it’s everyone’s favorite time of the year!

http://registrar.mit.edu/subjectevaluation

6.033 in the news

Gas Pipeline Hack Leads to Panic Buying in the Southeast

The national average for a gallon of regular gasoline rose 2 cents on Tuesday, and some airlines began to take small measures in response to the shutdown.

But the company said nothing about what factors will play into its decision on when to restart the pipeline. And it has not explained whether it found any evidence that the malware placed in its data systems could migrate to the operations of the pipeline.

Several experts noted that while the two networks are described as separate entities, they have considerable crossover. For example, one of the systems the ransomware group tied up tracks how much fuel each customer uses. Without that running, Colonial would not know how much fuel any of its customers were receiving — or how to get paid for it.

Higher fuel prices affect lower-income people the most because they spend the highest percentage of their incomes on gasoline and typically drive less fuel-efficient vehicles. That makes gasoline prices a potential political issue after they've been relatively low for several years.

Several airports around the South and the Washington area may be affected over the next few days because they are connected to the pipeline and typically retain only a few days' supply.

https://www.nytimes.com/2021/05/11/business/colonial-pipeline-shutdown-latest-news.html

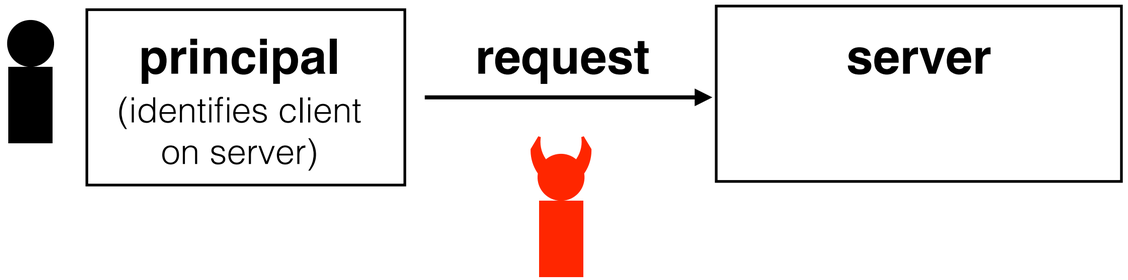
6.033 Spring 2021

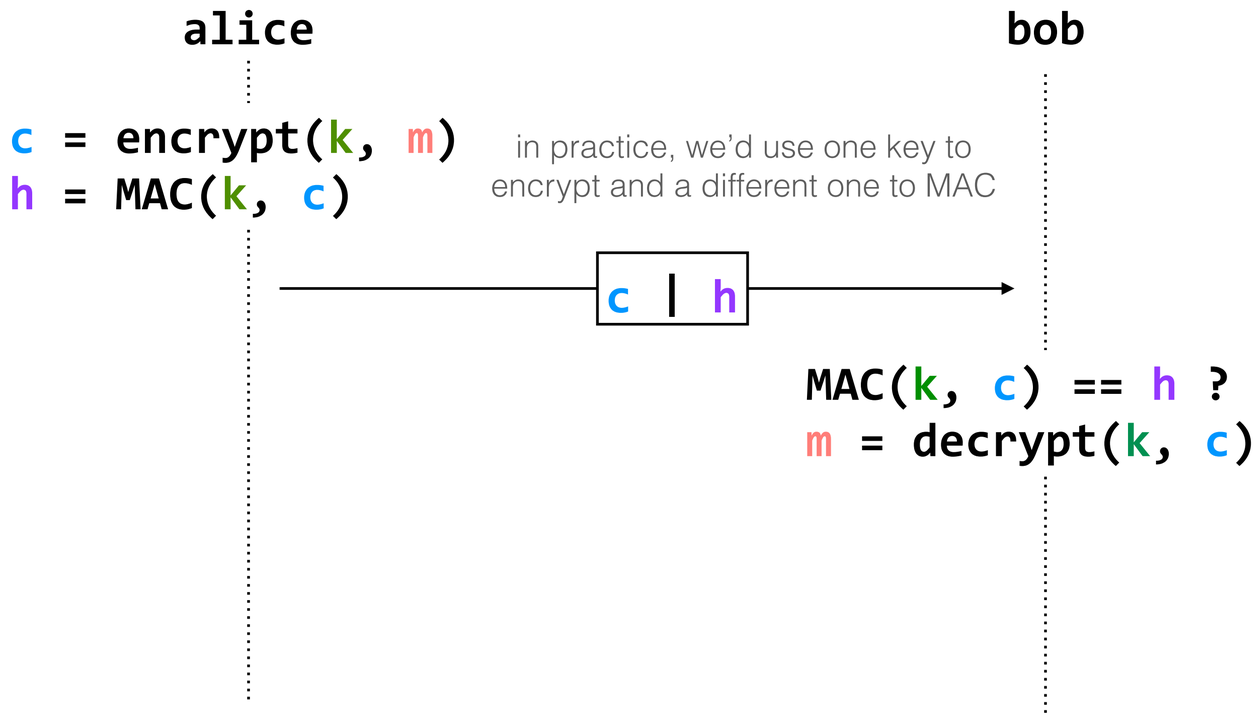
Lecture #23: TOR

what to do when secure channels aren’t enough

Katrina LaCurts | lacurts@mit.edu | 6.033 2021

this week, we're going to turn to adversaries that are observing data on the network





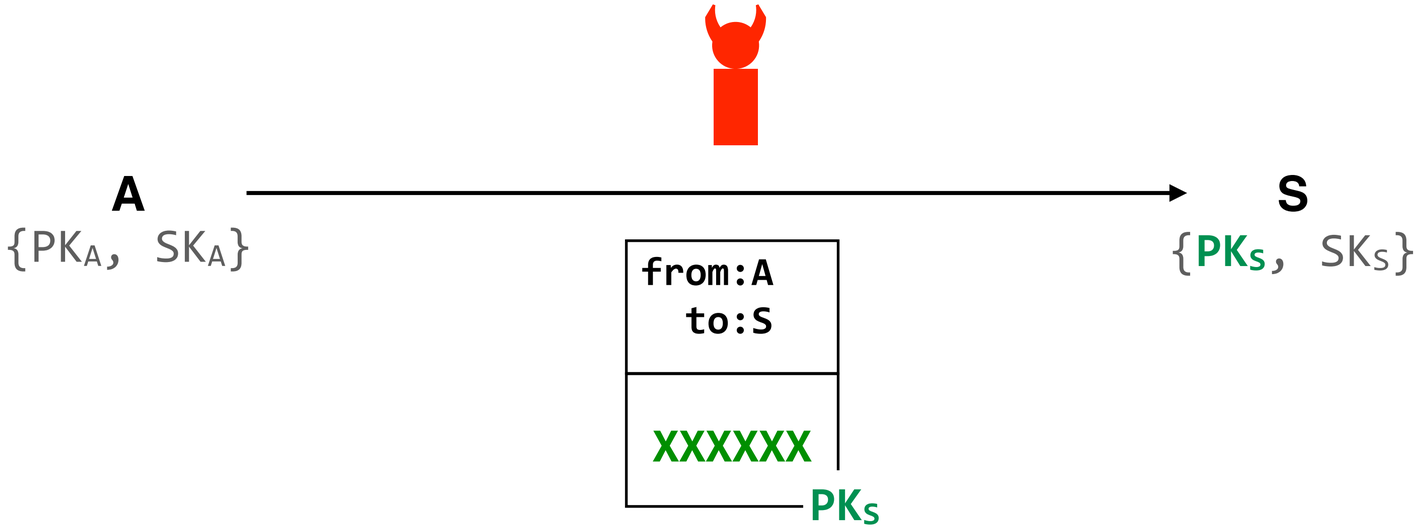
policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

```
encrypt(PKx, m) = c
decrypt(SKx, c) = m
```

this is different from how you saw public-key cryptography used for signatures, and different from how you saw symmetric keys used for encryption



alice is encrypting data **to** S using its public key

problem: packet header exposes to the adversary that **A** is communicating with **S**

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

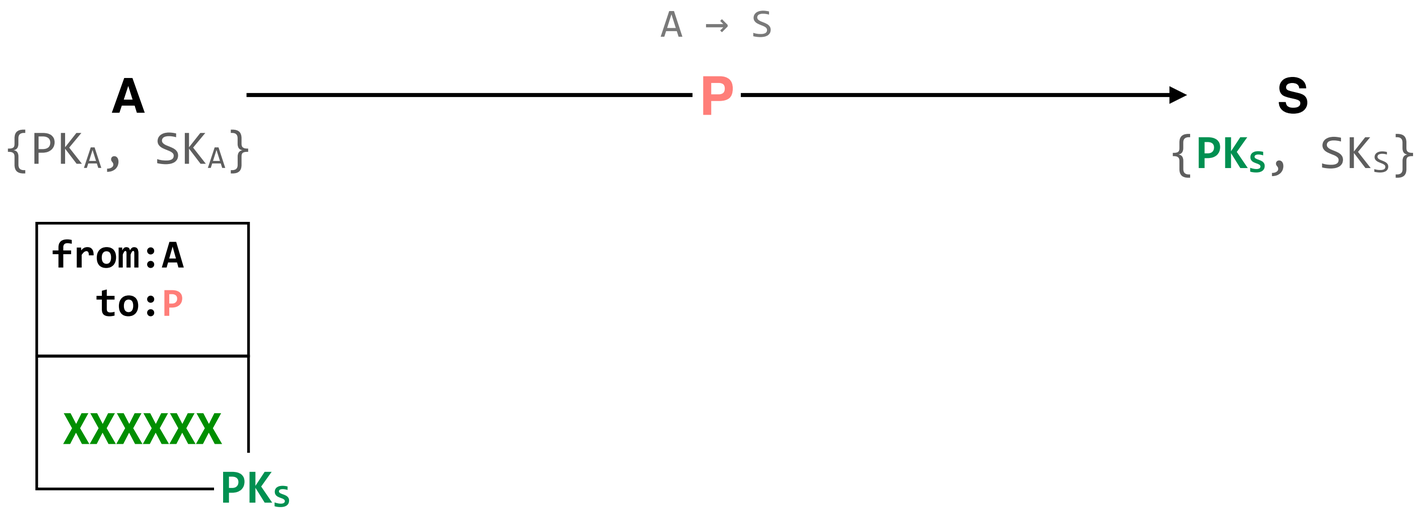
public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$

things to avoid

no packet should say "from: A; to: S"



policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

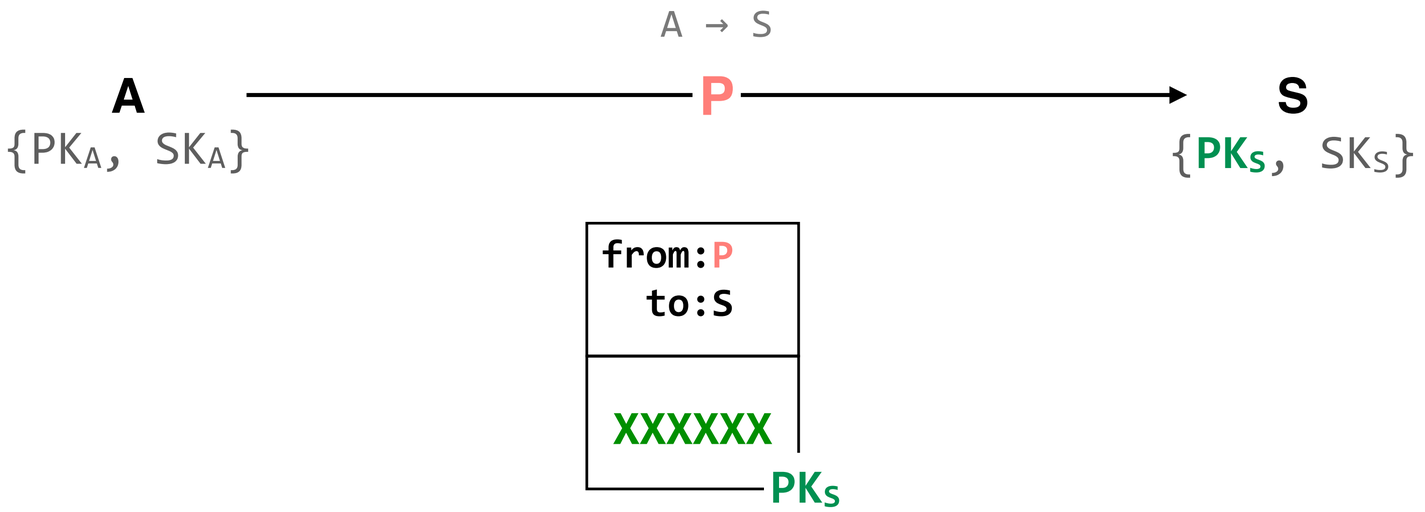
public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$

things to avoid

no packet should say "from: A; to: S"



policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

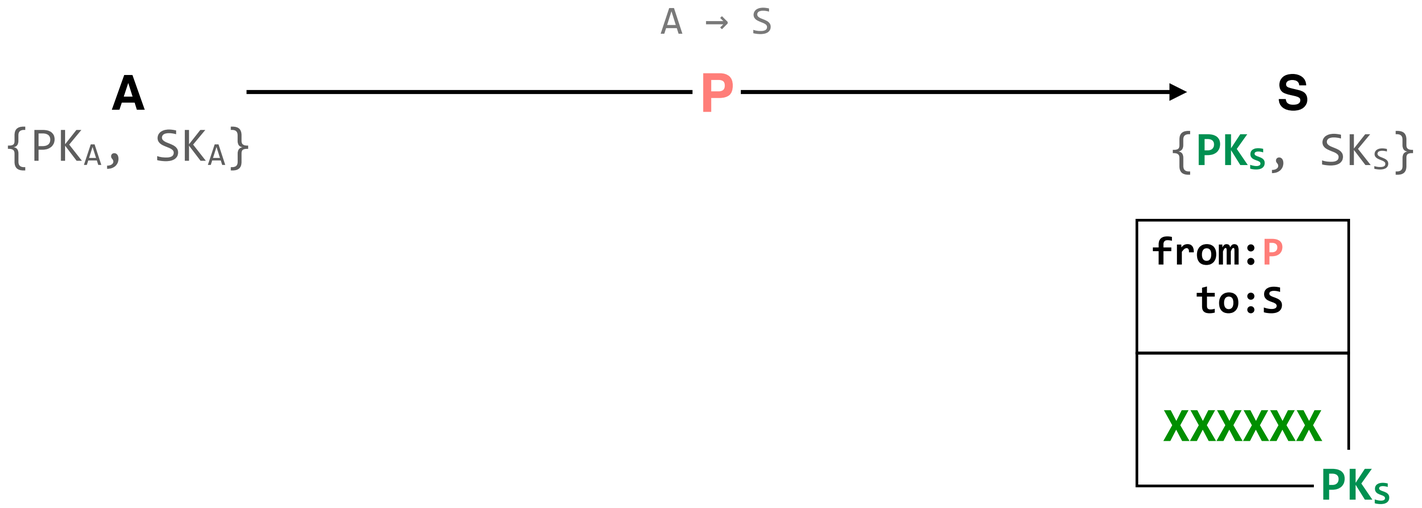
$$encrypt(PK_x, m) = c$$
$$decrypt(SK_x, c) = m$$

things to avoid

no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S



problem: P knows that **A** is communicating with **S**

a single proxy alone *can* be useful for other things;
we'll return to this later in the lecture.

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

`encrypt(PKx, m) = c`

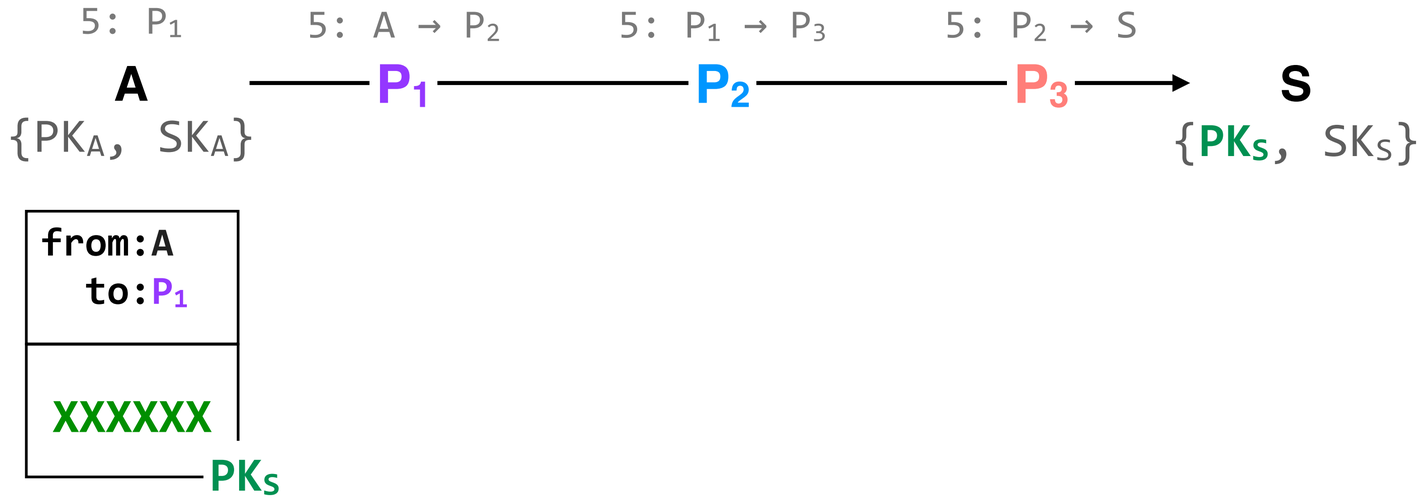
`decrypt(SKx, c) = m`

things to avoid

no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S



policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

`encrypt(PKx, m) = c`

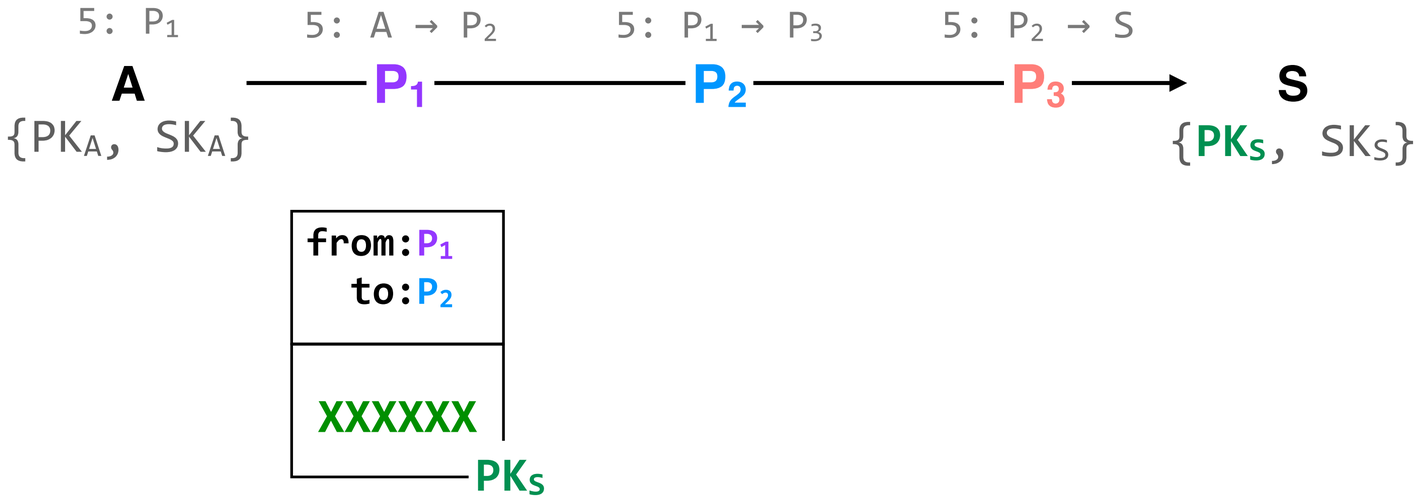
`decrypt(SKx, c) = m`

things to avoid

no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S



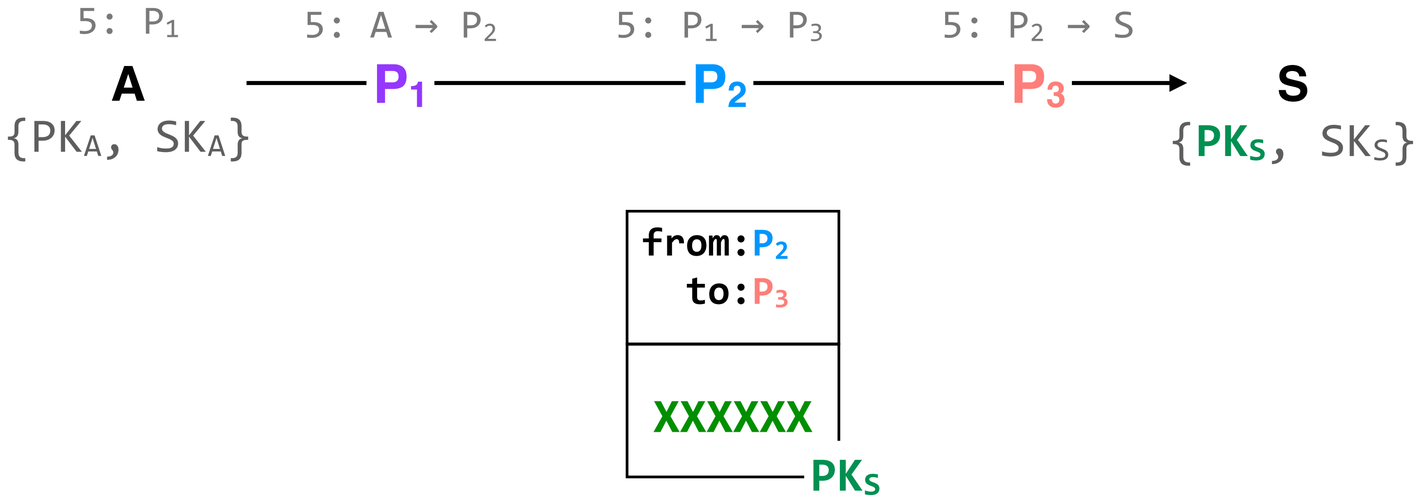
policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$



things to avoid

no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

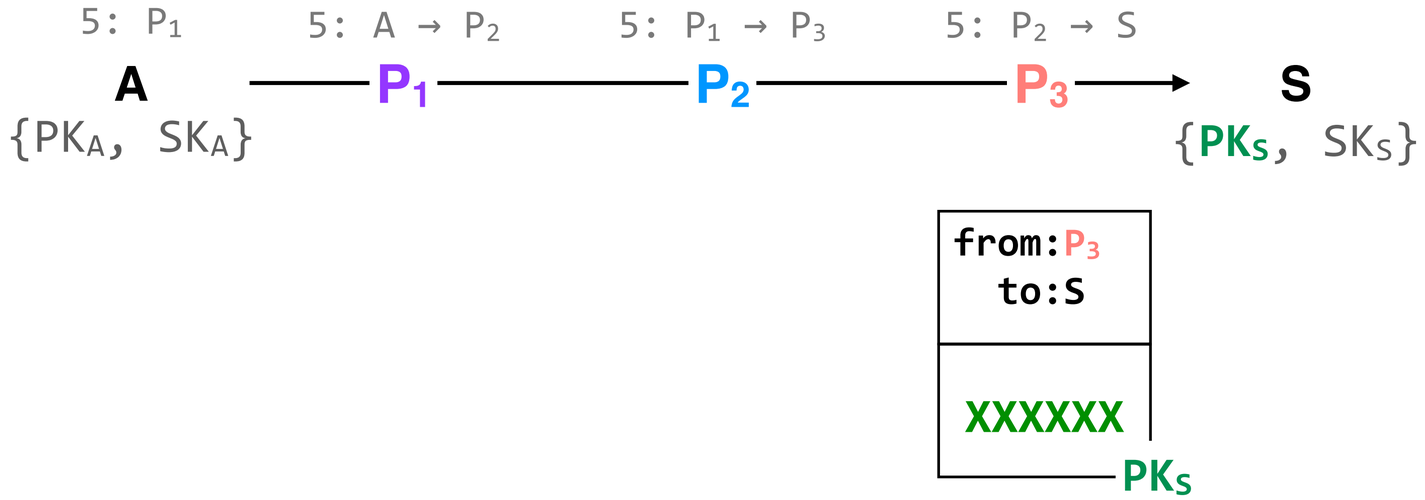
policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$



things to avoid

no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

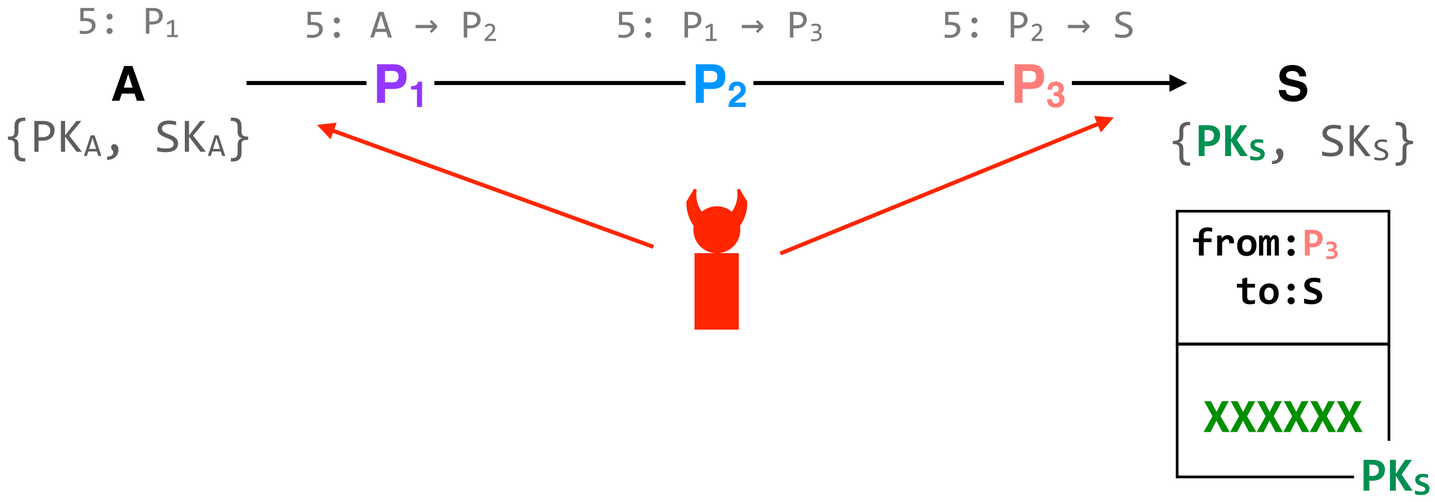
$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

things to avoid

- no packet should say "from: A; to: S"
- no entity in the network should receive a packet from A and send it directly to S
- no entity in the network should keep *state* that links A to S

data should not appear the same across packets



problem: an adversary with multiple vantage points can observe the same data traveling from **A** to **S**

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

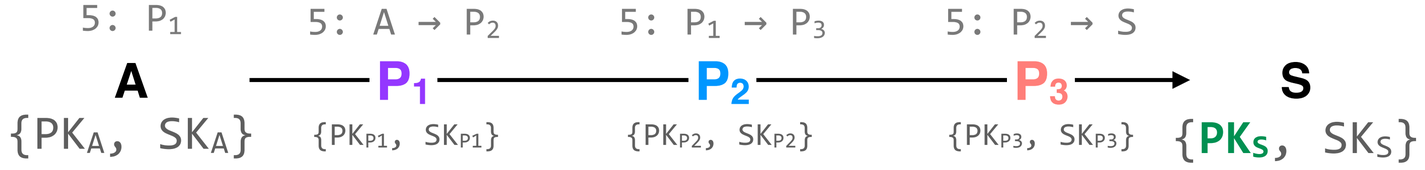
things to avoid

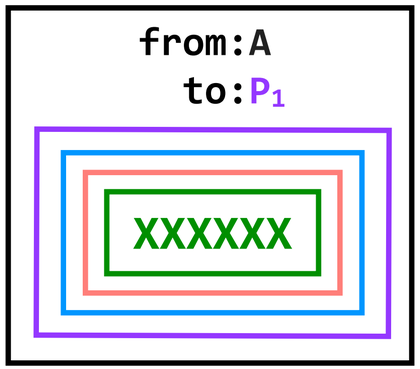
no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

data should not appear the same across packets





tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

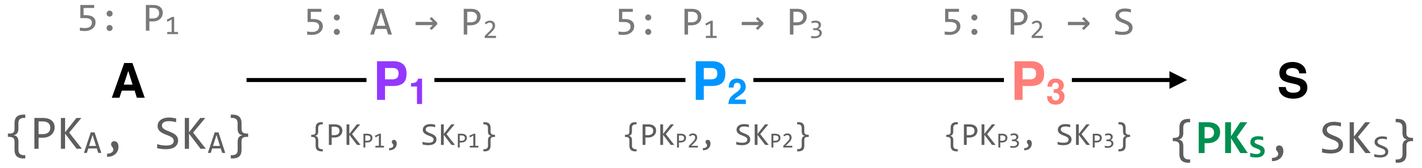
things to avoid

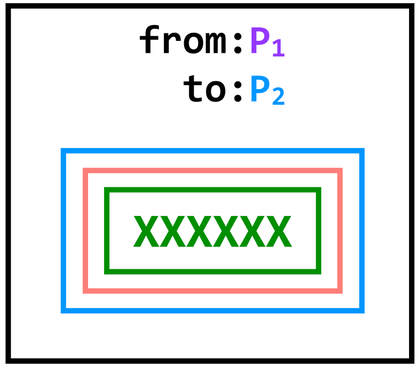
no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

data should not appear the same across packets





tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

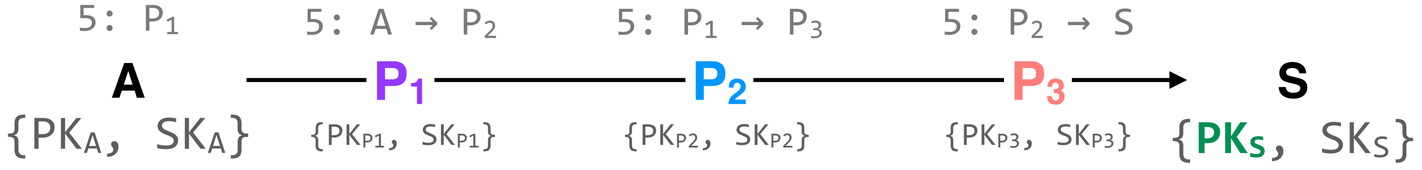
things to avoid

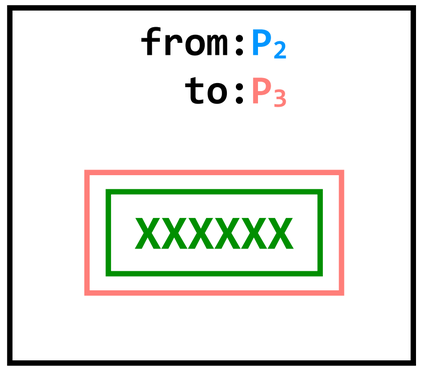
no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

data should not appear the same across packets





tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

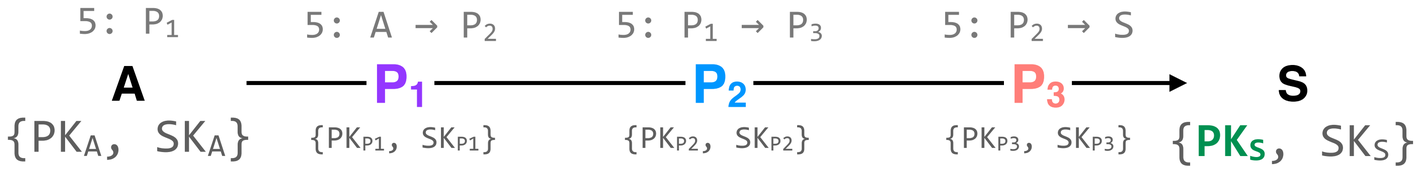
things to avoid

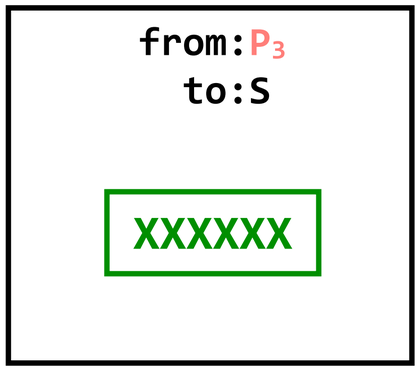
no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

data should not appear the same across packets





tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

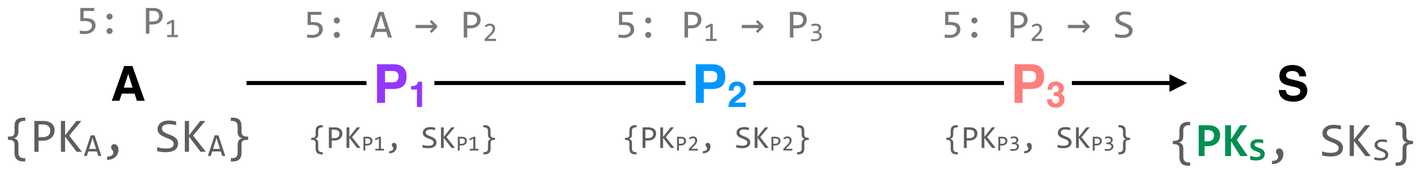
things to avoid

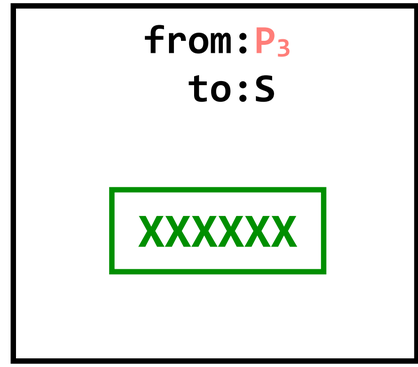
no packet should say "from: A; to: S"

no entity in the network should receive a packet from A and send it directly to S

no entity in the network should keep *state* that links A to S

data should not appear the same across packets





tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

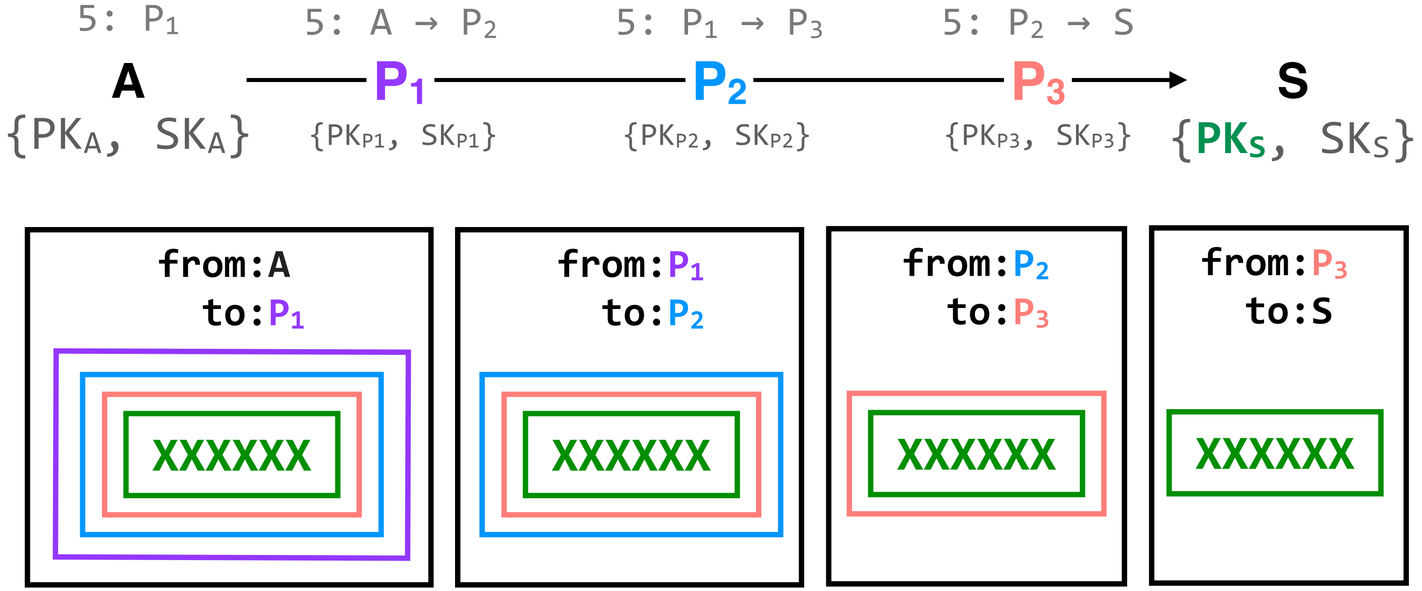
public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

$$encrypt(PK_x, m) = c$$

$$decrypt(SK_x, c) = m$$

things to avoid

- no packet should say "from: A; to: S"
- no entity in the network should receive a packet from A and send it directly to S
- no entity in the network should keep *state* that links A to S
- data should not appear the same across packets



tor adds layers of (public-key) encryption that nodes on the path can strip off as the packet traverses the network

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

So I'm totally anonymous if I use Tor?

No.

First, Tor protects the network communications. It separates where you are from where you are going on the Internet. What content and data you transmit over Tor is controlled by you. If you login to Google or Facebook via Tor, the local ISP or network provider doesn't know you are visiting Google or Facebook. Google and Facebook don't know where you are in the world. However, since you have logged into their sites, they know who you are. If you don't want to share information, you are in control.

Second, active content, such as Java, Javascript, Adobe Flash, Adobe Shockwave, QuickTime, RealAudio, ActiveX controls, and VBScript, are binary applications. These binary applications run as your user account with your permissions in your operating system. This means these applications can access anything that your user account can access. Some of these technologies, such as Java and Adobe Flash for instance, run in what is known as a virtual machine. This virtual machine may have the ability to ignore your configured proxy settings, and therefore bypass Tor and share information directly to other sites on the Internet. The virtual machine may be able to store data, such as cookies, completely separate from your browser or operating system data stores. Therefore, these technologies must be disabled in your browser to use Tor safely.

That's where Tor Browser comes in. We produce a web browser that is preconfigured to help you control the risks to your privacy and anonymity while browsing the Internet. Not only are the above technologies disabled to prevent identity leaks, Tor Browser also includes browser extensions like NoScript and Torbutton, as well as patches to the Firefox source code. The full design of Tor Browser can be read here. In designing a safe, secure solution for browsing the web with Tor, we've discovered that configuring other browsers to use Tor is unsafe.

Alternatively, you may find a Live CD or USB operating system more to your liking. The Tails team has created an entire bootable operating system configured for anonymity and privacy on the Internet.

Tor is a work in progress. There is still plenty of work left to do for a strong, secure, and complete solution.

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

What attacks remain against onion routing?

As mentioned above, it is possible for an observer who can view both you and either the destination website or your Tor exit node to correlate timings of your traffic as it enters the Tor network and also as it exits. Tor does not defend against such a threat model.

In a more limited sense, note that if a censor or law enforcement agency has the ability to obtain specific observation of parts of the network, it is possible for them to verify a suspicion that you talk regularly to your friend by observing traffic at both ends and correlating the timing of only that traffic. Again, this is only useful to verify that parties already suspected of communicating with one another are doing so. In most countries, the suspicion required to obtain a warrant already carries more weight than timing correlation would provide.

Furthermore, since Tor reuses circuits for multiple TCP connections, it is possible to associate non anonymous and anonymous traffic at a given exit node, so be careful about what applications you run concurrently over Tor. Perhaps even run separate Tor clients for these applications.

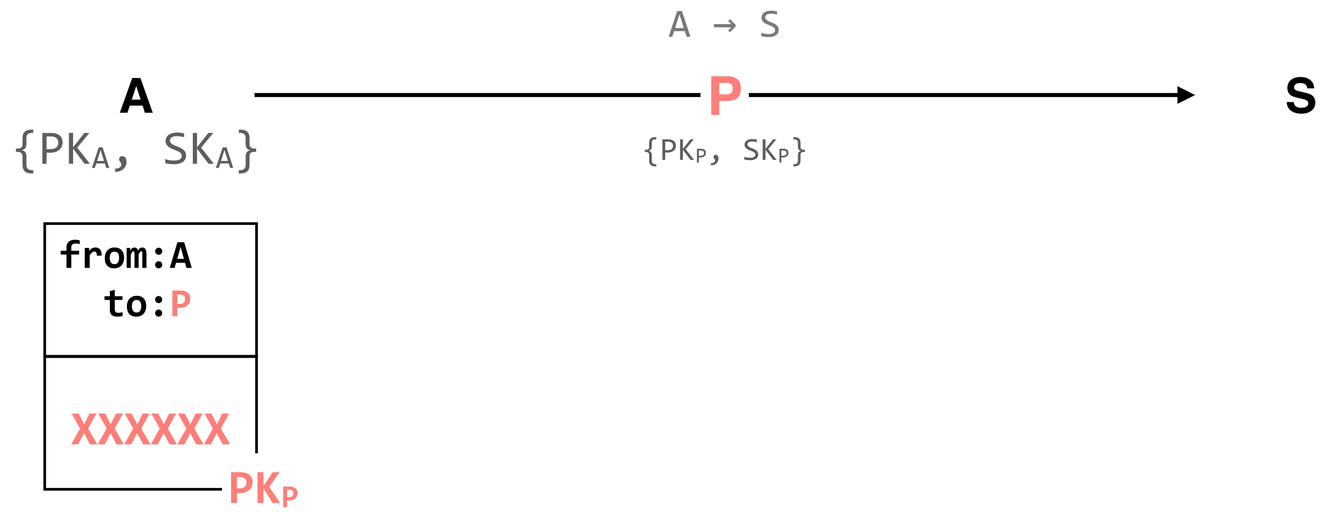
https://www.torproject.org/docs/faq.html.en

policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

```
encrypt(PKx, m) = c
decrypt(SKx, c) = m
```

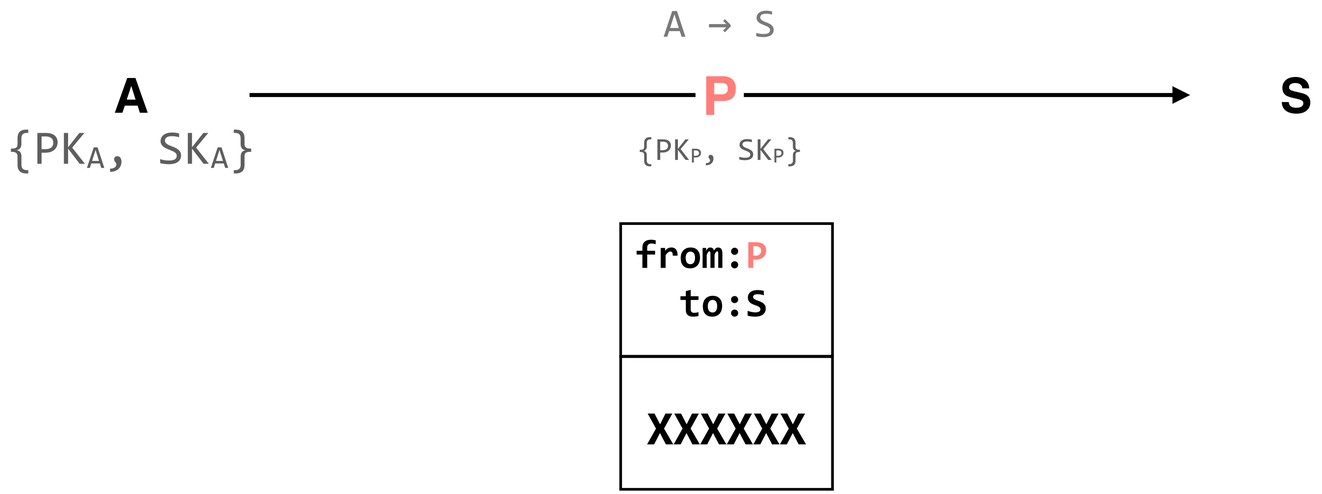


policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

```
encrypt(PKx, m) = c
decrypt(SKx, c) = m
```



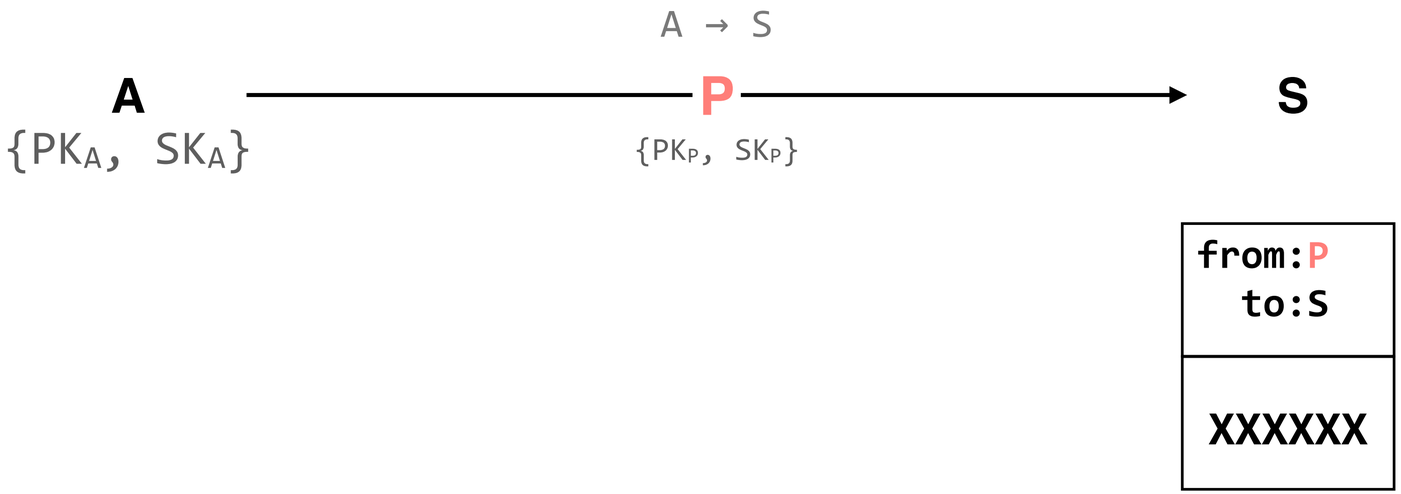
policy: provide **anonymity** (only the client should know that they're communicating with the server)

threat model: adversary is on the path between the client and the server

public-key cryptography: a message to **x** is encrypted with **x**'s public key; only **x**'s secret key can decrypt the message

`encrypt(PKx, m) = c`

`decrypt(SKx, c) = m`



assuming you trust the proxy, this type of service can be useful if you care about confidentiality on a local network

what we've shown here is a simplified version of some of the functionality you get when you use a VPN

tor provides some level of anonymity for users, preventing adversaries from linking a sender to its receiver

there are still ways to attack tor, namely by **correlating traffic** from various points in the network

a larger takeaway here is that a secure channel alone only provides confidentiality and integrity of the message data; **packet headers can reveal information** that may be sensitive in certain contexts