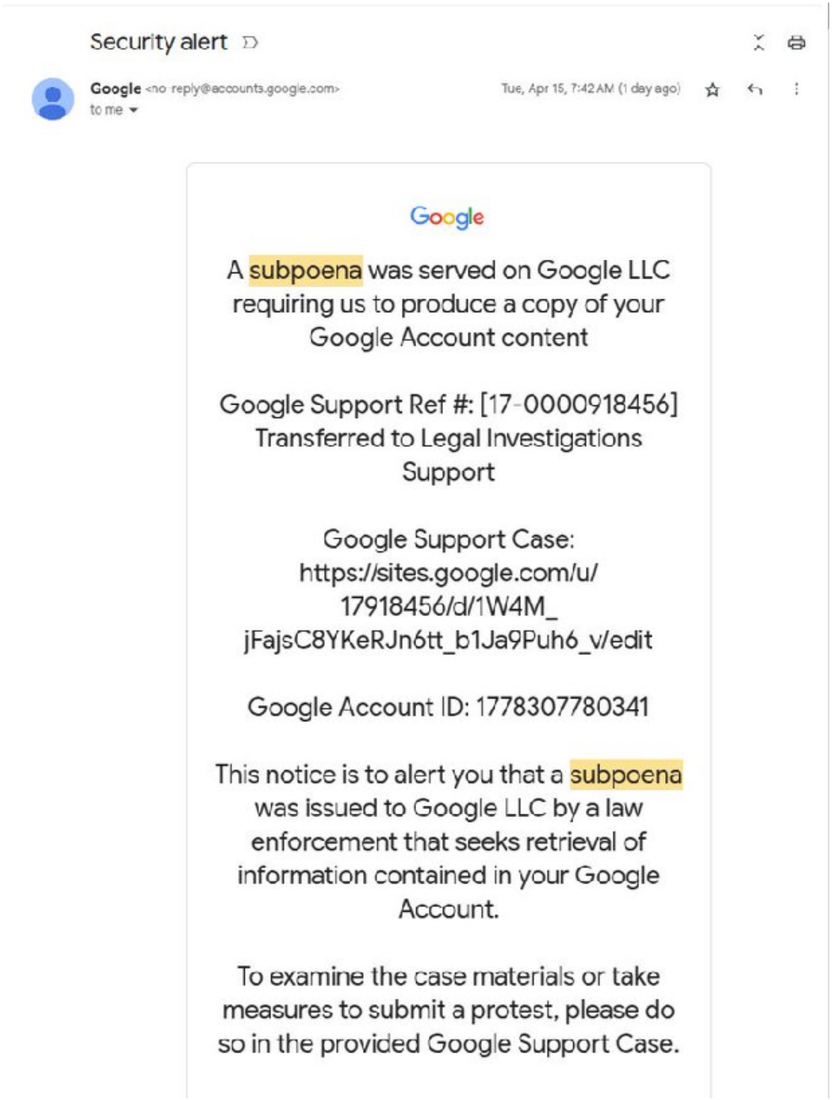
6.1800 Spring 2025

Lecture #24: Tor

what to do when secure channels aren’t enough

Katrina LaCurts | lacurts@mit.edu | 6.1800 2025

6.1800 in the news



Security alert

Google <no-reply@accounts.google.com> Tue, Apr 15, 7:42 AM (1 day ago)
to me

Google

A subpoena was served on Google LLC requiring us to produce a copy of your Google Account content

Google Support Ref #: [17-0000918456] Transferred to Legal Investigations Support

Google Support Case: https://sites.google.com/u/17918456/d/1W4M_jFajsC8YKeRJn6tt_b1Ja9Puh6_v/edit

Google Account ID: 1778307780341

This notice is to alert you that a subpoena was issued to Google LLC by a law enforcement that seeks retrieval of information contained in your Google Account.

To examine the case materials or take measures to submit a protest, please do so in the provided Google Support Case.

All Gmail users at risk from clever replay attack

Posted: April 22, 2025 by Pieter Arntz

Cybercriminals are abusing Google's infrastructure, creating emails that appear to come from Google in order to persuade people into handing over their Google account credentials.

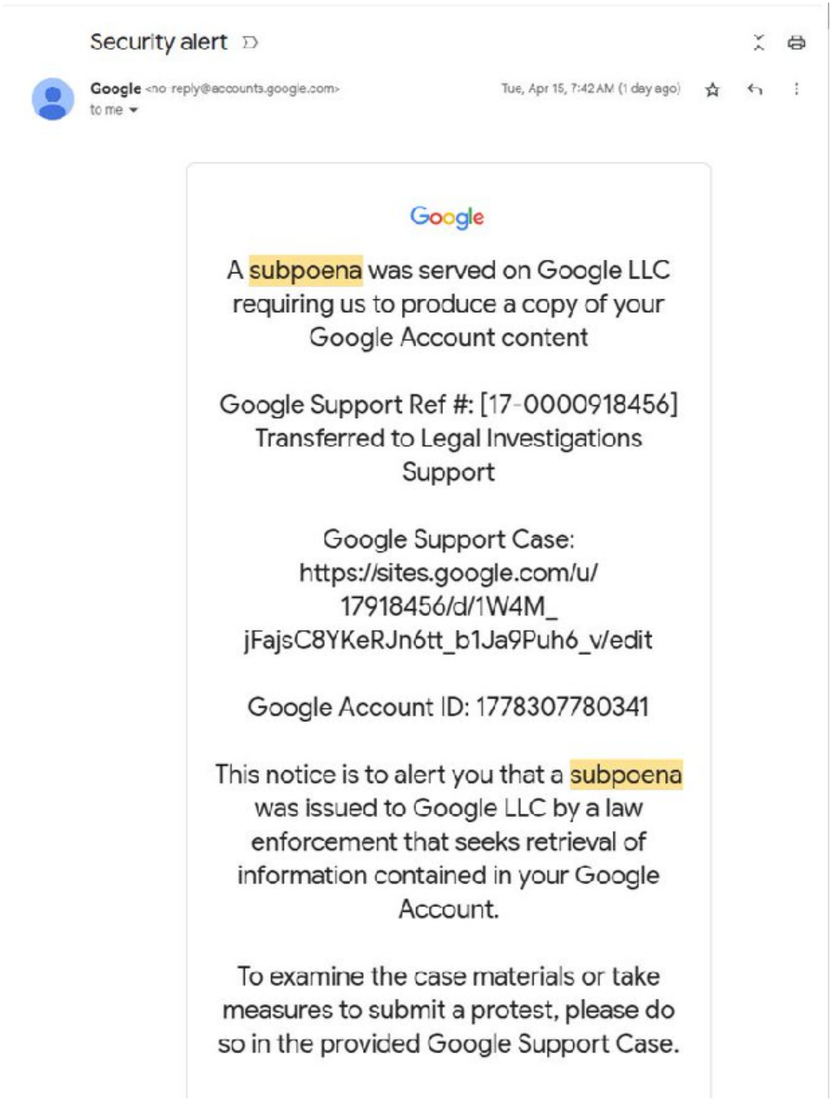
This attack, first flagged by Nick Johnson, the lead developer of the Ethereum Name Service (ENS), a blockchain equivalent of the popular internet naming convention known as the Domain Name System (DNS).

Nick received a very official looking security alert about a subpoena allegedly issued to Google by law enforcement to information contained in Nick's Google account. A URL in the email pointed Nick to a sites.google.com page that looked like an exact copy of the official Google support portal.

As a computer savvy person, Nick spotted that the official site should have been hosted on `accounts.google.com` and not `sites.google.com`. The difference is that anyone with a Google account can create a website on `sites.google.com`. And that is exactly what the cybercriminals did.

https://www.malwarebytes.com/blog/news/2025/04/all-gmail-users-at-risk-by-clever-replay-attack

6.1800 in the news



So, what the cybercriminals did was:

- Set up a Gmail account starting with me@ so the visible email would look as if it was addressed to "me."
- Register an OAuth app and set the app name to match the phishing link
- Grant the OAuth app access to their Google account which triggers a legitimate security warning from `no-reply@accounts.google.com`
- This alert has a valid DKIM signature, with the content of the phishing email embedded in the body as the app name.
- Forward the message untouched which keeps the DKIM signature valid.

Nick submitted a bug report to Google about this. Google originally closed the report as 'Working as Intended,' but later Google got back to him and said it had reconsidered the matter and it will fix the OAuth bug.

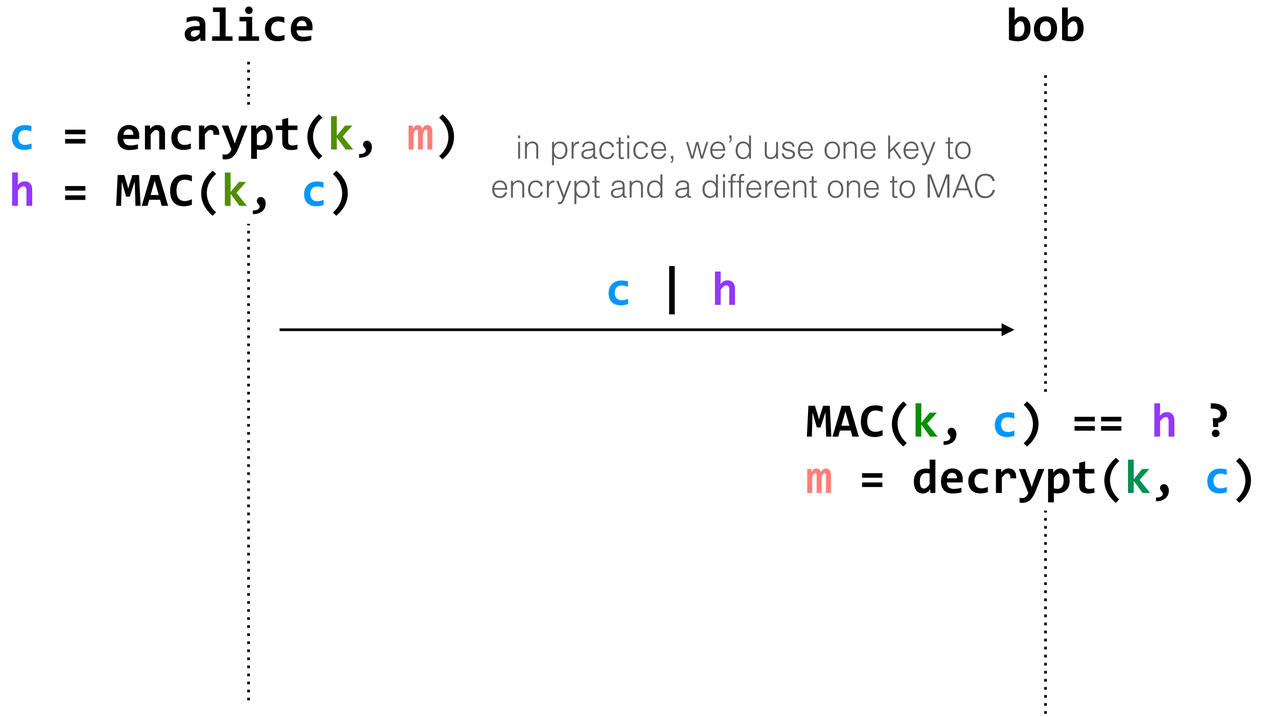
https://www.malwarebytes.com/blog/news/2025/04/all-gmail-users-at-risk-by-clever-replay-attack

today, we're still considering adversaries that are observing data on the network

symmetric-key cryptography: the same key is used to encrypt and decrypt

```
encrypt(key, message) → ciphertext
decrypt(key, ciphertext) → message
```

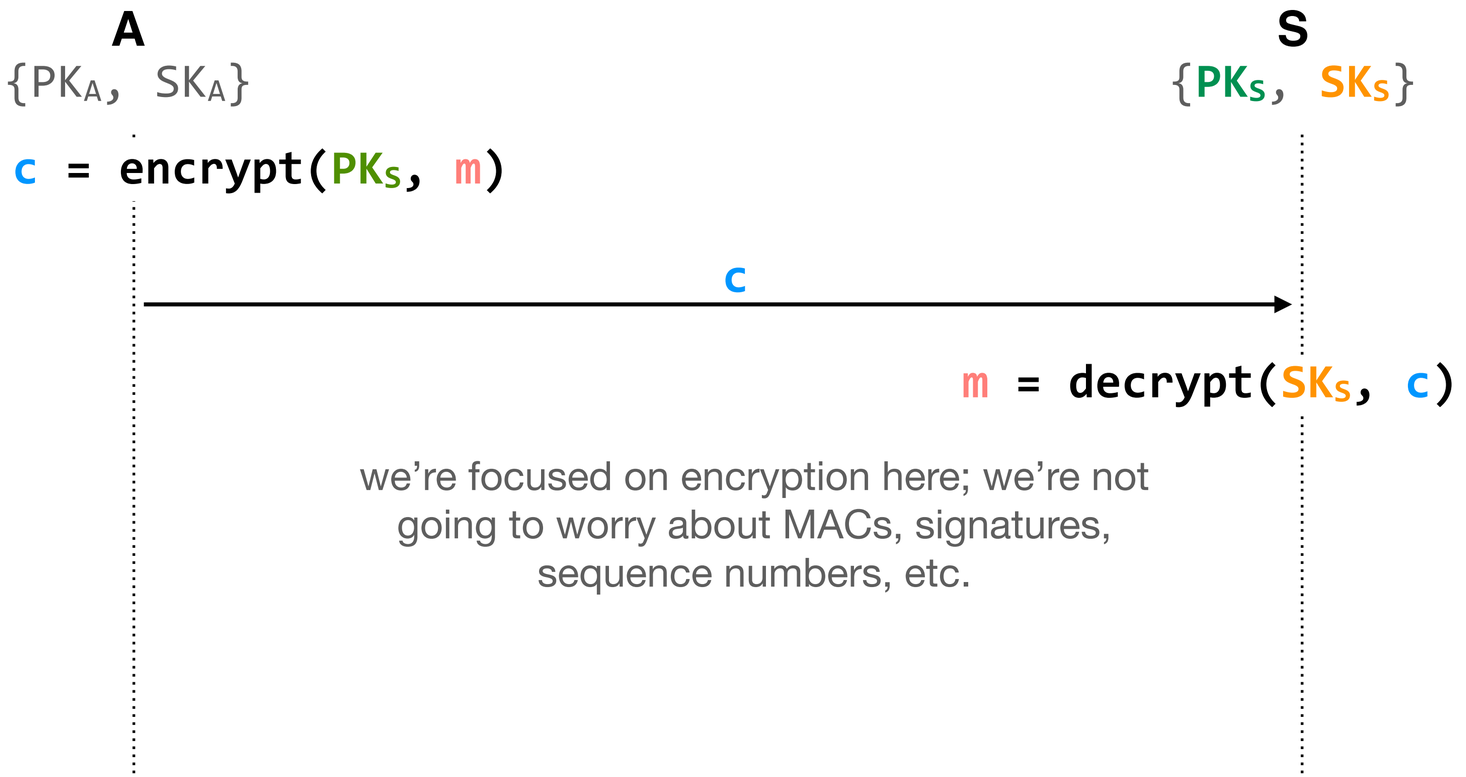
this is how we performed encryption for secure channels in the last lecture. Alice and bob shared the same key, and that key needed to be secret from everyone else (which brought us to the problem of how to exchange the key in the first place)



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

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

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



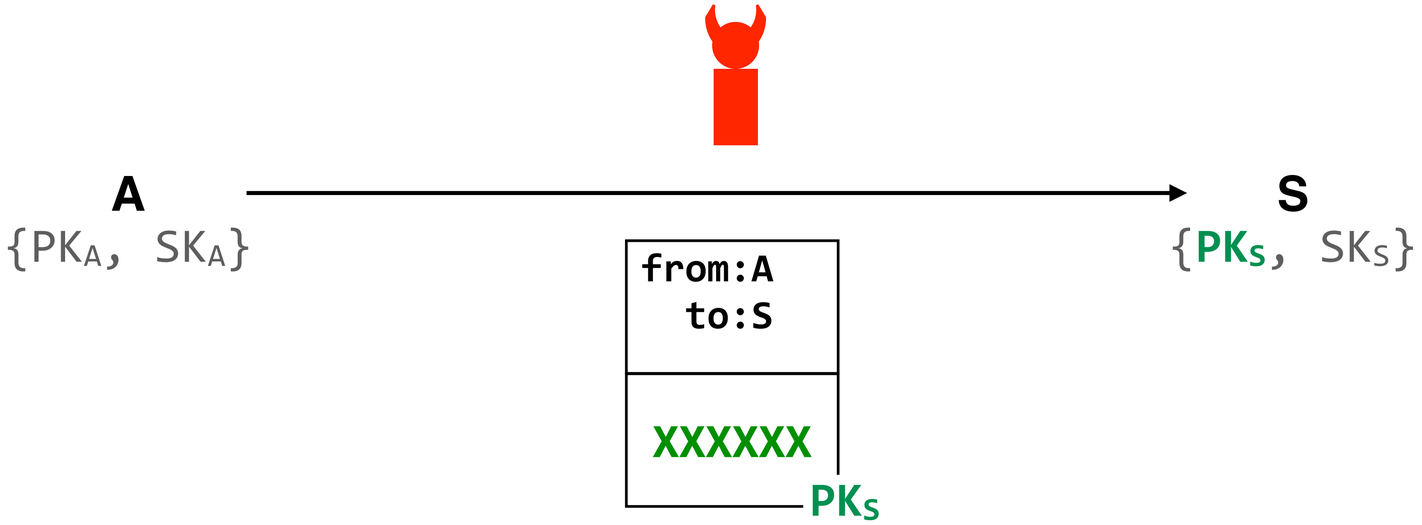
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$



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

things to avoid

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

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

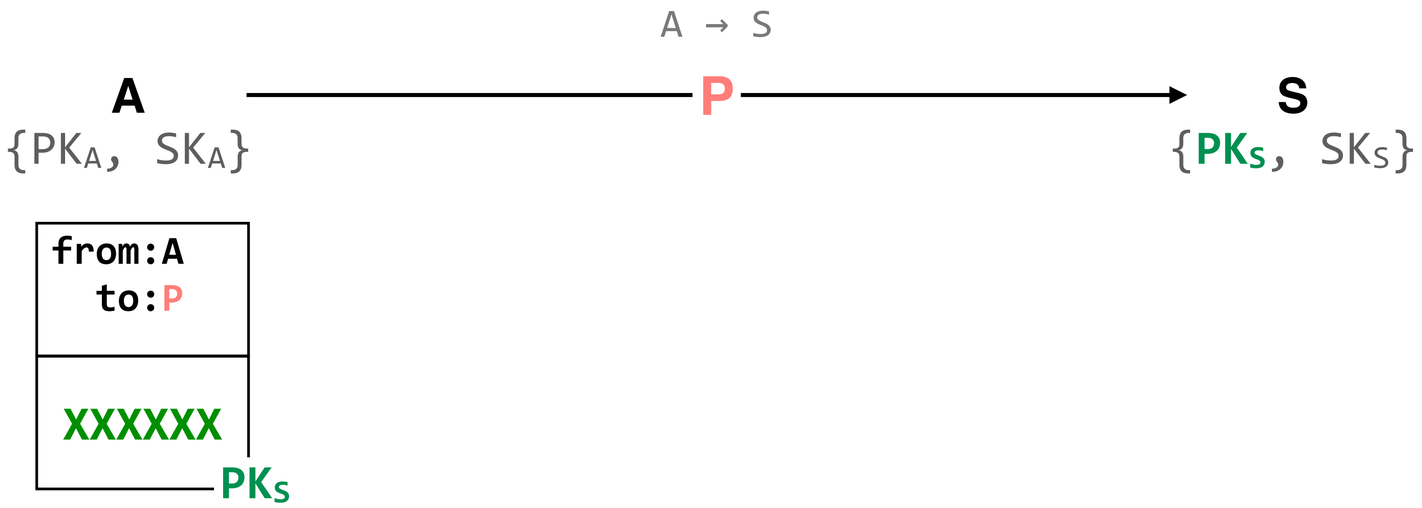
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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"

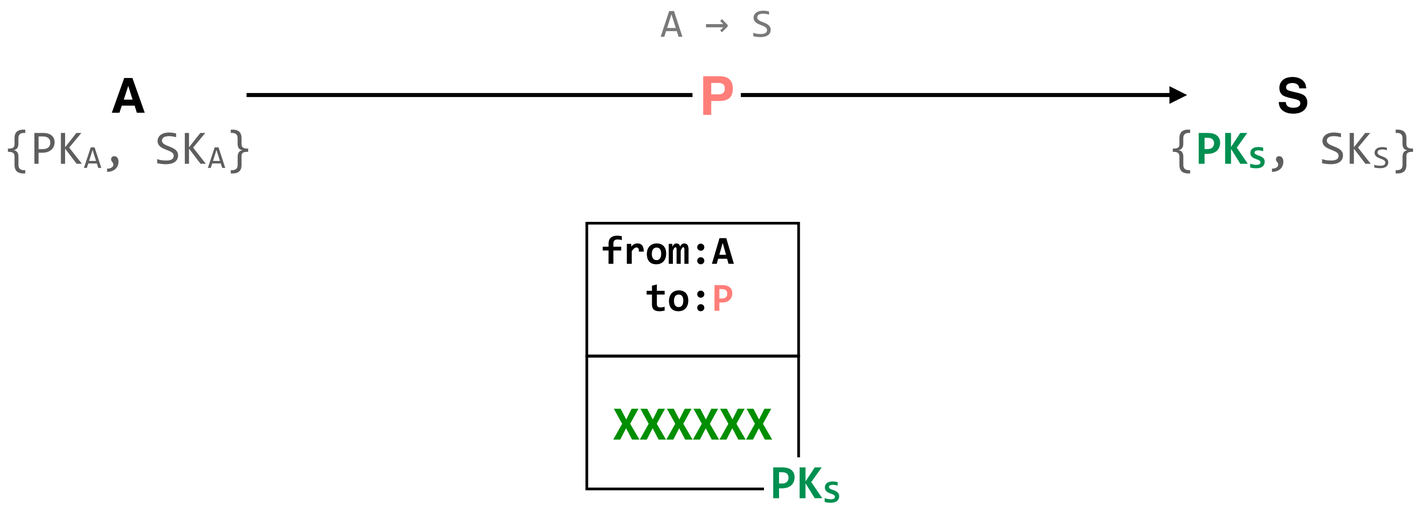
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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"

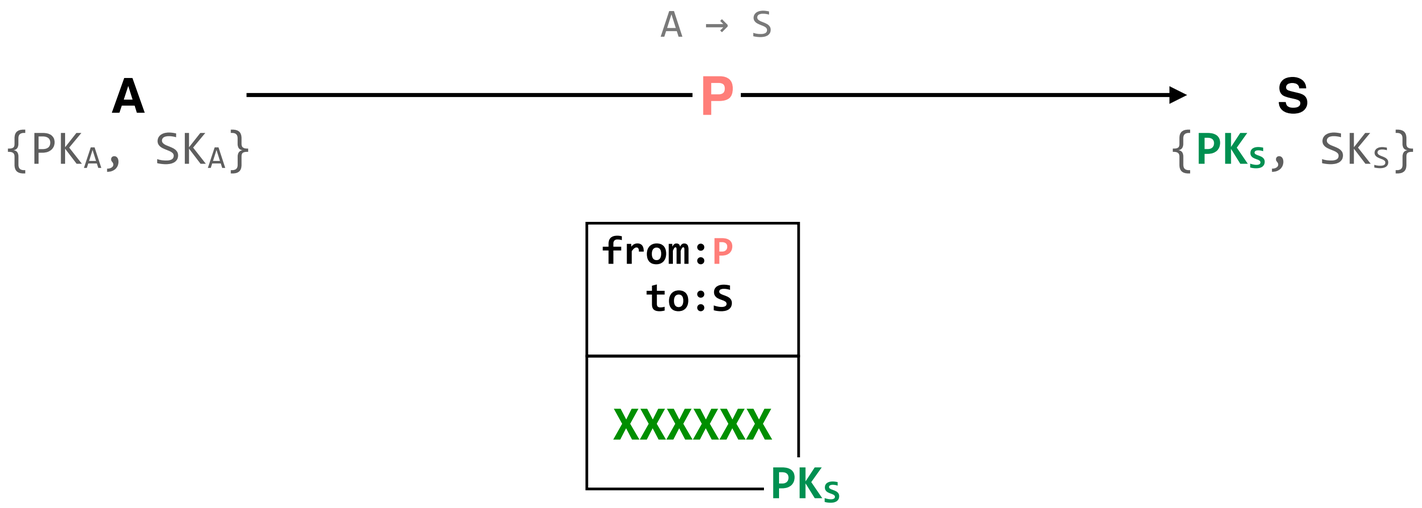
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

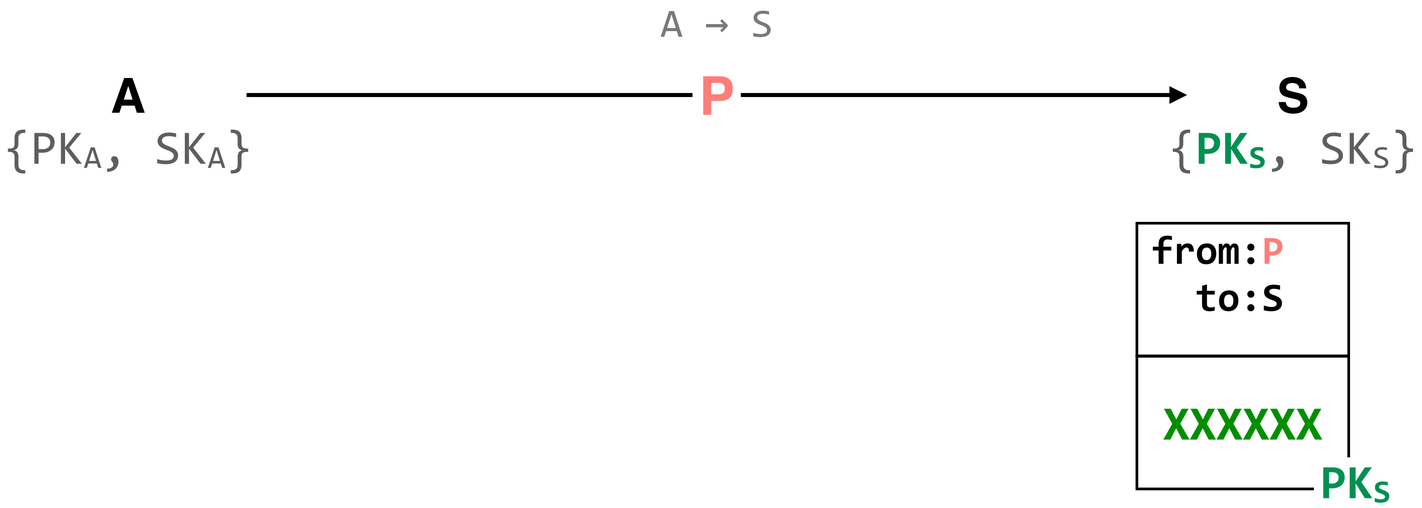
no packet should say "from: A; to: S"

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

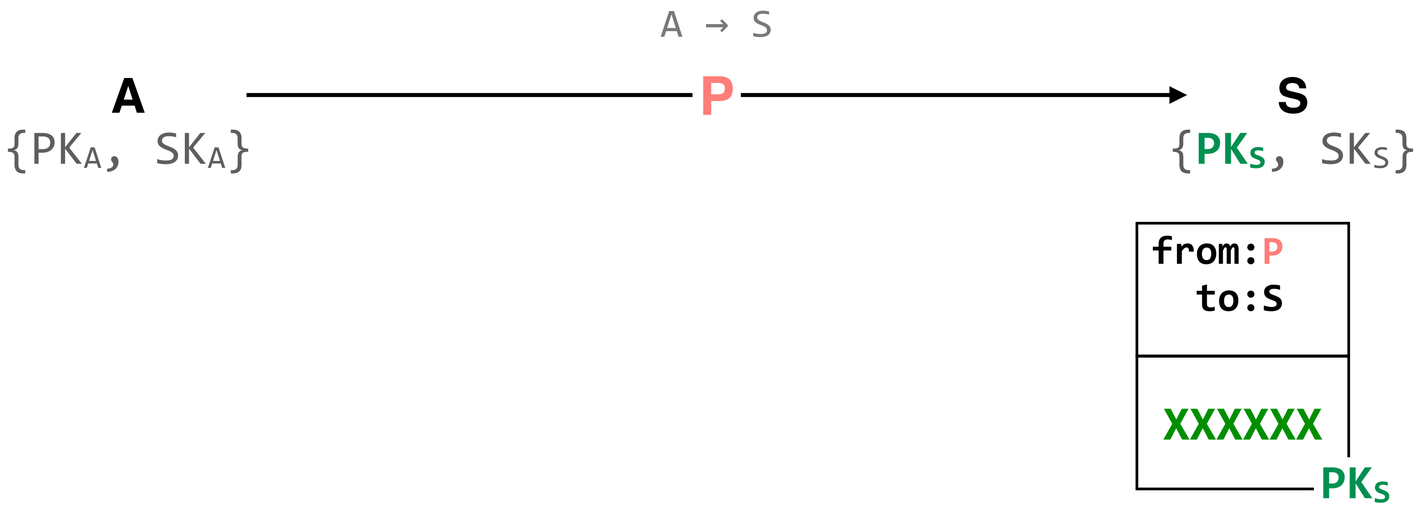
👍 no packet should say "from: A; to: S"

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.

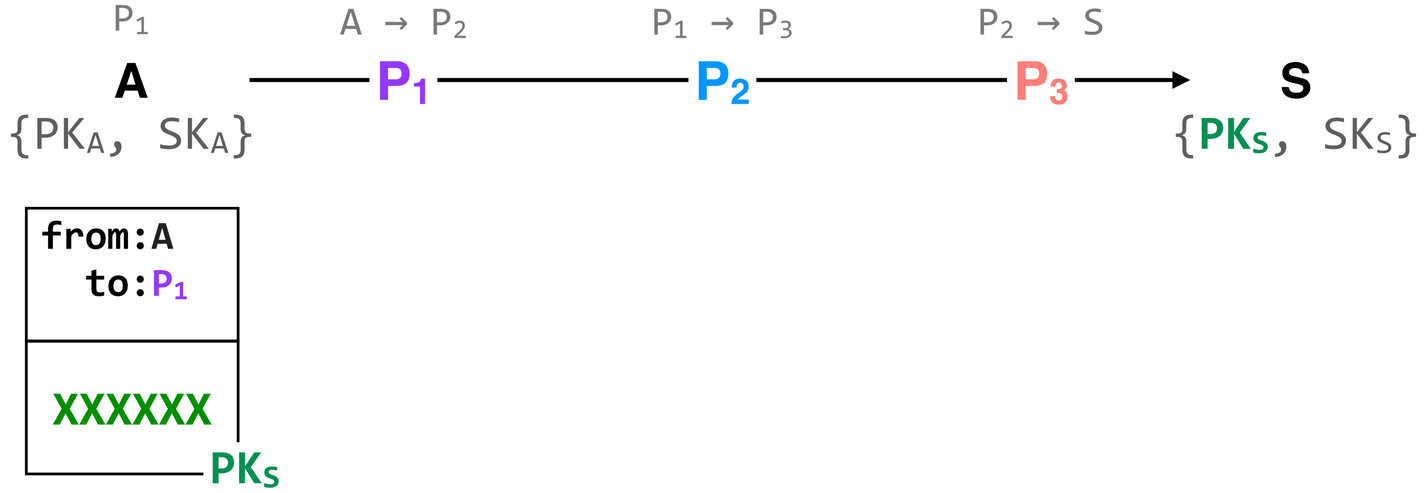
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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



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

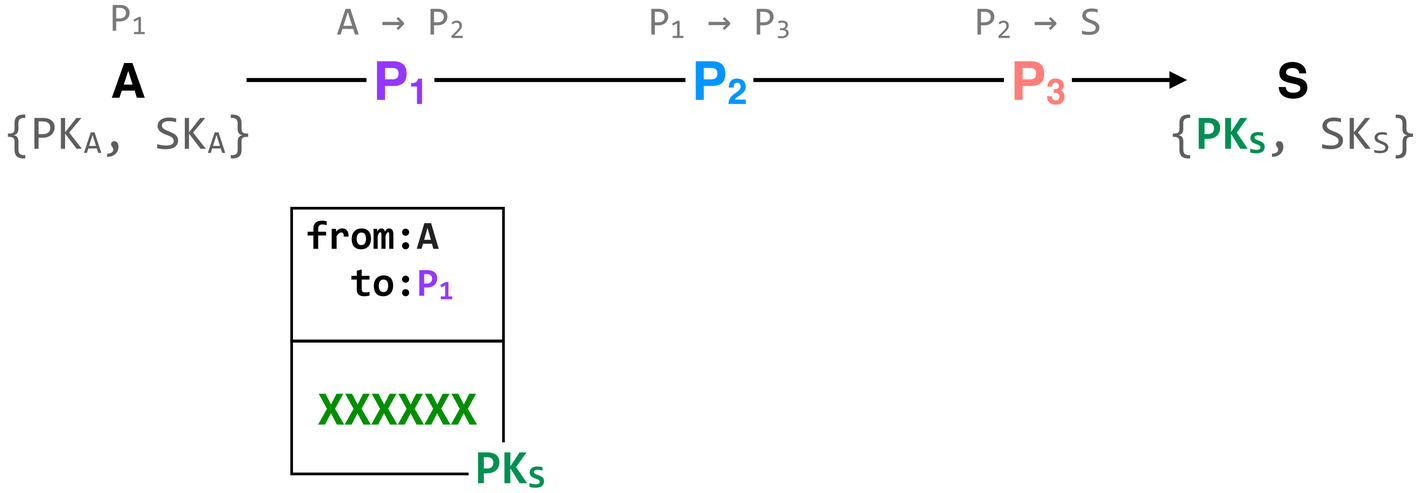
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

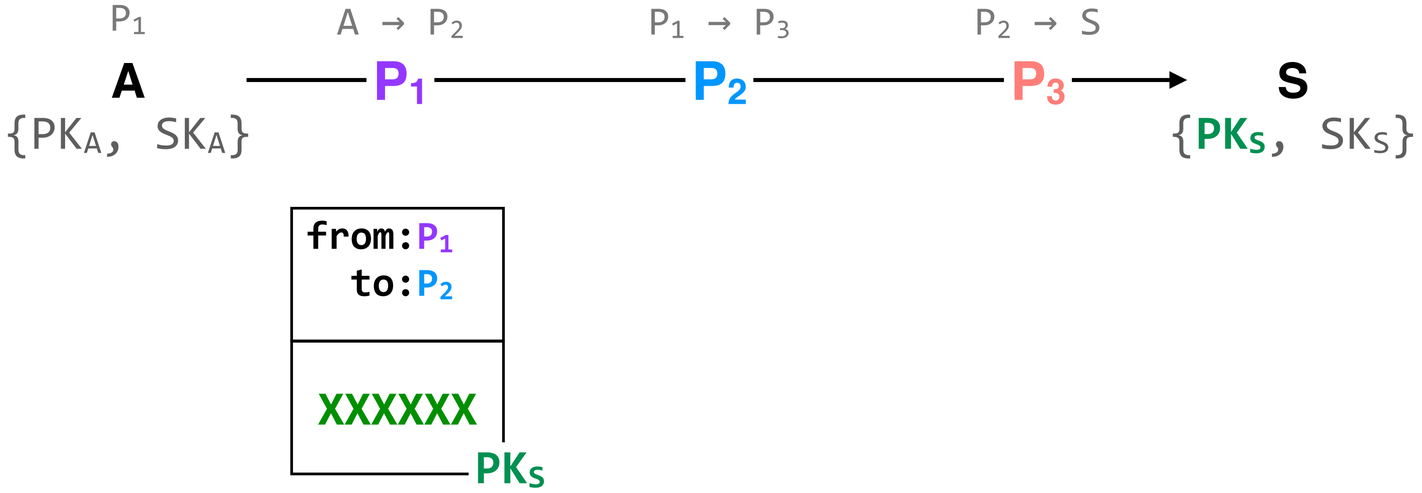
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

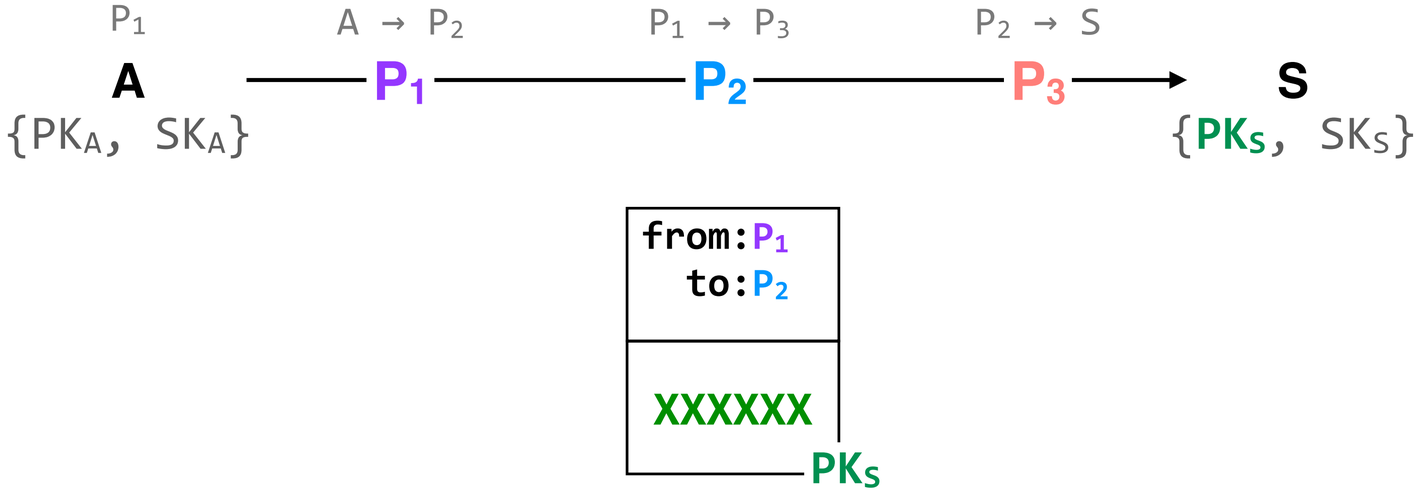
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

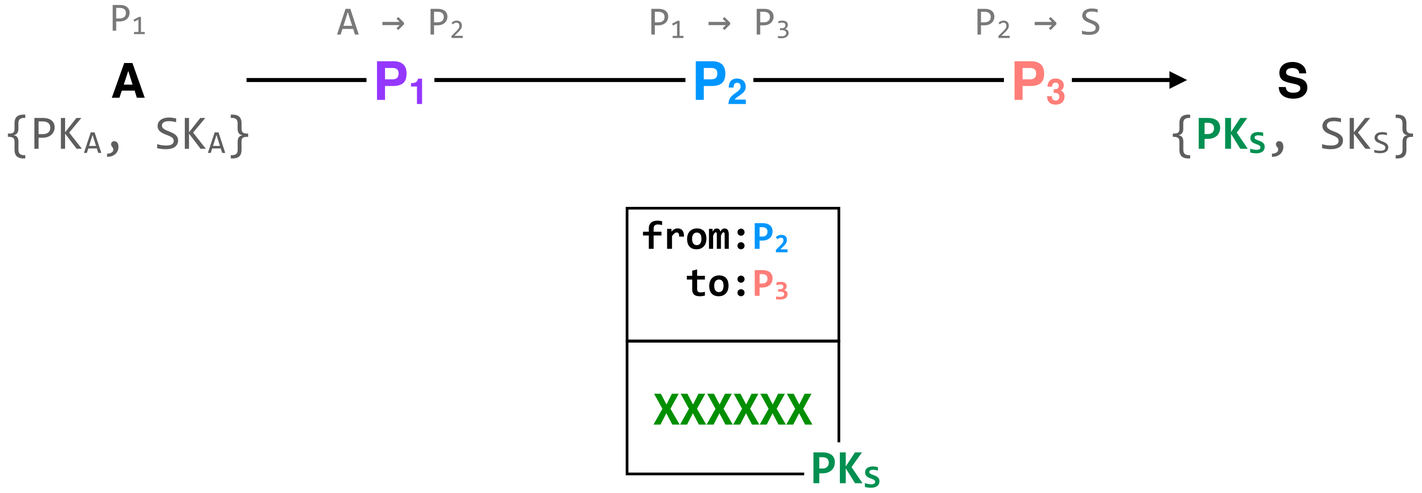
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

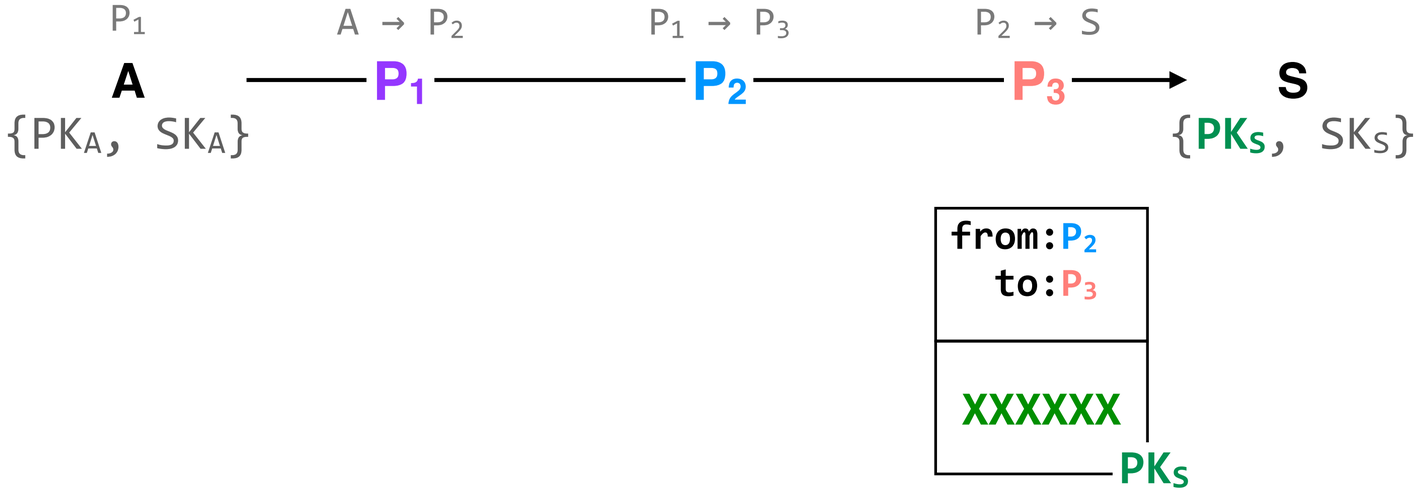
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

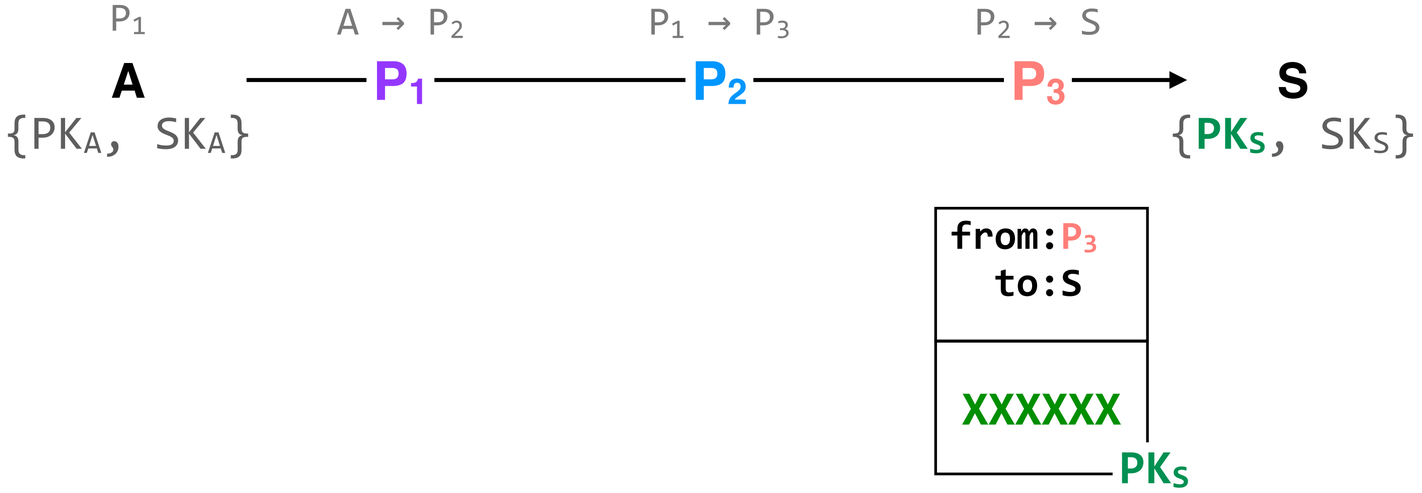
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

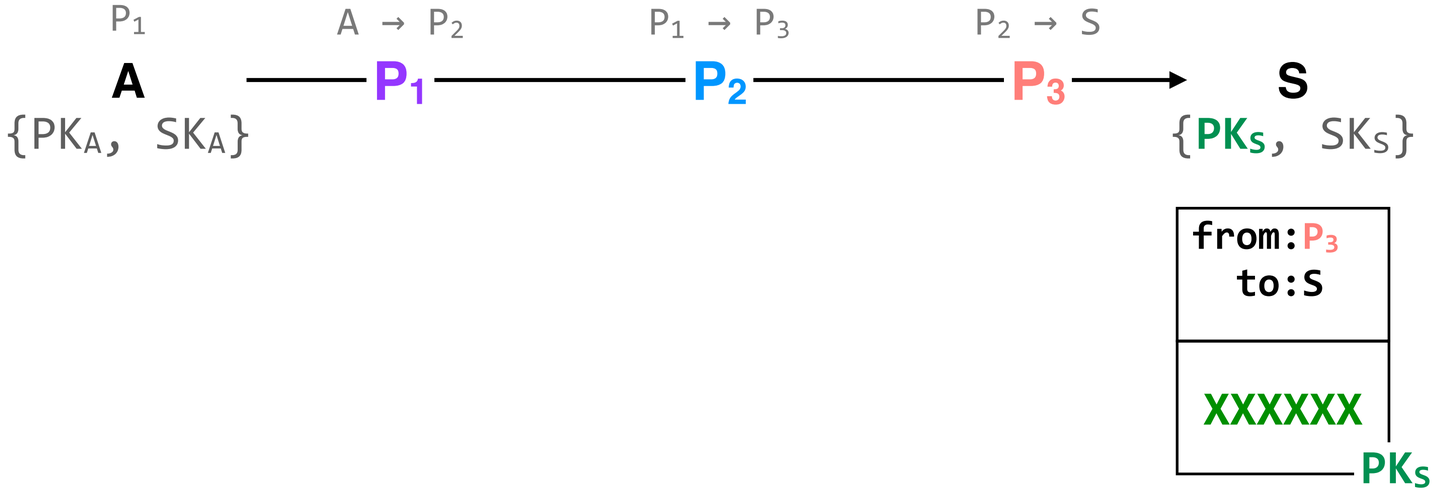
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

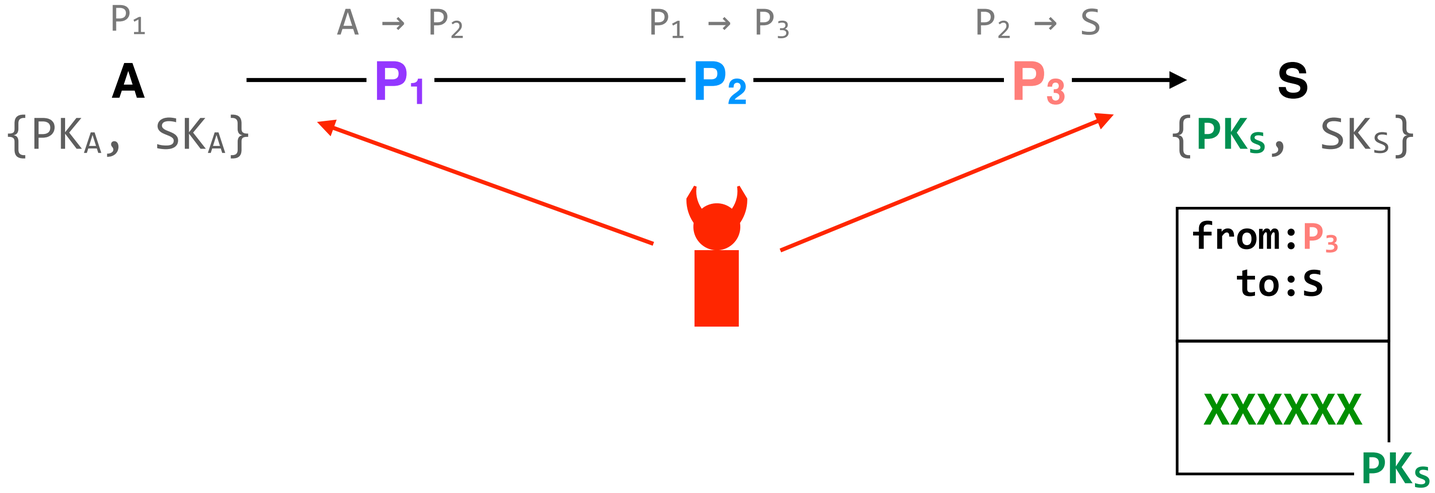
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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**

even though the data is encrypted and so can't be meaningfully read, the packet data still appears the same at every point in this network

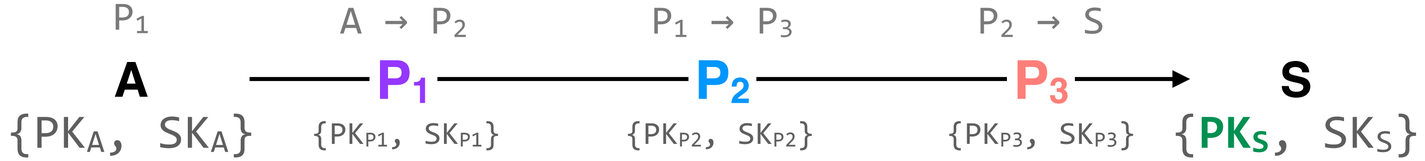
policy: provide **anonymity** (only the client should know that they're communicating with the server)

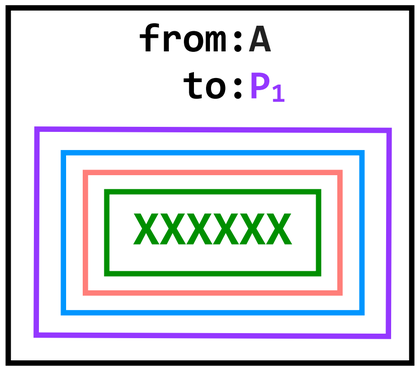
threat model: adversary is on the path between the client and the server

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

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

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





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

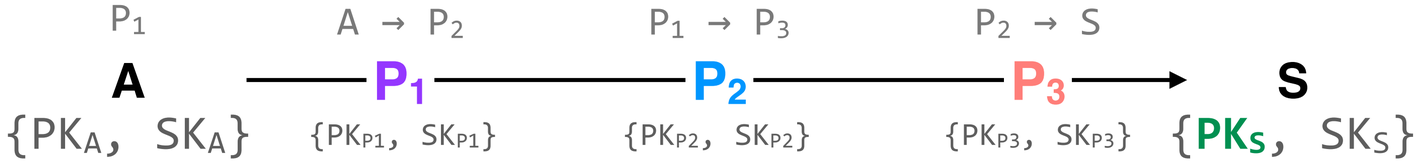
policy: provide **anonymity** (only the client should know that they're communicating with the server)

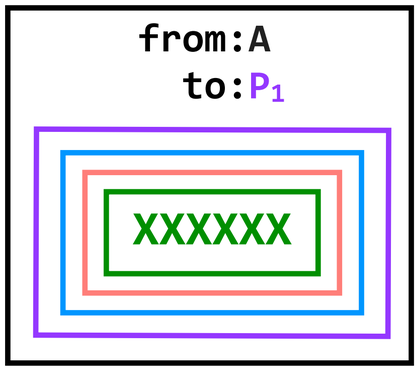
threat model: adversary is on the path between the client and the server

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

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

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





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

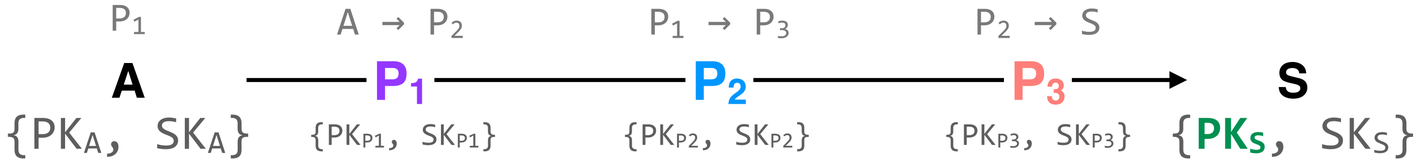
policy: provide **anonymity** (only the client should know that they're communicating with the server)

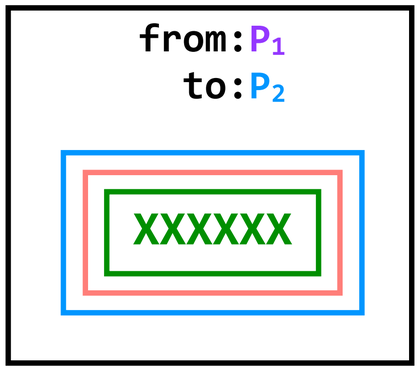
threat model: adversary is on the path between the client and the server

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

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

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





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

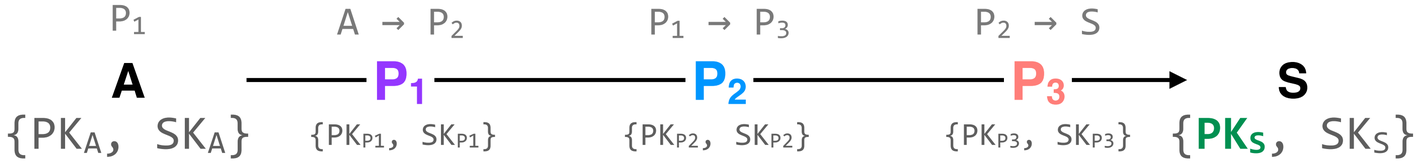
policy: provide **anonymity** (only the client should know that they're communicating with the server)

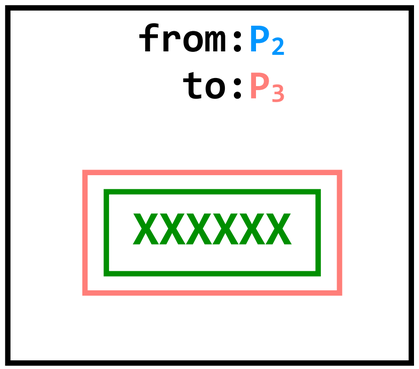
threat model: adversary is on the path between the client and the server

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

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

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





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

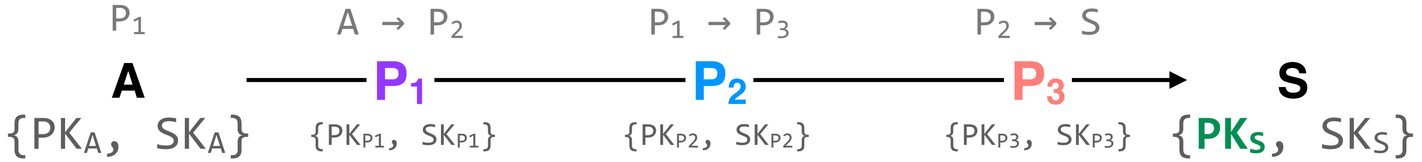
policy: provide **anonymity** (only the client should know that they're communicating with the server)

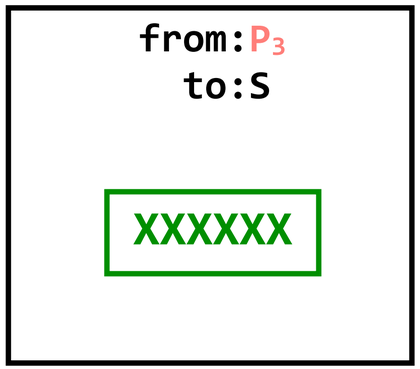
threat model: adversary is on the path between the client and the server

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

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

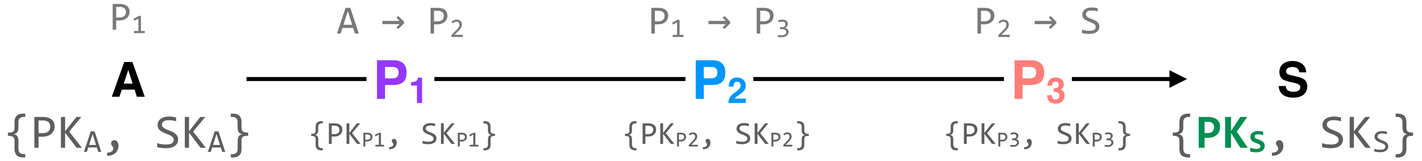
policy: provide **anonymity** (only the client should know that they're communicating with the server)

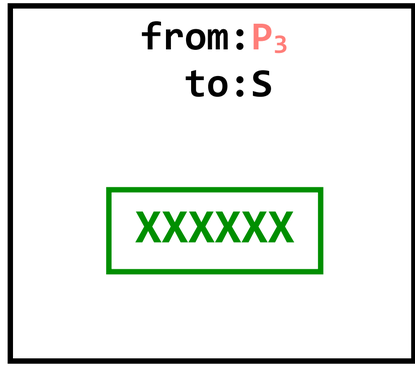
threat model: adversary is on the path between the client and the server

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

onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

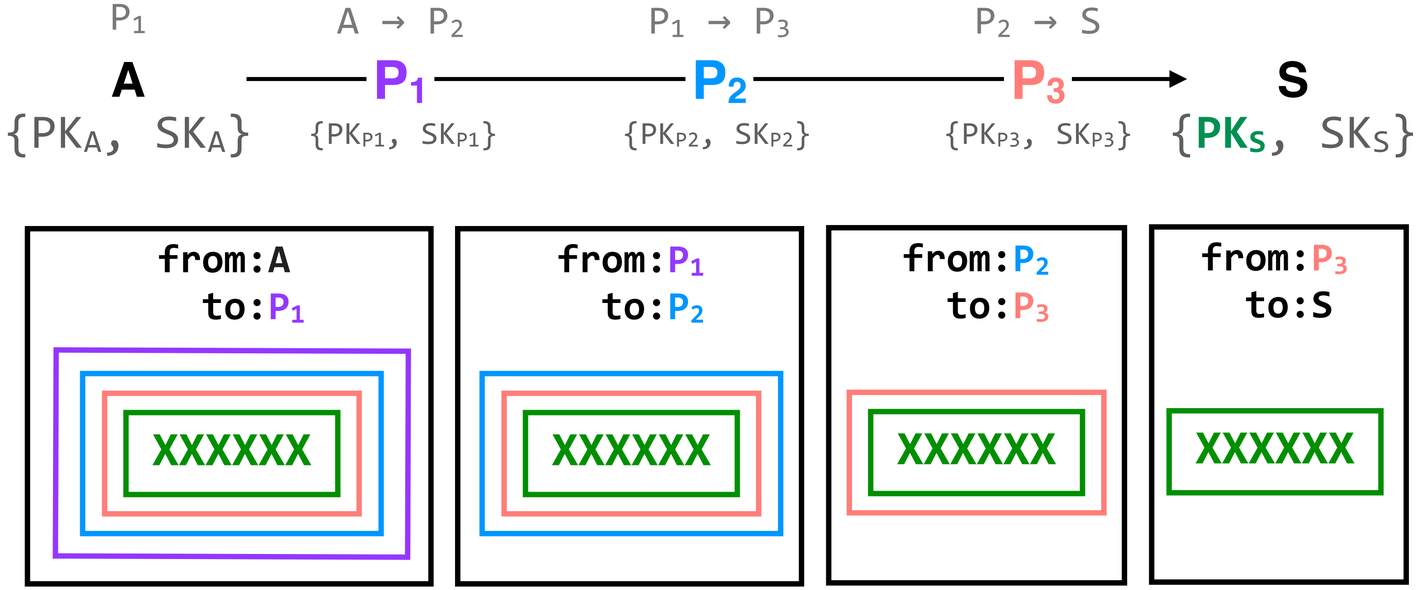
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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



onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

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

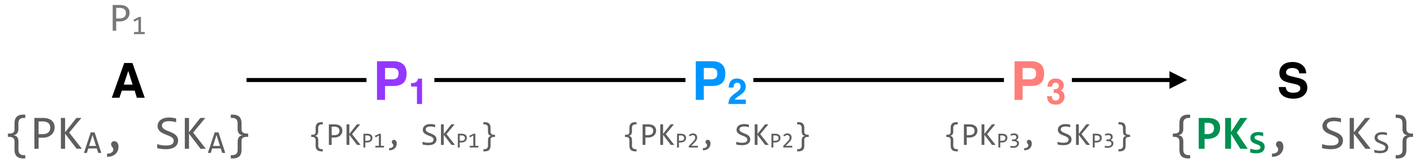
policy: provide **anonymity** (only the client should know that they're communicating with the server)

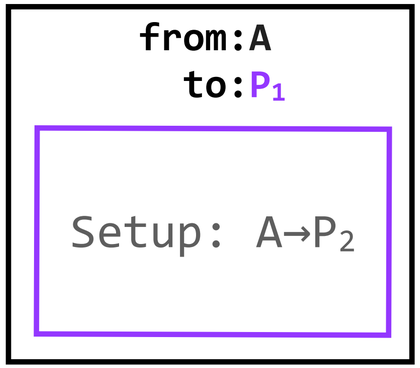
threat model: adversary is on the path between the client and the server

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

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$





A uses the same strategy to set up the circuit

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

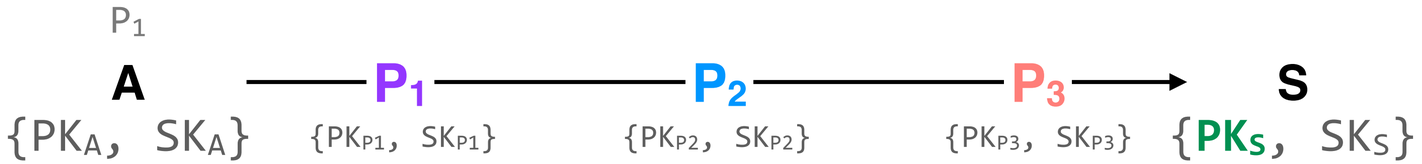
policy: provide **anonymity** (only the client should know that they're communicating with the server)

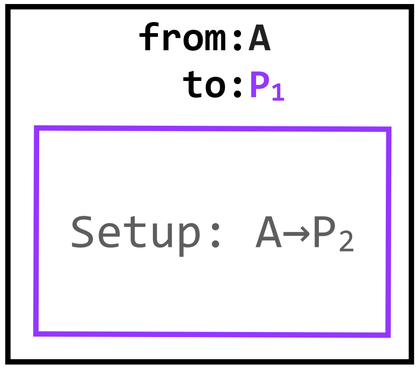
threat model: adversary is on the path between the client and the server

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

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

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





A uses the same strategy to set up the circuit

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

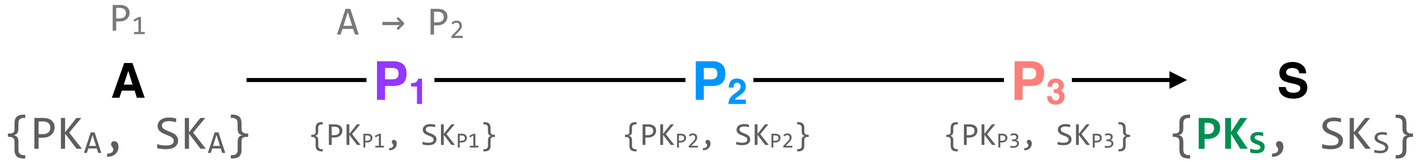
policy: provide **anonymity** (only the client should know that they're communicating with the server)

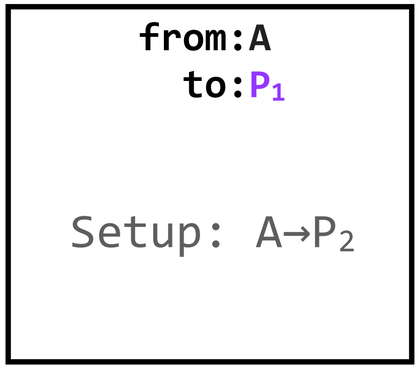
threat model: adversary is on the path between the client and the server

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

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

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





A uses the same strategy to set up the circuit

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

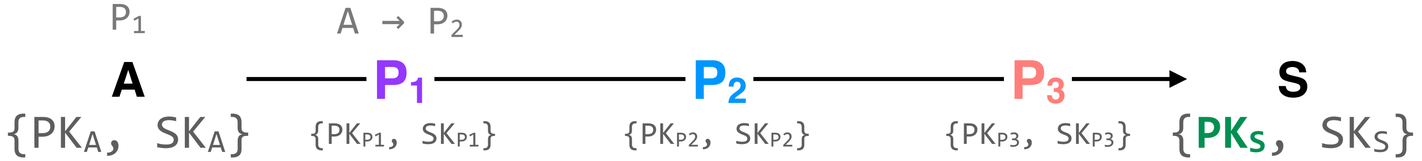
policy: provide **anonymity** (only the client should know that they're communicating with the server)

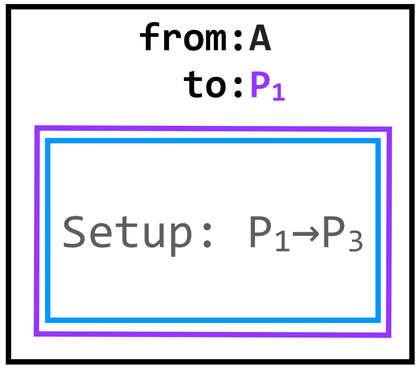
threat model: adversary is on the path between the client and the server

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

$encrypt(PK_x, m) = c$

$decrypt(SK_x, c) = m$





A uses the same strategy to set up the circuit

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

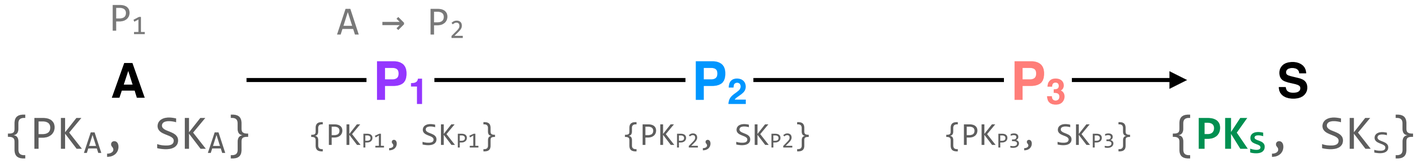
policy: provide **anonymity** (only the client should know that they're communicating with the server)

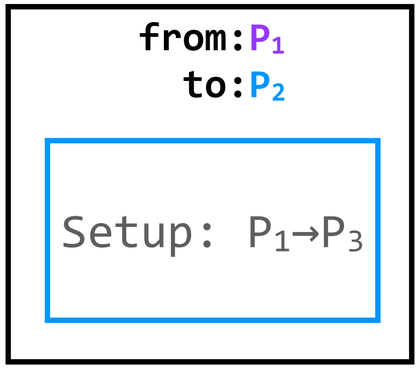
threat model: adversary is on the path between the client and the server

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

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

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





A uses the same strategy to set up the circuit

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

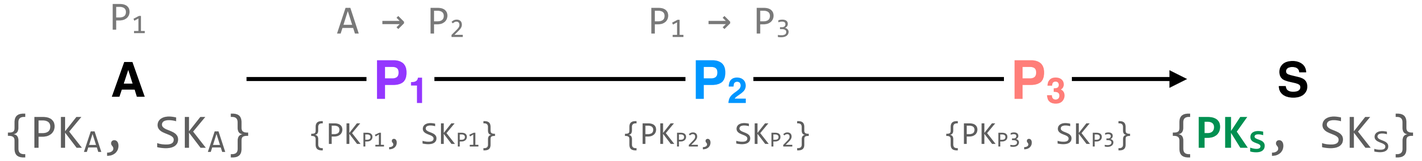
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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




```
from:P1
  to:P2

Setup: P1→P3
```

A uses the same strategy to set up the circuit

things to avoid

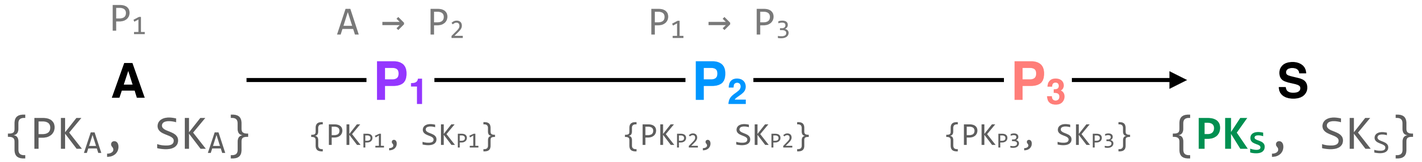
- no packet should say "from: A; to: S"
- no entity in the network should receive a packet from A and send it directly to S
- no entity in the network should keep *state* that links A to S
- data should not appear the same across packets

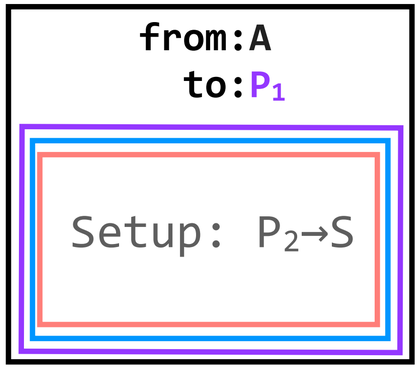
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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





A uses the same strategy to set up the circuit

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

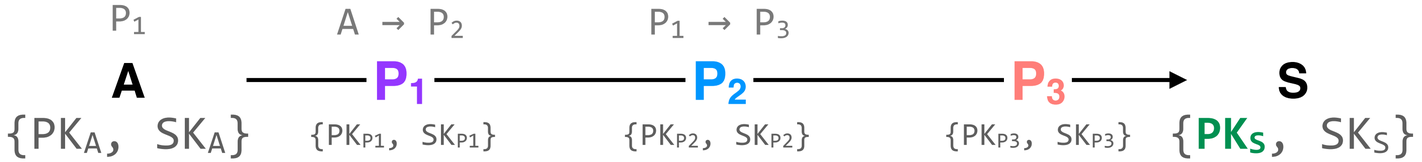
policy: provide **anonymity** (only the client should know that they're communicating with the server)

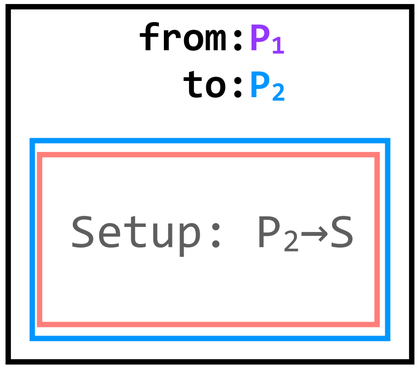
threat model: adversary is on the path between the client and the server

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

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

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





A uses the same strategy to set up the circuit

things to avoid

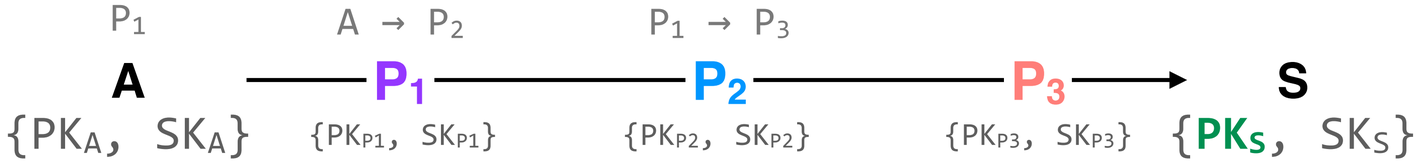
- no packet should say "from: A; to: S"
- no entity in the network should receive a packet from A and send it directly to S
- no entity in the network should keep *state* that links A to S
- data should not appear the same across packets

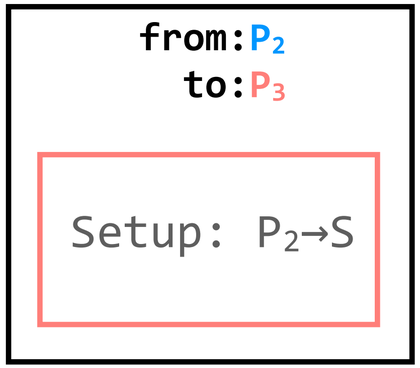
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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





A uses the same strategy to set up the circuit

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

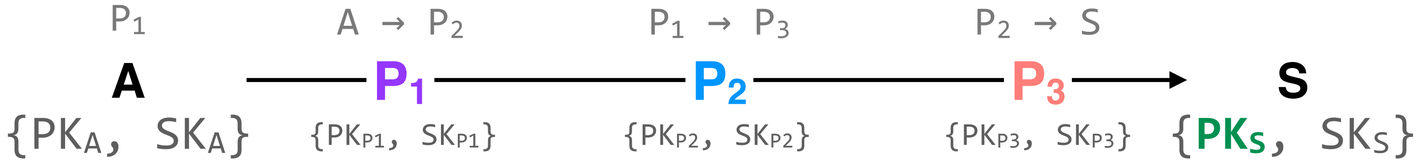
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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




```
from:P2
  to:P3

Setup: P2→S
```

A uses the same strategy to set up the circuit

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

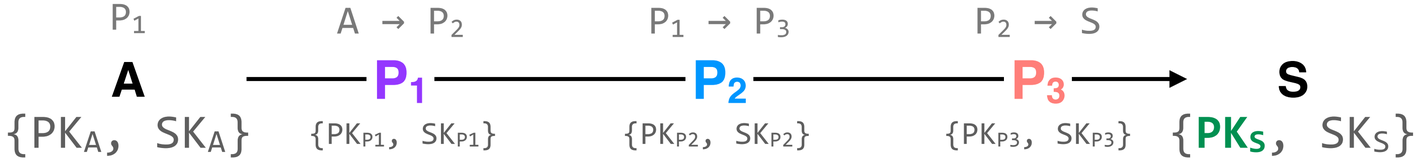
policy: provide **anonymity** (only the client should know that they're communicating with the server)

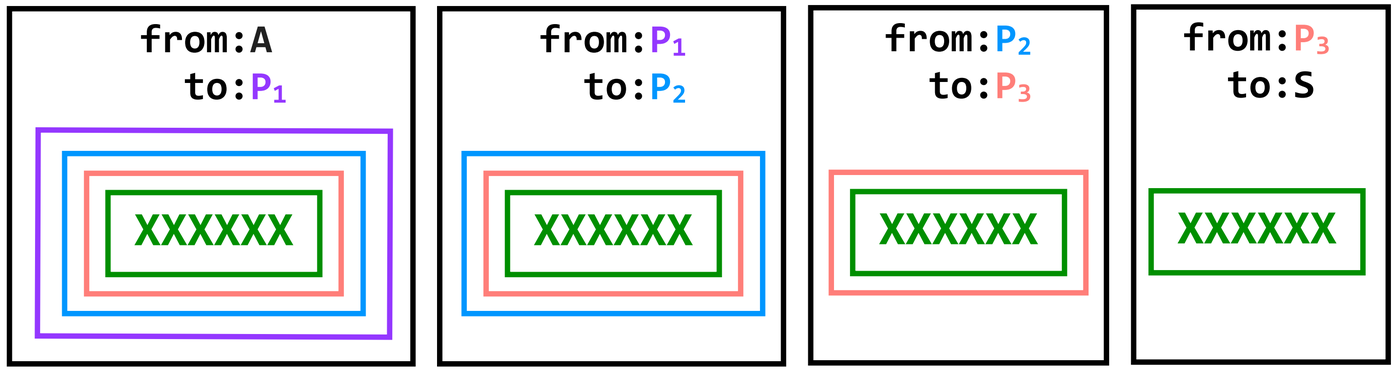
threat model: adversary is on the path between the client and the server

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

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

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





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

in practice, **tor** uses public-key cryptography to securely exchange **symmetric keys** between A and each node in the circuit, and the layers of encryption use those symmetric keys; this is what allow traffic to travel in both directions

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

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

Am I totally anonymous if I use Tor?

Generally it is impossible to have perfect anonymity, even with Tor. Though there are some things you can practice to improve your anonymity while using Tor and offline.

Use Tor Browser and software specifically configured for Tor

Tor does not protect all of your computer's Internet traffic when you run it. Tor only protects applications that are properly configured to send their Internet traffic through Tor.

Web browsing:

- Safe: Tor Browser
- Unsafe: Any other browser configured to use Tor as a proxy

File sharing:

- Safe: OnionShare
- Unsafe: BitTorrent over Tor

https://support.torproject.org/faq/staying-anonymous/

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

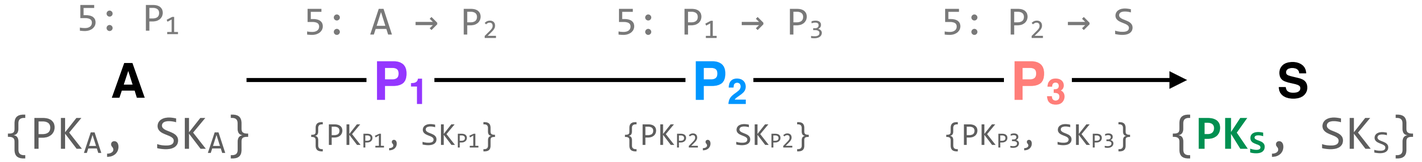
$$encrypt(PK_x, m) = c$$

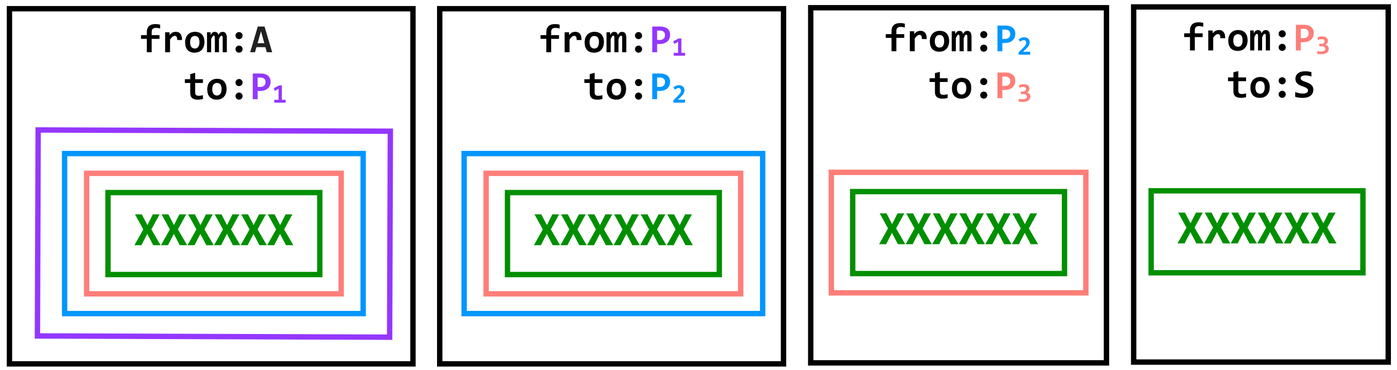
$$decrypt(SK_x, c) = m$$

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

5 is the (random) "circuit ID". a proxy may be involved in more than one circuit, so it needs a way to differentiate





onion routing adds layers of encryption that nodes on the path can strip off as the packet traverses the network

in practice, **tor** uses public-key cryptography to securely exchange **symmetric keys** between A and each node in the circuit, and the layers of encryption use those symmetric keys; this is what allow traffic to travel in both directions

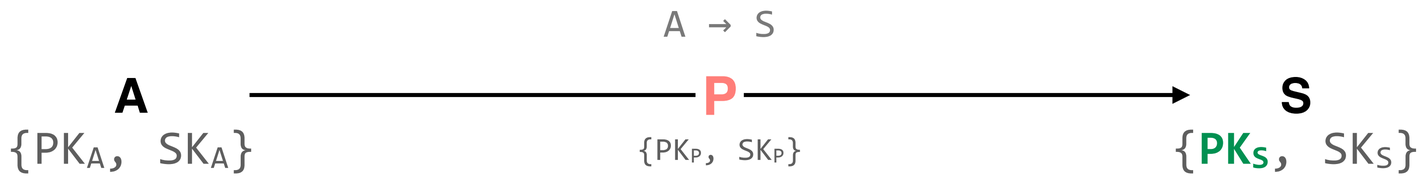
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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



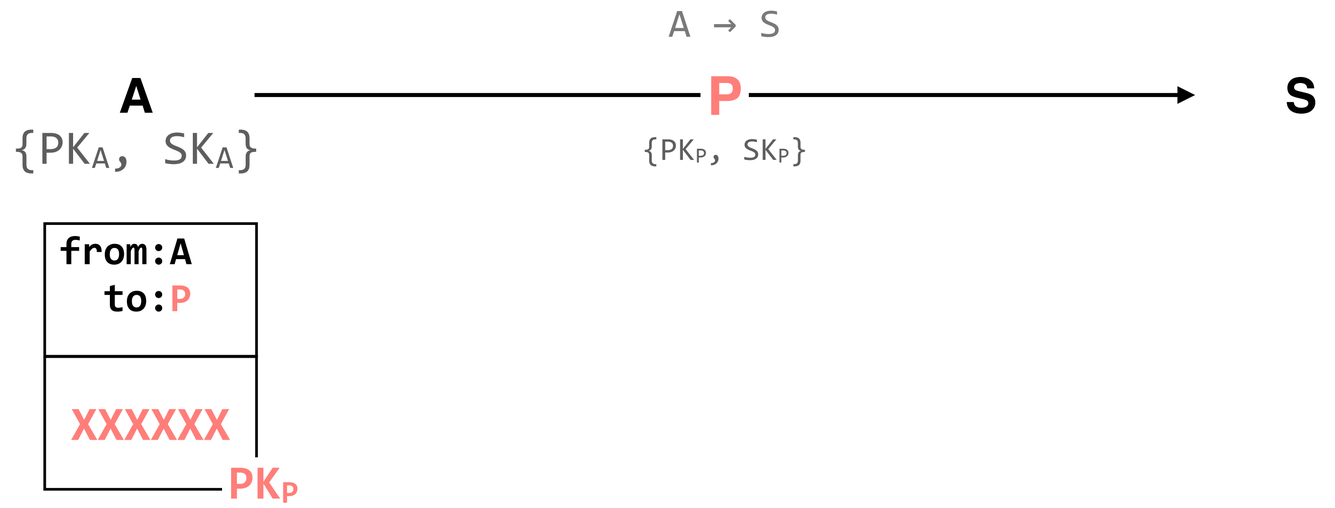
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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



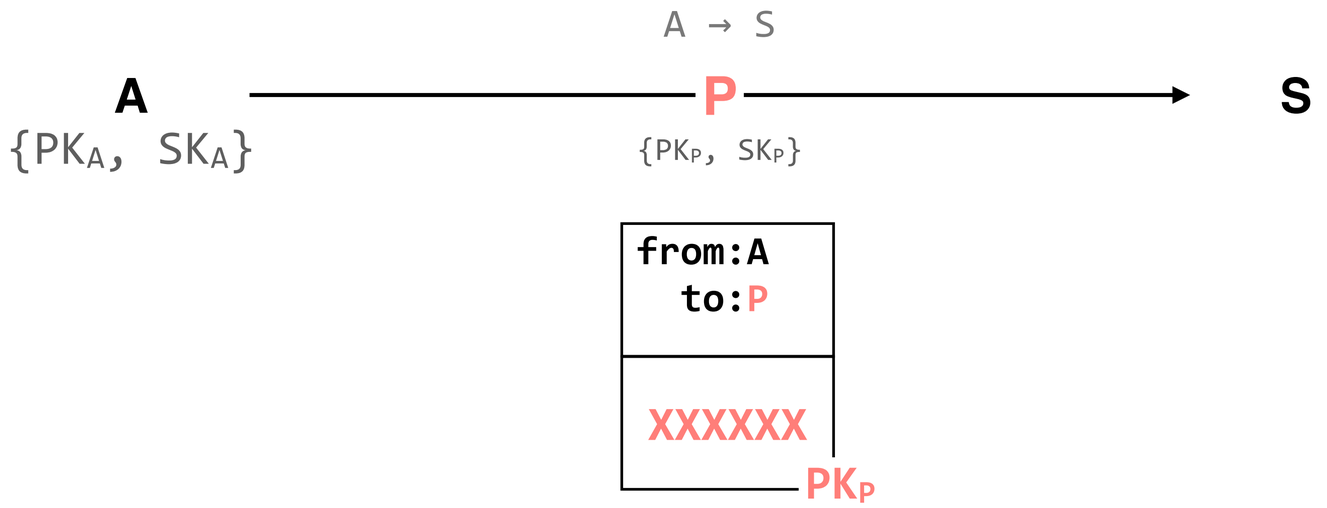
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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`



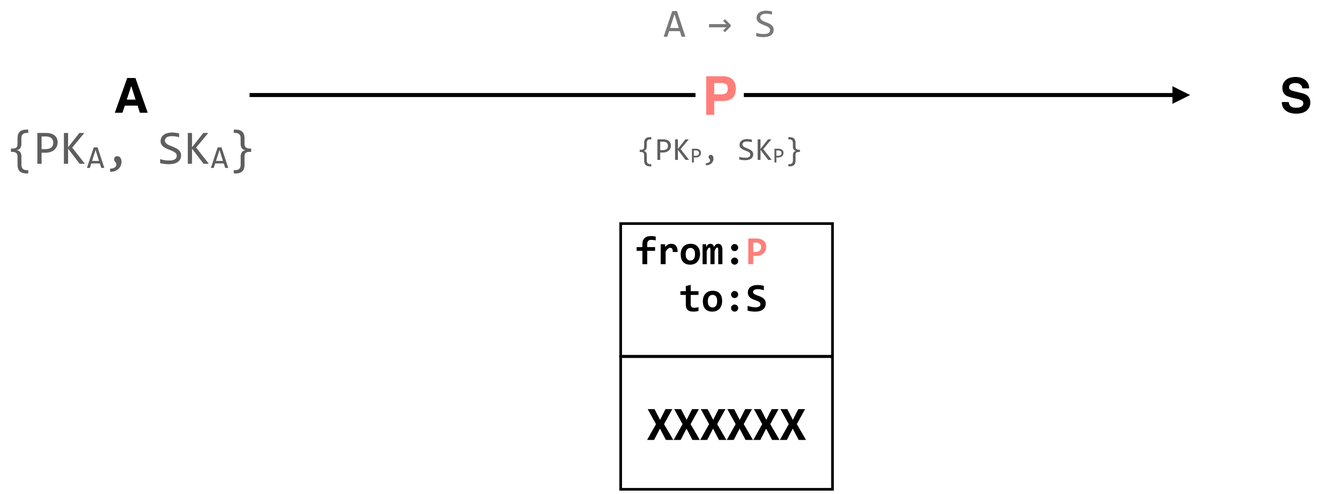
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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`



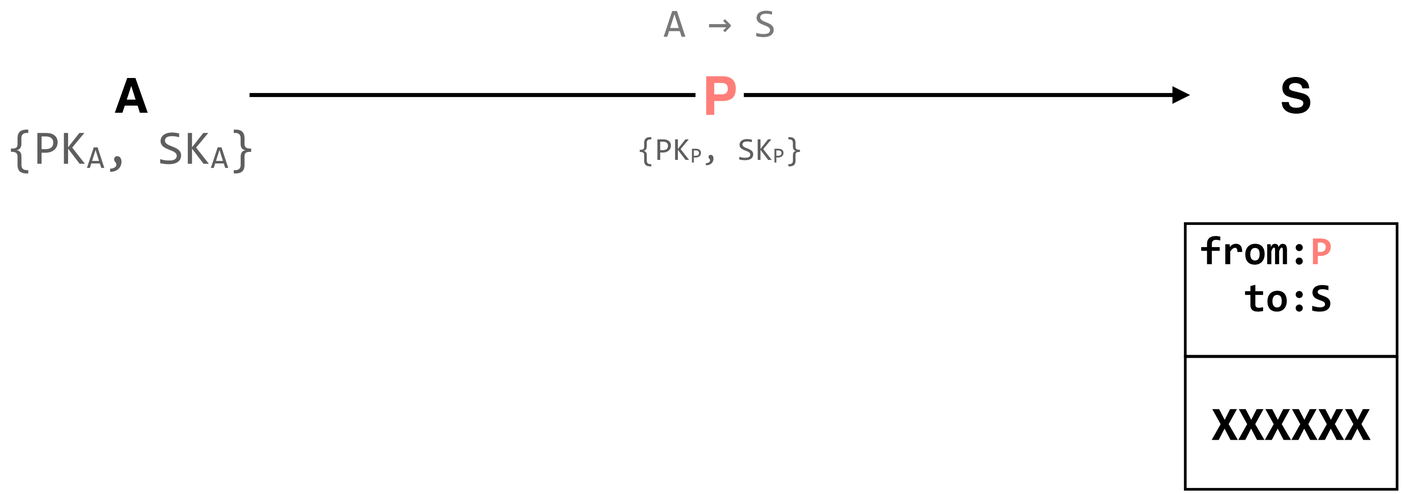
policy: provide **anonymity** (only the client should know that they're communicating with the server)

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

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

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



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

much like when we discussed certificate authorities, there are interesting questions about who should run tor. how do we trust that the relay nodes are behaving as they should?

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

as system designers, it's important to think about what traffic you're sending over the network to clients, and whether that traffic can be sent in a more secure way (and what the trade-offs would be)

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