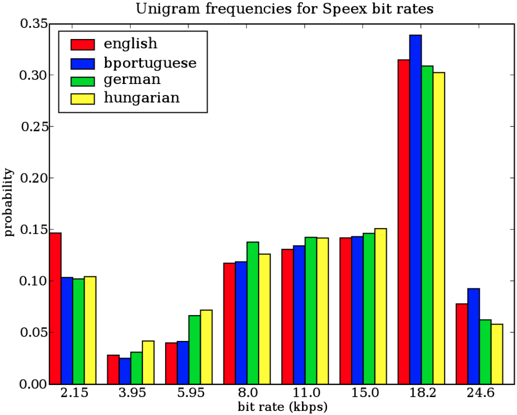
- Encrypted packet patterns resemble VBR codec bitrates
- Can we infer meaningful semantics from the transmission of encrypted audio frames?



Wright et al., USENIX SEC '07

Noticeable (coarse-grained) differences

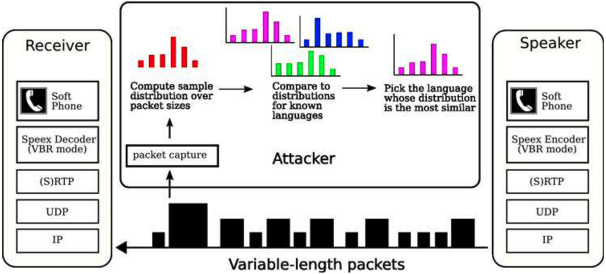
- Maybe we can identify the language being spoken?
- Different languages have different bitrate frequencies



Wright et al., USENIX SEC '07

How to distinguish different languages?

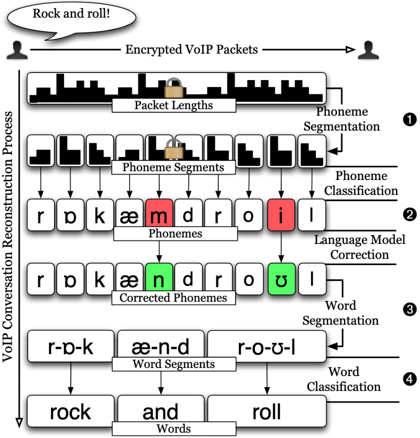
- Compute distance between probability distributions
- Samples from same language have similar distribution
- Compute packet size n-grams for even better results
  - Given sequence 10, 20, 30, 15 –> {(10, 20), (20, 30), (30, 15)}



Wright et al., USENIX SEC '07

Noticeable (fine-grained) differences

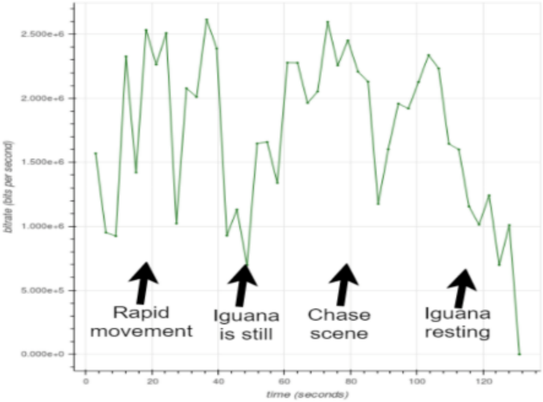
- Can we segment packet size sequences into individual phonems?
- Then we can recover approximated transcripts of a conversation!



White et al., IEEE S&P ’11

Video re-identification

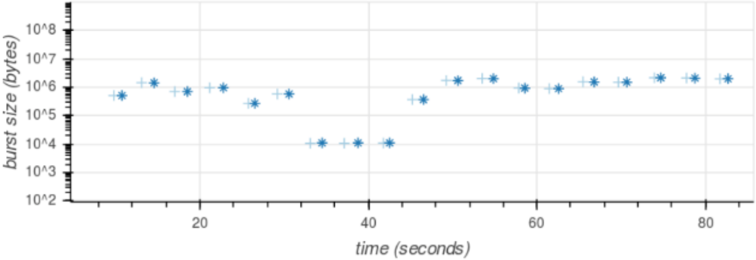
- At this point, you've probably guessed it, traffic analysis can also be used to uncover which videos you are streaming
- The bitrate of VBR video sequences also leaks some information



Schuster et al., USENIX SEC '17

Re-identification of Netflix video streaming

- Burst sizes of a streamed scene of “Reservoir Dogs”
  - Very similar, even when watched over different networks



Schuster et al., USENIX SEC ’17

Outline

1 Overview

2 Network Analytics

3 Network Security

4 Privacy Breaches

5 Countermeasures

Countermeasures to traffic analysis

- Introduce padding
- Add chaff traffic
- Shape traffic (look like something)
- Aggregate traffic (e.g, multiplex IoT traffic in single connection)
- Split a single connection across multiple networks

- Main trade-off to consider is overhead
  - Achievable throughput
  - Spent bandwidth

CS 458 / 658: Computer Security and Privacy

Module 5 - Security and Privacy of Internet Applications

Part 6 - An Introduction to Blockchain Technologies

Spring 2022

Outline

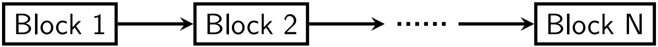
1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

Outline

1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

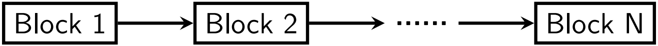
What is a blockchain?

A blockchain is ... a chain of blocks!



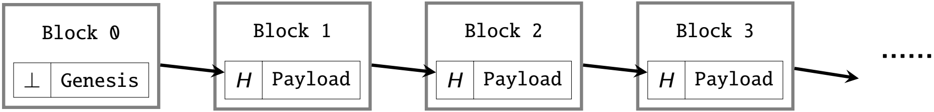
What is a blockchain?

A blockchain is ... a chain of blocks!



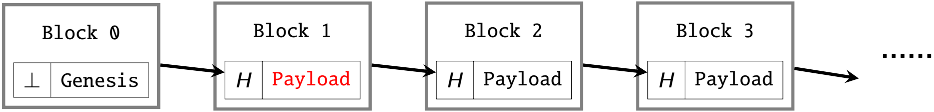
- What does chaining mean here?
  - Linked list? Some cryptographic construct?
- What goes into these blocks?
  - Anything? A fixed format? What makes a block valid?
- Who can put up a block?
  - A single entity? A group of people? Anyone with Internet access?
- How to ensure a same view of the chain?
  - Centralized? Distributed? How to resolve a dispute?

A basic chaining scheme



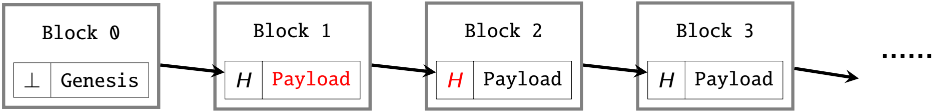
- Each block contains a **cryptographic hash** of the previous block.
- Each block depends on the previous one.

A basic chaining scheme



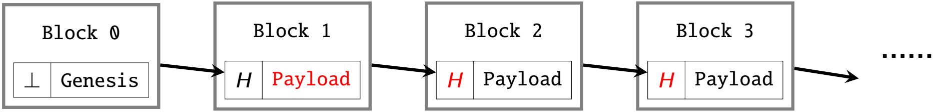
- Each block contains a **cryptographic hash** of the previous block.
- Each block depends on the previous one.

A basic chaining scheme



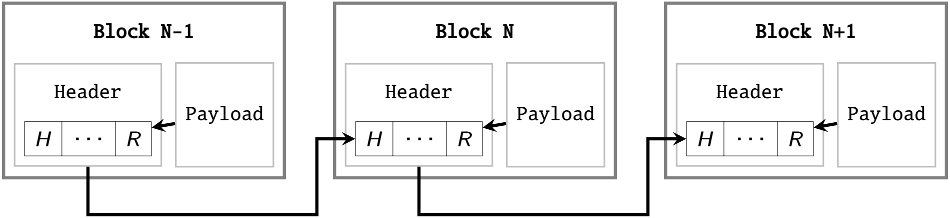
- Each block contains a **cryptographic hash** of the previous block.
- Each block depends on the previous one.

A basic chaining scheme



- Each block contains a **cryptographic hash** of the previous block.
- Each block depends on the previous one.

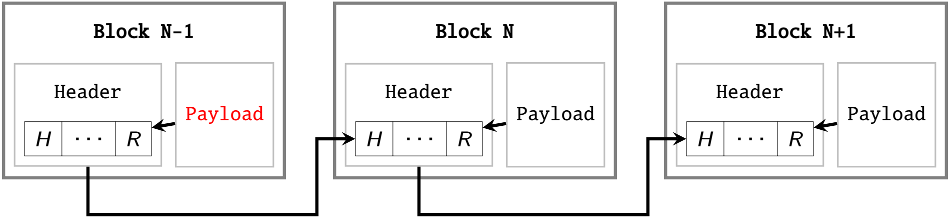
A better chaining scheme



Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

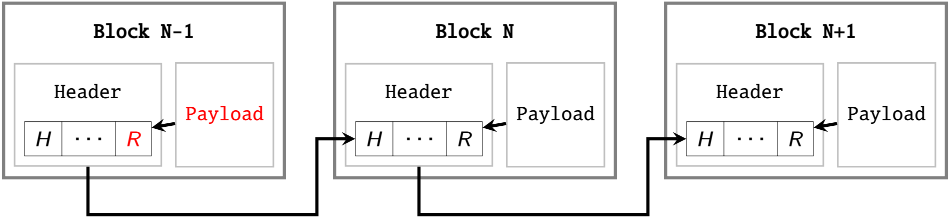
A better chaining scheme



Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

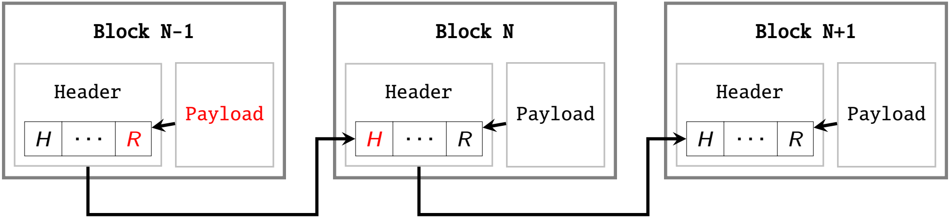
A better chaining scheme



Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

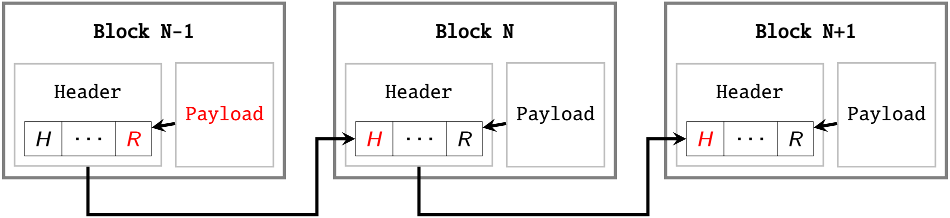
A better chaining scheme



Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

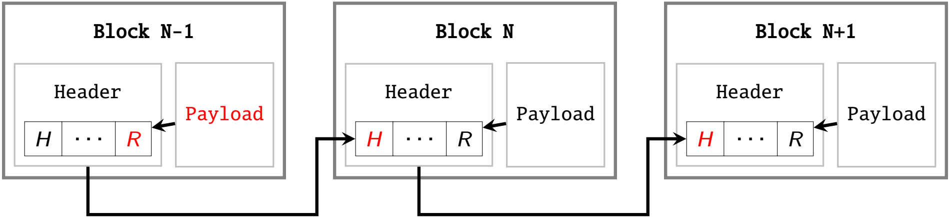
A better chaining scheme



Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

A better chaining scheme

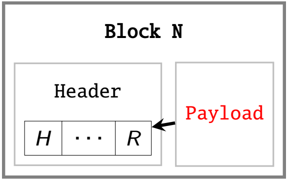


Each block is split into two parts:

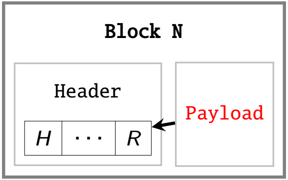
- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

Q: Why this is a better chaining scheme?

What goes into the payload?

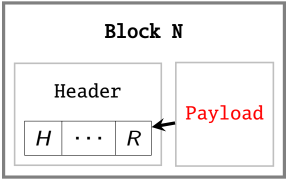


What goes into the payload?



Anything! Depending on how you plan to use this blockchain.

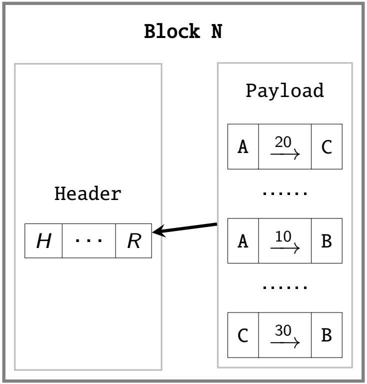
What goes into the payload?



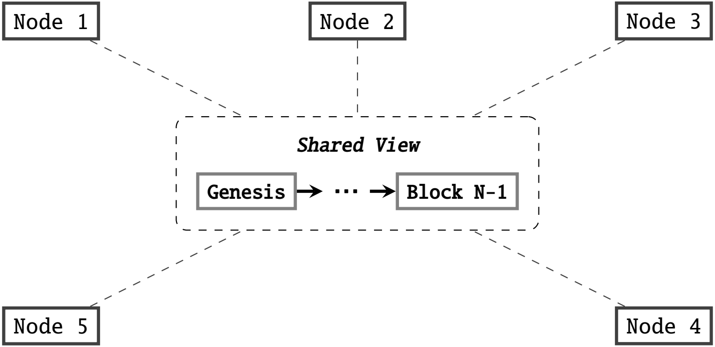
Anything! Depending on how you plan to use this blockchain.

- Bitcoin blockchain: ledger
- Ethereum blockchain: state machine

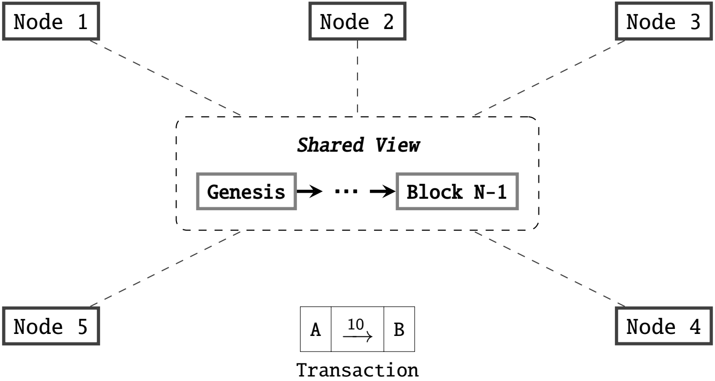
Payload example: a ledger



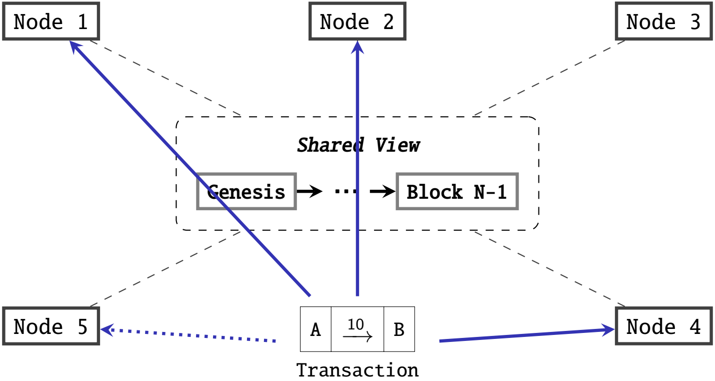
How does the data get into the block?



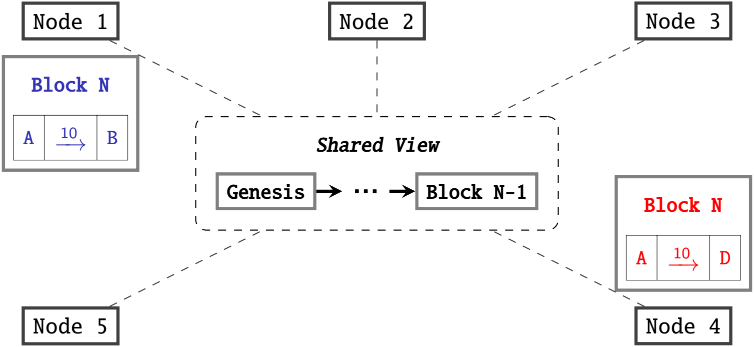
How does the data get into the block?



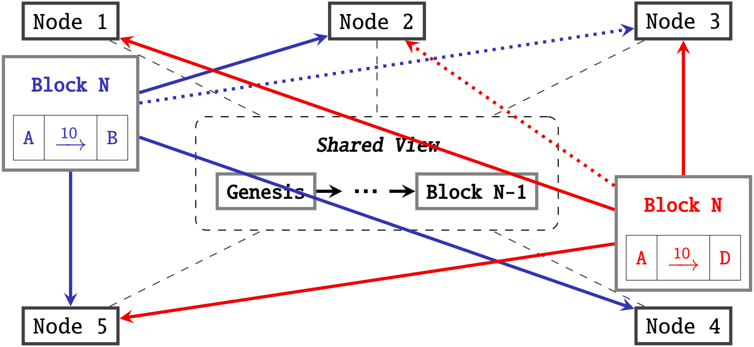
How does the data get into the block?



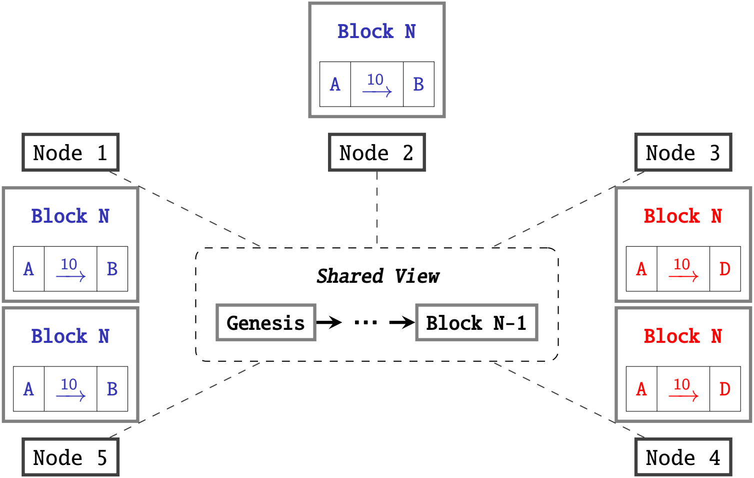
How does the data get into the block?



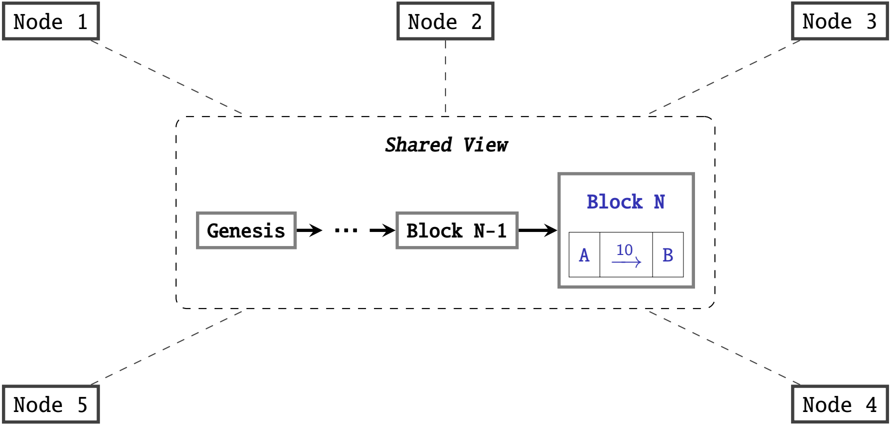
How does the data get into the block?



How does the data get into the block?



How does the data get into the block?



The power of consensus

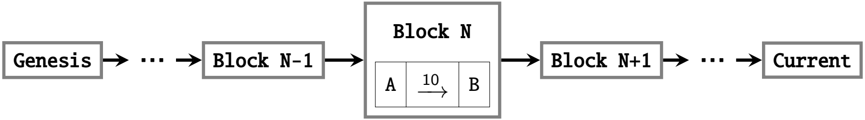
Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

- cash, debit card, credit card, e-transfer (e.g., Interac®)

The power of consensus

Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

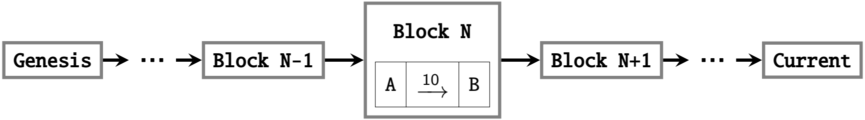
- cash, debit card, credit card, e-transfer (e.g., Interac®)
- an entry in the blockchain-based ledger



The power of consensus

Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

- cash, debit card, credit card, e-transfer (e.g., Interac®)
- an entry in the blockchain-based ledger



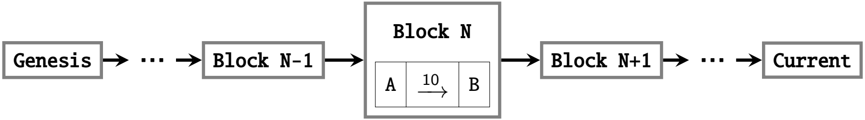
To the best of Bob's knowledge:

- It is hard for Alice to produce such a chain of blocks

The power of consensus

Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

- cash, debit card, credit card, e-transfer (e.g., Interac®)
- an entry in the blockchain-based ledger



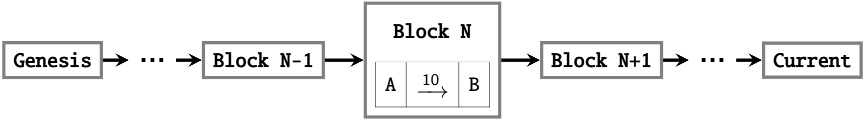
To the best of Bob's knowledge:

- It is hard for Alice to produce such a chain of blocks
- There does not exist a better chain of blocks as of now

The power of consensus

Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

- cash, debit card, credit card, e-transfer (e.g., Interac®)
- an entry in the blockchain-based ledger



Genesis → ··· → Block N-1 → Block N → Block N+1 → ··· → Current

Block N: A $\xrightarrow{10}$ B

To the best of ~~Bob~~ everyone's knowledge:

- It is hard for Alice to produce such a chain of blocks
- There does not exist a better chain of blocks as of now

Summary

Pay attention to two aspects when you design/analyze a blockchain:

- What goes into a block?
- How to ensure consensus?

Summary

Pay attention to two aspects when you design/analyze a blockchain:

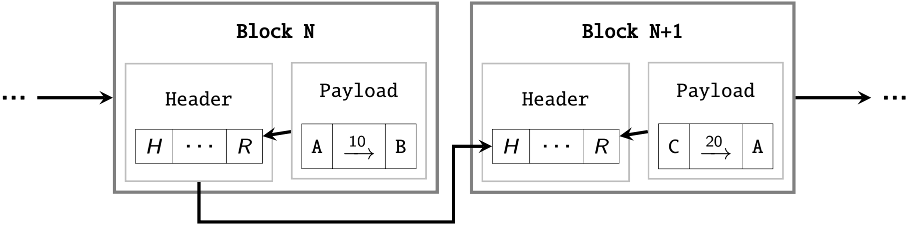
- What goes into a block?
- How to ensure consensus?

In most blockchain systems, these two aspects are orthogonal.

Outline

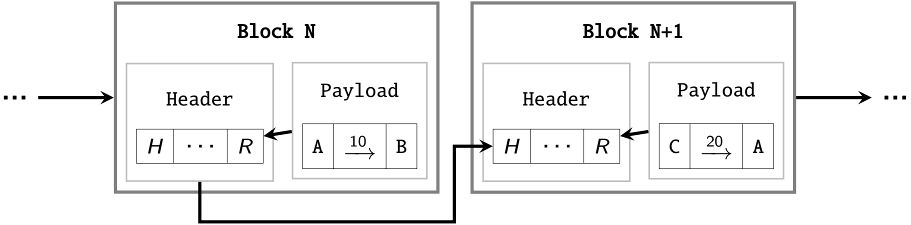
1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

How hard it is to alter this chain?

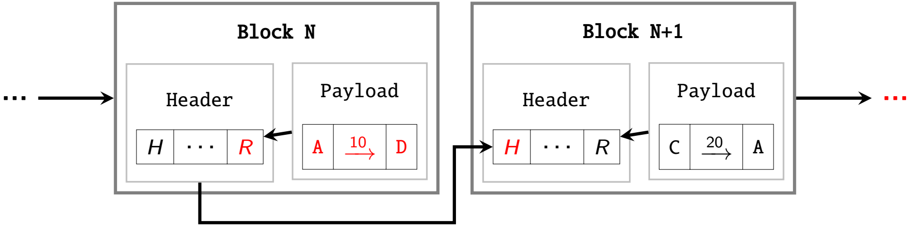


This is the chain Alice shows Bob w.r.t her payment to Bob.

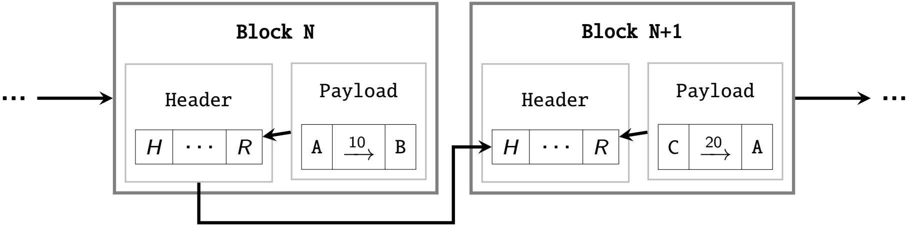
How hard it is to alter this chain?



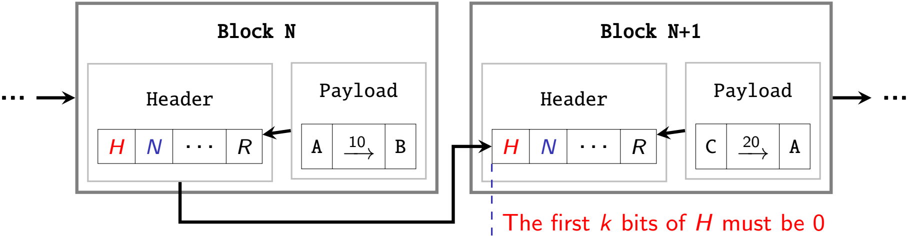
It is not hard at all for Alice to revert the payment to Bob!



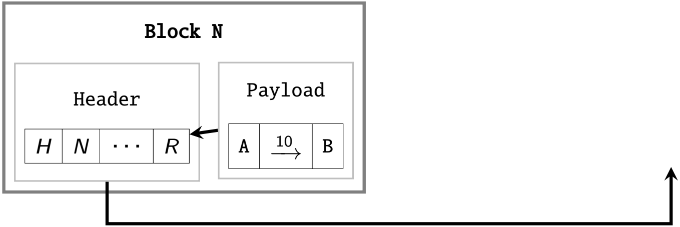
Increase the difficulty



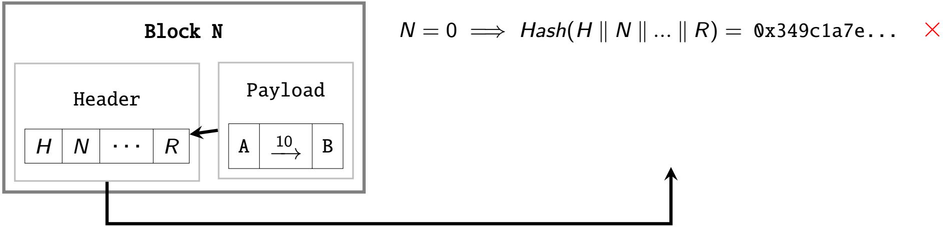
Bob decides to make it harder for Alice to alter her payment



Mining for a valid hash

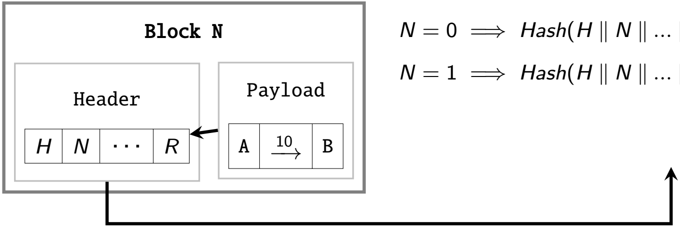


Mining for a valid hash



$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) =$ `0x349c1a7e...` ×

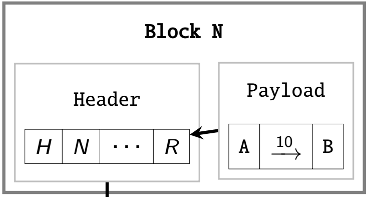
Mining for a valid hash



$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...}$ ✗

$N = 1 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...}$ ✗

Mining for a valid hash

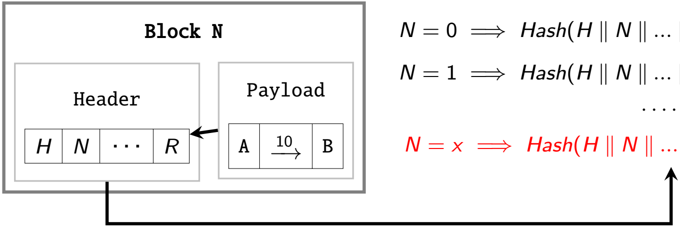


$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...}$ ×

$N = 1 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...}$ ×

......

Mining for a valid hash



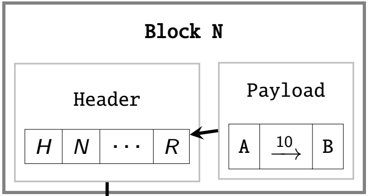
$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...}$ ×

$N = 1 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...}$ ×

$\cdots\cdots$

$N = x \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x00.}_k\texttt{.004f7fed1a}$

Mining for a valid hash



$$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...} \quad \times$$

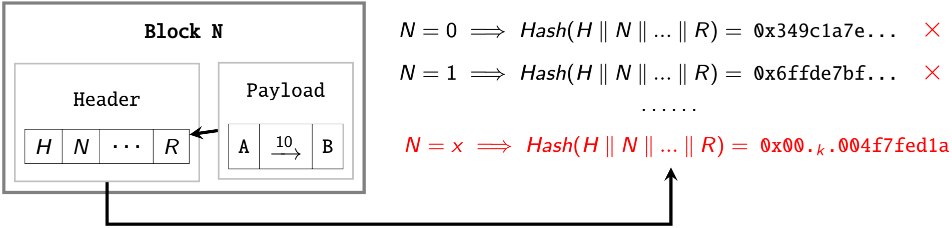
$$N = 1 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...} \quad \times$$

$$\cdots\cdots$$

$$N = x \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x00.}_{k}\texttt{.004f7fed1a}$$

Q: What is the chance of finding a valid N assuming an m-bit hash?

Mining for a valid hash



$N = 0 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...}$ $\times$

$N = 1 \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...}$ $\times$

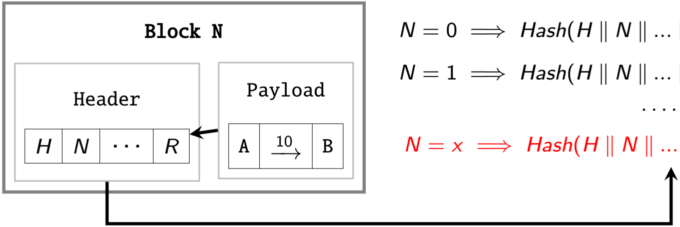
$\cdots\cdots$

$N = x \implies Hash(H \parallel N \parallel \ldots \parallel R) = \texttt{0x00.}_{k}\texttt{.004f7fed1a}$

Q: What is the chance of finding a valid N assuming an m-bit hash?

A: $\frac{2^{m-k}}{2^m}$, a larger $k \implies$ a higher difficulty of finding N

Mining for a valid hash



$N = 0 \implies \mathit{Hash}(H \parallel N \parallel \ldots \parallel R) = \texttt{0x349c1a7e...}$ ×

$N = 1 \implies \mathit{Hash}(H \parallel N \parallel \ldots \parallel R) = \texttt{0x6ffde7bf...}$ ×

$\ldots\ldots$

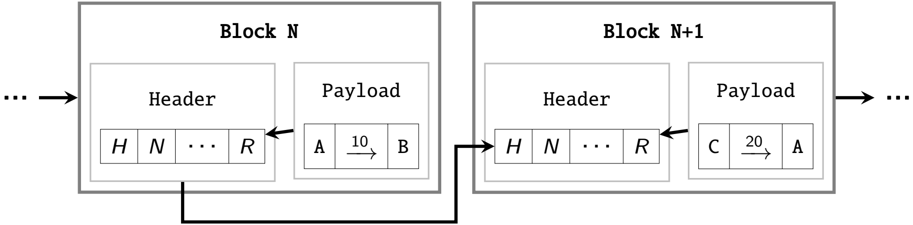
$N = x \implies \mathit{Hash}(H \parallel N \parallel \ldots \parallel R) = \texttt{0x00.}_{k}\texttt{.004f7fed1a}$

Q: What is the chance of finding a valid N assuming an m-bit hash?

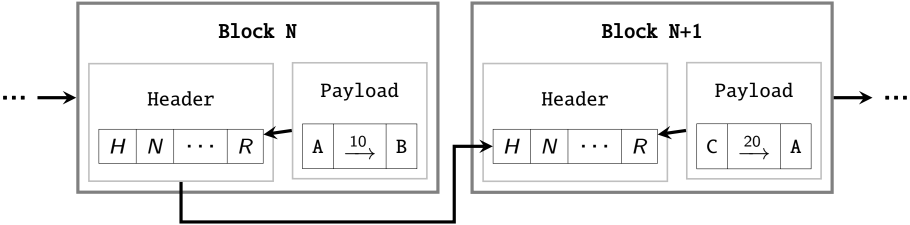
A: $\frac{2^{m-k}}{2^m}$, a larger $k \implies$ a higher difficulty of finding N

i.e., expect 2^k hash operations to find a valid N on average.

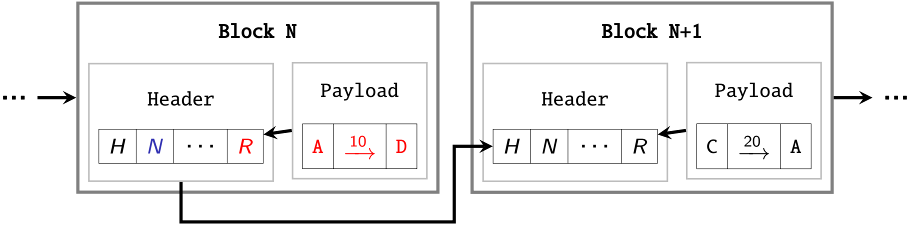
How does mining deter alteration? - Case 1



How does mining deter alteration? - Case 1

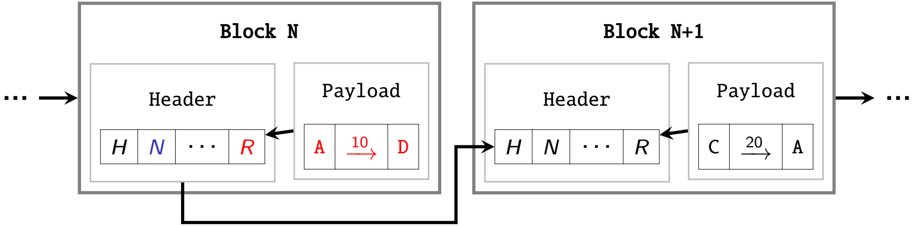


Surgical change: Alice re-mines block N and finds a new nonce such that the block header hash remains unchanged



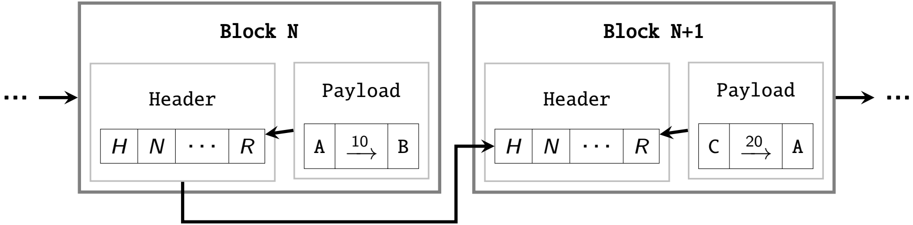
How does mining deter alteration? - Case 1

Surgical change: Alice re-mines block N and finds a new nonce such that the block header hash remains unchanged

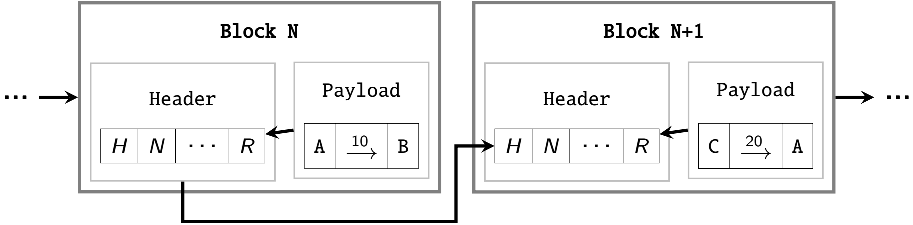


Deterrent: This is extremely hard for a cryptographic hash function that has *preimage resistance* and *second-preimage resistance*.

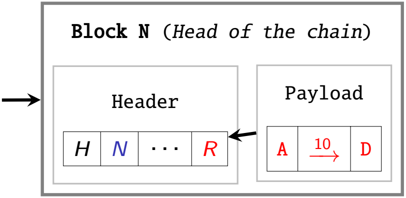
How does mining deter alteration? - Case 2



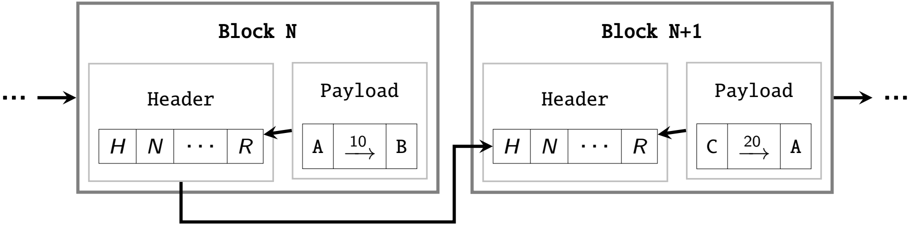
How does mining deter alteration? - Case 2



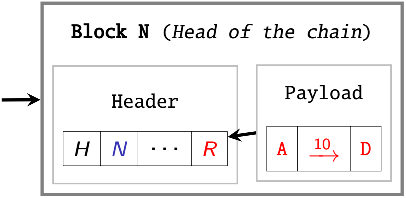
Change-and-cut: Alice re-mines the nonce for block N and stops



How does mining deter alteration? - Case 2

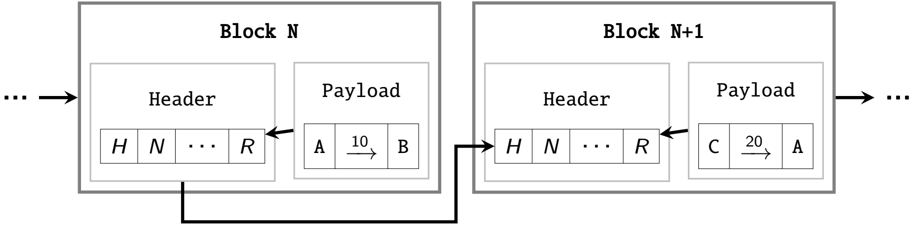


Change-and-cut: Alice re-mines the nonce for block N and stops

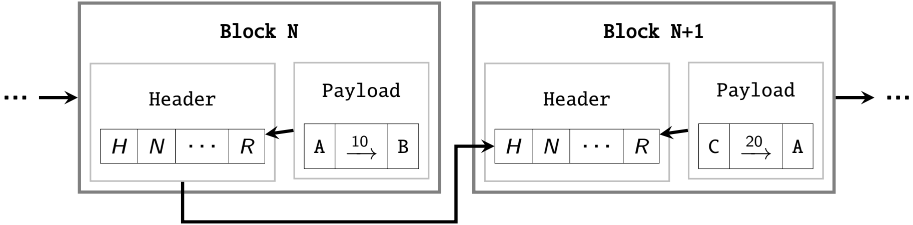


Deterrent: longer chains are preferred over shorter chains.

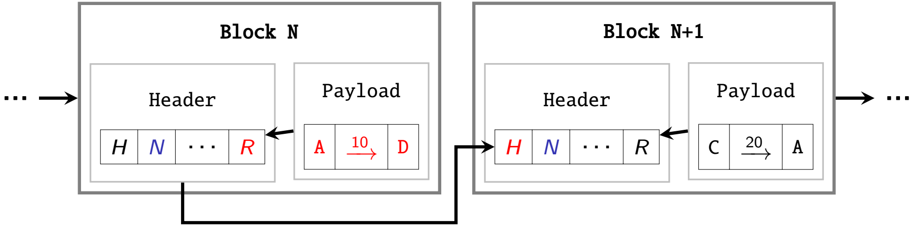
How does mining deter alteration? - Case 3



How does mining deter alteration? - Case 3

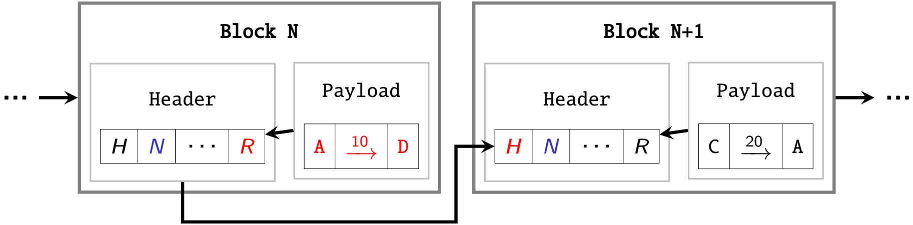


Partial chain re-mining: Alice re-mines all the nonces since block N



How does mining deter alteration? - Case 3

Partial chain re-mining: Alice re-mines all the nonces since block N

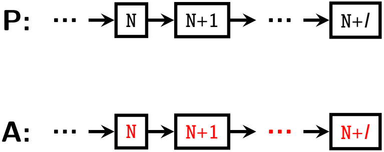


Deterrent: If there are l blocks between and including block N and the chain head, Alice is expected to perform $l \times 2^k$ hash operations to build-up a equally competitive chain assuming the difficulty level k does not change.

51% attack

There is a catch in the deterrent:

Alice mines slower than the rest of the participants combined.



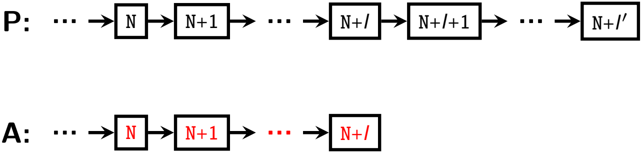
P: ··· → N → N+1 → ··· → N+*l*

A: ··· → N → N+1 → ··· → N+*l*

51% attack

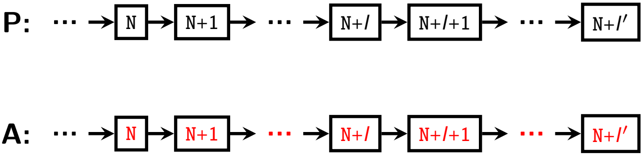
There is a catch in the deterrent:

Alice mines slower than the rest of the participants combined.



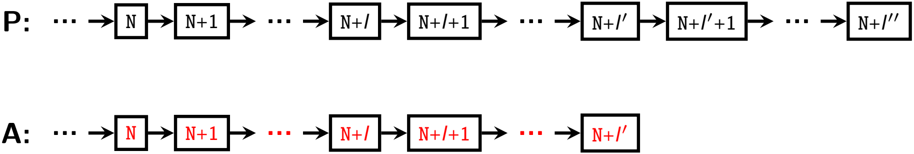
51% attack

There is a catch in the deterrent:
Alice mines slower than the rest of the participants combined.


P:
⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l′
A:
⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l′

51% attack

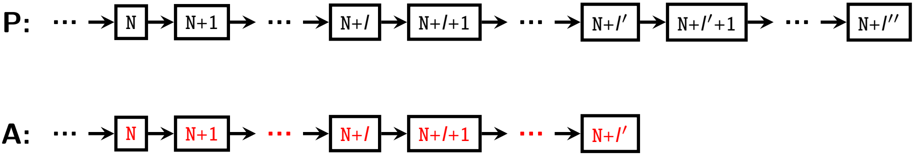
There is a catch in the deterrent:
Alice mines slower than the rest of the participants combined.


P:
⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l' → N+l'+1 → ⋯ → N+l''
A:
⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l'

$\Longrightarrow$ the public chain grows faster than Alice's chain.

51% attack

There is a catch in the deterrent:
Alice mines slower than the rest of the participants combined.



P: $\cdots \to \boxed{\mathtt{N}} \to \boxed{\mathtt{N+1}} \to \cdots \to \boxed{\mathtt{N+}l} \to \boxed{\mathtt{N+}l\mathtt{+1}} \to \cdots \to \boxed{\mathtt{N+}l'} \to \boxed{\mathtt{N+}l'\mathtt{+1}} \to \cdots \to \boxed{\mathtt{N+}l''}$

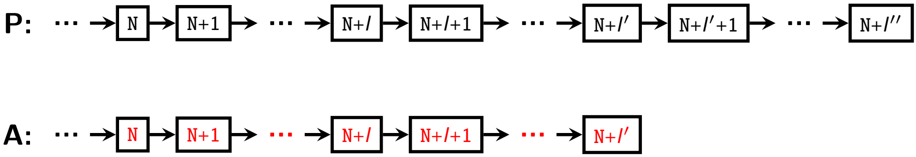
A: $\cdots \to \boxed{\mathtt{N}} \to \boxed{\mathtt{N+1}} \to \cdots \to \boxed{\mathtt{N+}l} \to \boxed{\mathtt{N+}l\mathtt{+1}} \to \cdots \to \boxed{\mathtt{N+}l'}$

$\Longrightarrow$ the public chain grows faster than Alice's chain.

Q: what if Alice mines faster?

51% attack

There is a catch in the deterrent:
Alice mines slower than the rest of the participants combined.



P: ⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l' → N+l'+1 → ⋯ → N+l''

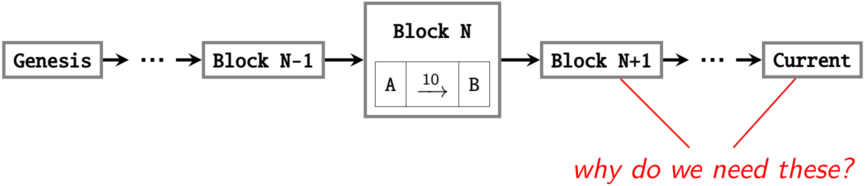
A: ⋯ → N → N+1 → ⋯ → N+l → N+l+1 → ⋯ → N+l'

$\Longrightarrow$ the public chain grows faster than Alice's chain.

Q: what if Alice mines faster?
A: Alice gets to rewrite the history.

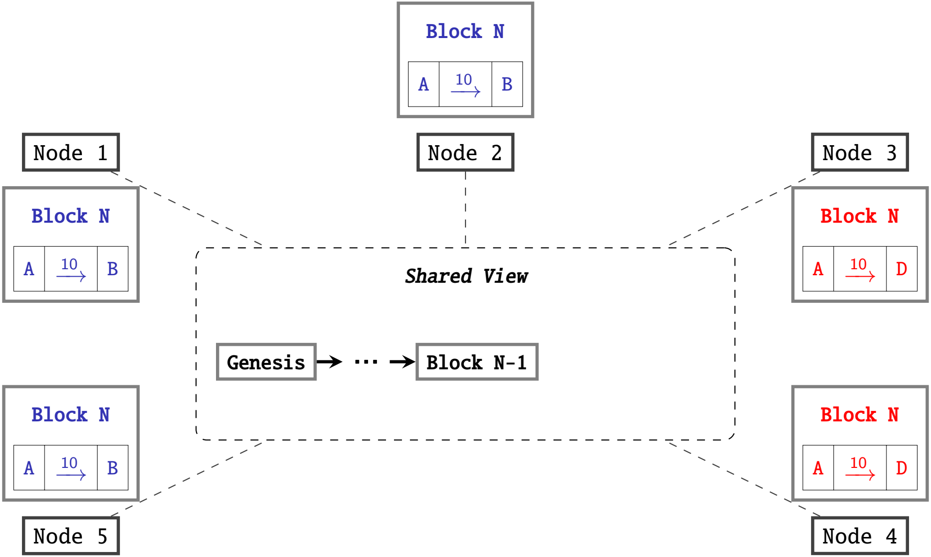
Confirmation level

Recall that when we show a proof of payment, we need a few extra blocks after the block that hosts the ledger entry.

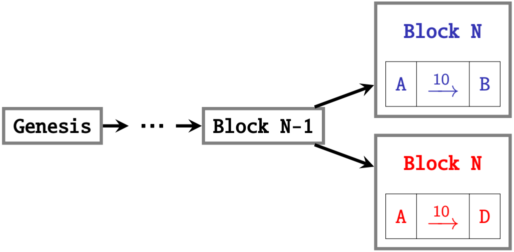


Q: Why do we need these extra blocks even when
1) Alice does not control over 50% of computational power and
2) everyone else is honest and cooperative?

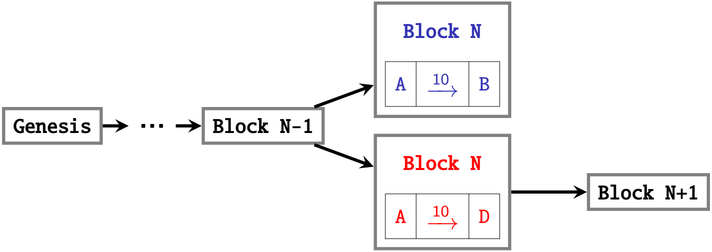
How does the data get into the block?



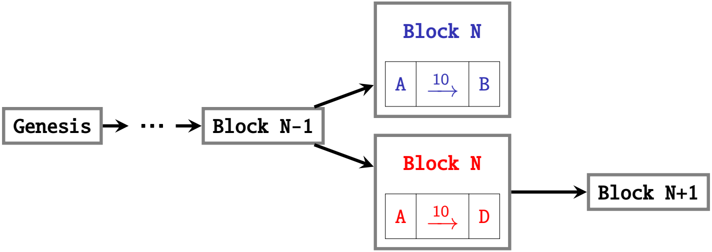
Back to confirmation level



Back to confirmation level



Back to confirmation level



To trigger a fork, Alice could

- Send two transactions in a short time window
- Send two transactions to separate halves of the network
- Pre-mine one block and only reveal it after the first transaction is sent to the network

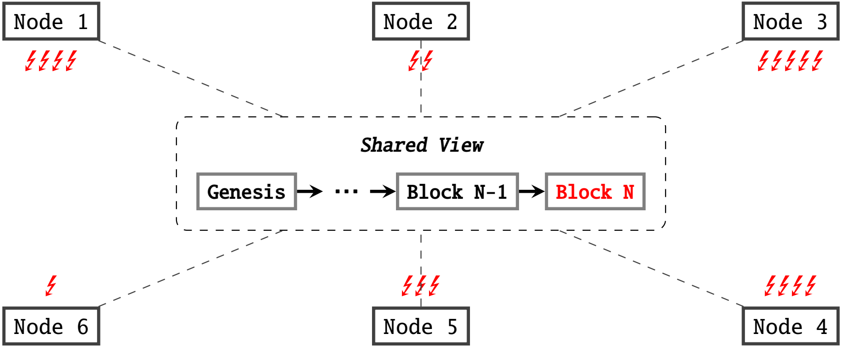
Drawbacks of Proof-of-Work consensus

- Speed of confirmation
  - E.g., a Bitcoin transaction takes on average 10 minutes to confirm
  - Even worse, it is advised to wait for 6 confirmations, i.e., 1 hour.
- Vulnerable to 51% attacks
  - In 2014, mining pool `Ghash.io` obtained 51% hash rate in Bitcoin
  - Bitcoin Gold, was hit by such attacks twice in 2018 and 2020
- Energy consumption
  - Hashing itself is not useful
  - And such useless operations are repeated across the fleet of nodes

Outline

1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

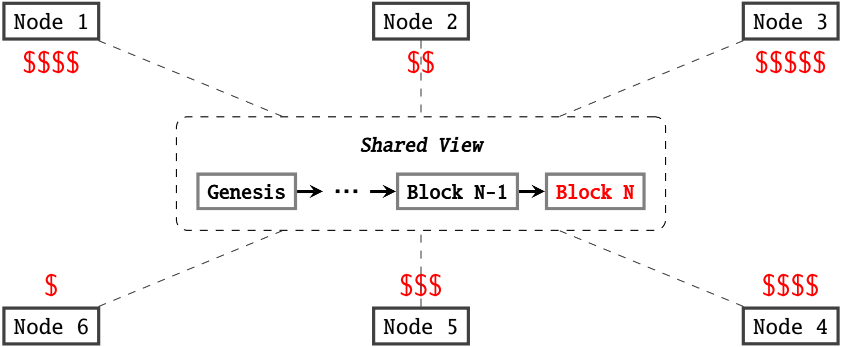
Block production as election



In a proof-of-work scheme,

- the chance of which node is elected to propose a new block is proportional to its hashing power
- collisions are allowed and are resolved by the longest chain rule

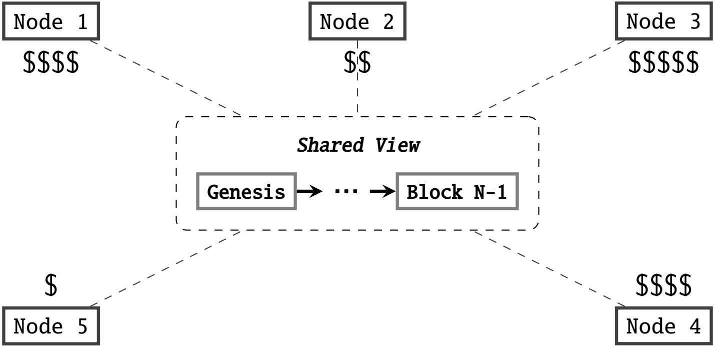
Block production as election



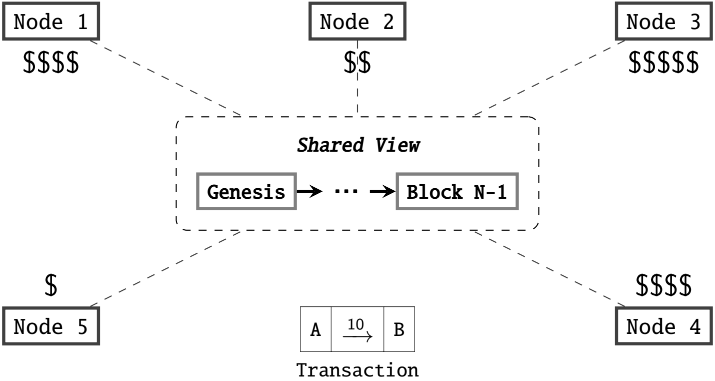
In a proof-of-stake scheme,

- the chance of which node is elected to propose a new block is proportional to its staked value
- collisions are not allowed by design, only the leader creates a block

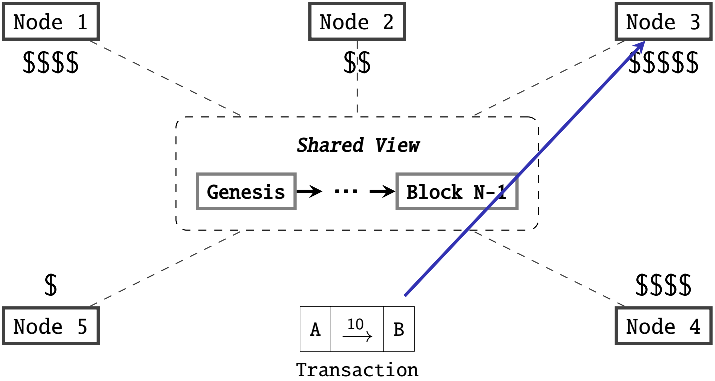
Transaction lifecycle in PoS



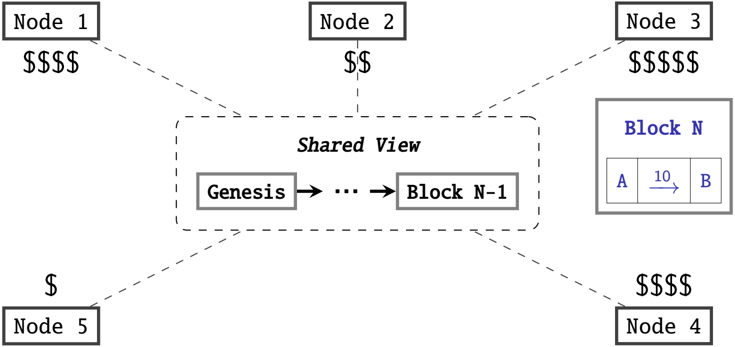
Transaction lifecycle in PoS



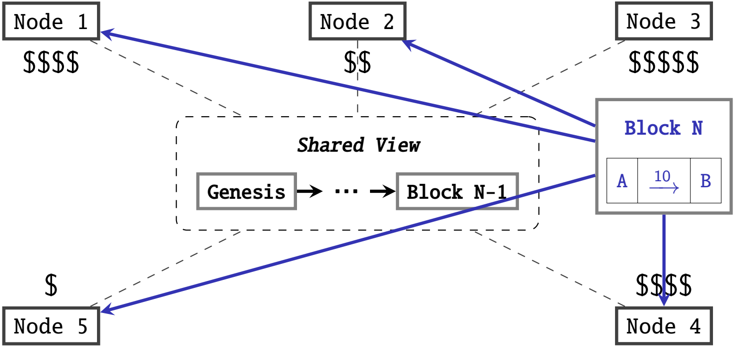
Transaction lifecycle in PoS



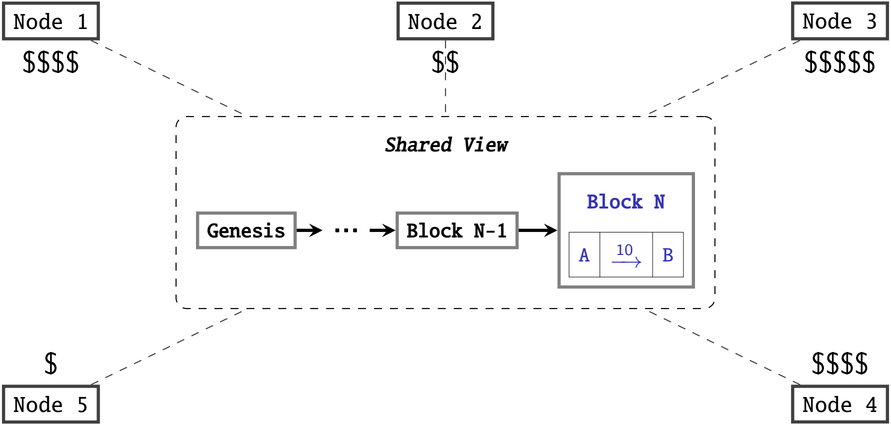
Transaction lifecycle in PoS



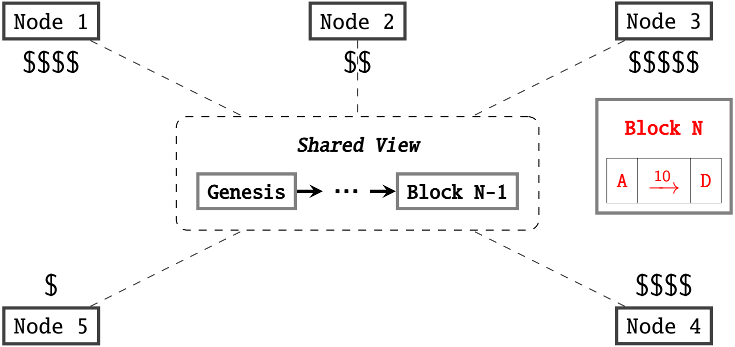
Transaction lifecycle in PoS



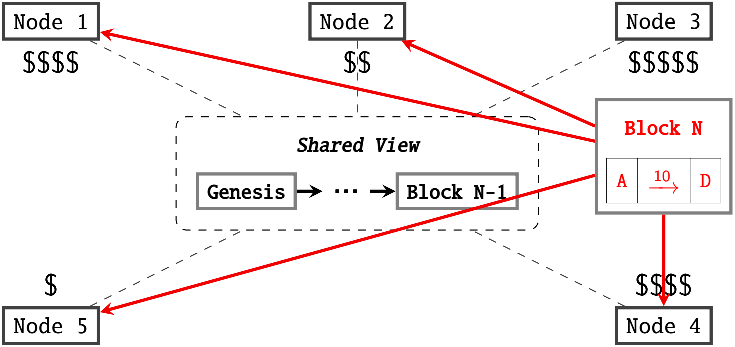
Transaction lifecycle in PoS



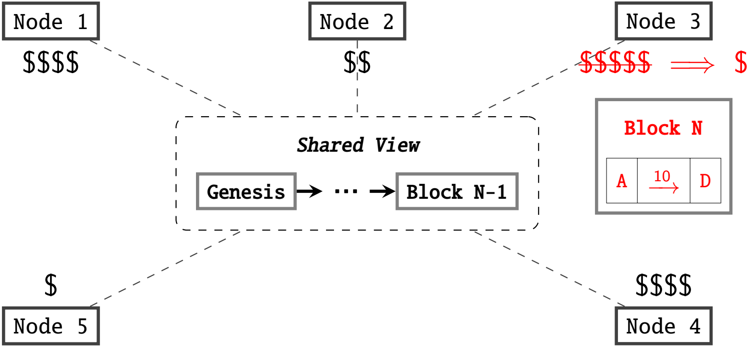
Transaction lifecycle in PoS



Transaction lifecycle in PoS



Transaction lifecycle in PoS



Catching lies

- If a validator node gets caught lying, its stake is burned!
- Other nodes may catch a fraudulent block by comparing it with the transaction that Alice intended to perform
    - e.g., by checking Ethereum's "mempool"
- This works as long as the attacker does not control a majority of stake in the system

The 51% attack in PoS

Q: What if the attacker controls $\geq$ 50% of staked resources?

A: The attacker can prove fraudulent transactions.

The 51% attack in PoS

Q: What if the attacker controls $\geq$ 50% of staked resources?

A: The attacker can prove fraudulent transactions.

Q: Is 51% attack less likely in PoS compared with PoW?

The 51% attack in PoS

Q: What if the attacker controls $\geq$ 50% of staked resources?

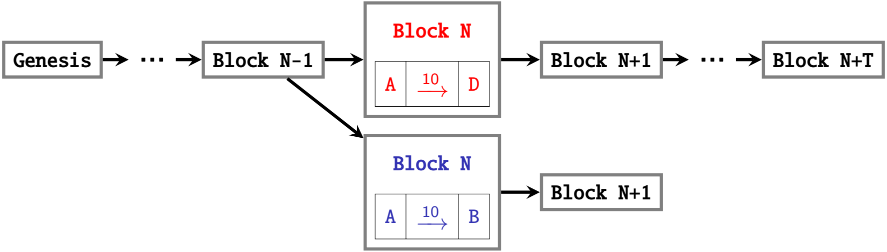
A: The attacker can prove fraudulent transactions.

Q: Is 51% attack less likely in PoS compared with PoW?

A: Yes, because in PoS, the attacker loses the weapon to future attacks, i.e., all the stake are gone, and is not easily recoverable!

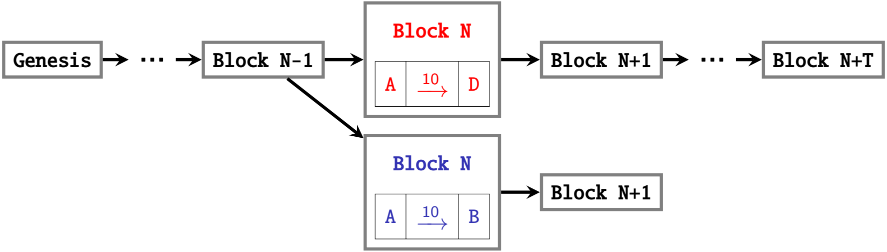
Hard fork as a recovery of a 51% attack

To recover from a 51% attack, the only solution is to hard fork the blockchain in order to invalidate the fraudulent transactions added by the attackers.



Hard fork as a recovery of a 51% attack

To recover from a 51% attack, the only solution is to hard fork the blockchain in order to invalidate the fraudulent transactions added by the attackers.



NOTE: the forked chain can be shorter than the previous chain!

$\Longrightarrow$ a higher level of social coordination is required

Hard fork as a recovery of a 51% attack

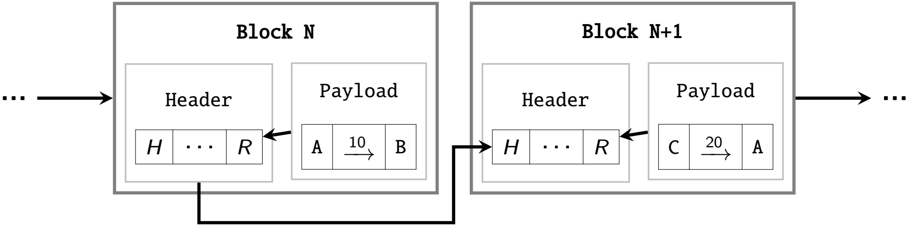
In PoS, we do a hard fork to invalidate fraudulent transactions AND wipe out the attacker who controls $\geq$ 50% of the staked resources.

In PoW, the hard fork can only invalidate transaction WHILE the $\geq$ 50% computational power is still controlled by the attacker.

Chain validation

If Alice shows Bob, the Pizzeria owner, the following blockchain, why would Bob accept it? Why would Bob believe that

- It is hard for Alice to produce such a chain of blocks
- There does not exist a better chain of blocks as of now



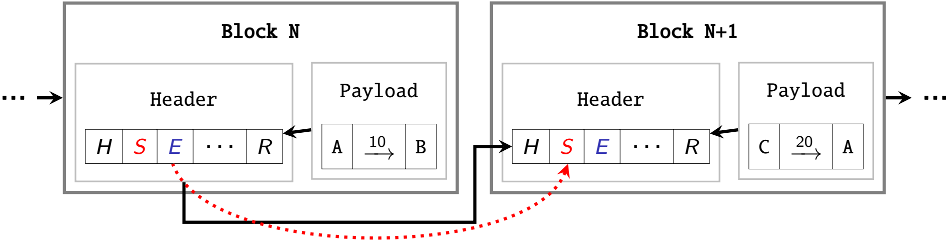
- With PoS, forging a blockchain would be easy!

Chain validation

This turns out to be an extremely complicated problem!

Chain validation

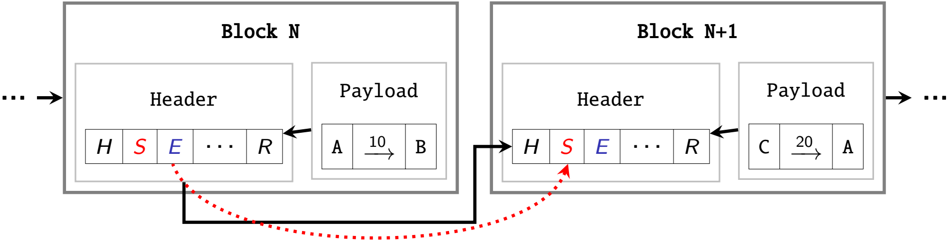
This turns out to be an extremely complicated problem!



- S - Signature of the proposer of this block
- E - Election packet that records how this proposer is elected

Chain validation

This turns out to be an extremely complicated problem!

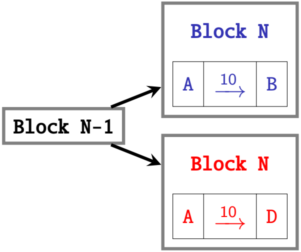


- S - Signature of the proposer of this block
- E - Election packet that records how this proposer is elected

Q: What are the issues with this scheme?

The Nothing-at-Stake problem

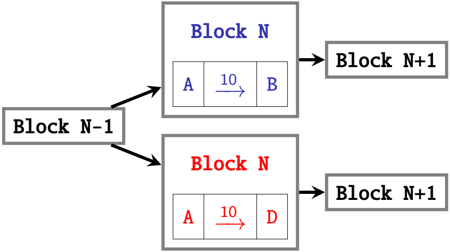
Assuming Alice has some stake (e.g., 1%) and can be elected as a block proposer:



In one of her turn as a block proposer, Alice triggers a fork in the chain with an attempt to double-spend.

The Nothing-at-Stake problem

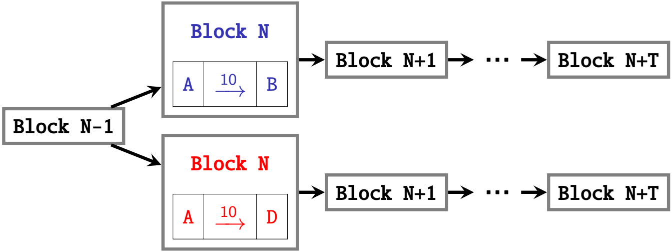
Assuming Alice has some stake (e.g., 1%) and can be elected as a block proposer:



The next block proposer, even honest, has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future, the logical thing to do is to mine on both.

The Nothing-at-Stake problem

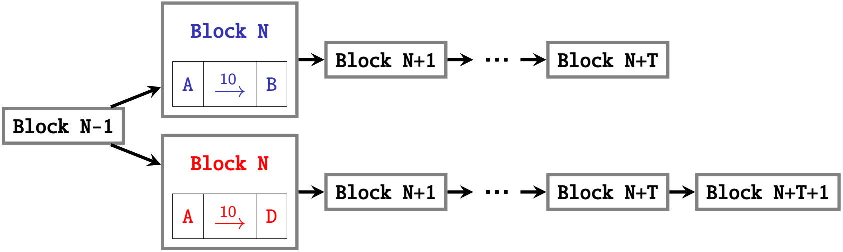
Assuming Alice has some stake (e.g., 1%) and can be elected as a block proposer:



The next block proposer, even honest, has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future, the logical thing to do is to mine on both.

The Nothing-at-Stake problem

Assuming Alice has some stake (e.g., 1%) and can be elected as a block proposer:



When its Alice's turn again, she only append a block to the chain that is more favorable to her. The other chain dies as a result. This is sometimes called the 1% attack.

The Nothing-at-Stake problem

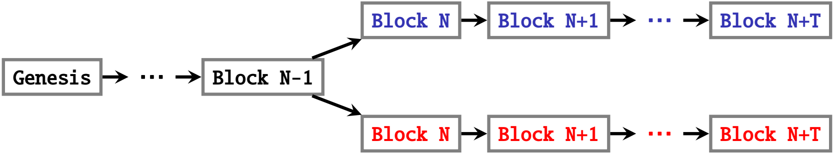
Solution? There is no common solution. Different PoS chains adopt different mechanisms.

The Slash protocol (Ethereum PoS candidate) has two rules:

- Penalize those who “equivocated” on a given block, i.e., voted on two different versions of it.
- Penalize those who voted on the wrong block, regardless of whether or not they double-voted.

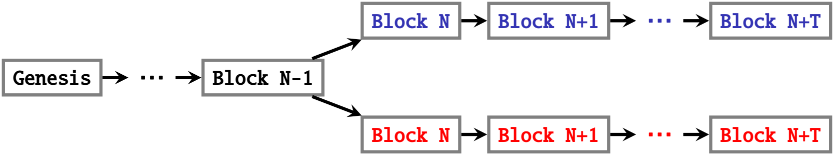
Long-range attacks (the bootstraping problem)

- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Long-range attacks (the bootstraping problem)

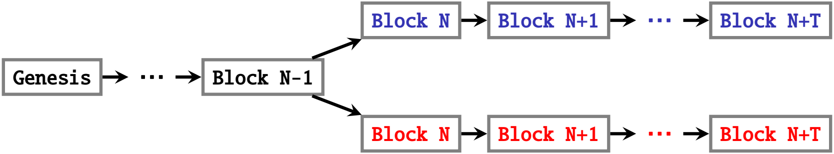
- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why this is not a problem in PoW?

Long-range attacks (the bootstraping problem)

- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why this is not a problem in PoW?

A: Because it is computationally expensive to create a counterfeit chain in PoW. But it is easy (almost no cost) in the PoS case.

Long-range attacks (the bootstraping problem)

Solution? In short, there is no simple solutions.

- Casper (Ethereum's PoS protocol) depends on trusted nodes to broadcast the correct block hash.
- Peercoin, broadcasts the hash of the "legitimate" chain on a daily basis.
- Extremely complicated solutions have been proposed e.g., Ouroboros Genesis.