6.033 Spring 2019
Lecture #25

- **Anonymity**
  - **Tor**

Bitcoin and Tor

two technologies that deal, either directly or somewhat-tangentially, with **anonymity**

encryption with public keys

Alice's key pair

$\{PK_A, SK_A\}$

Bob's key pair

$\{PK_B, SK_B\}$

Alice encrypts messages to Bob with his public key;
Bob decrypts the messages with his secret key

$$encrypt(PK_B, m) = c \longrightarrow decrypt(SK_B, c) = m$$

problem: header links `Alice` to `S`

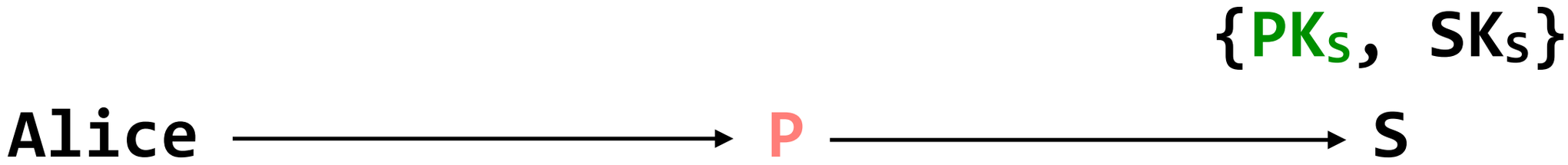
Tor's goal

provide **anonymity** — only Alice should know that she is communicating with the server S

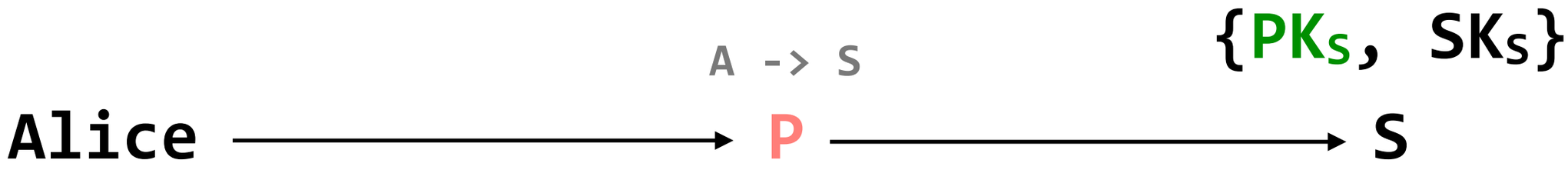
things to avoid

- no packet should say "from: Alice, to: S"

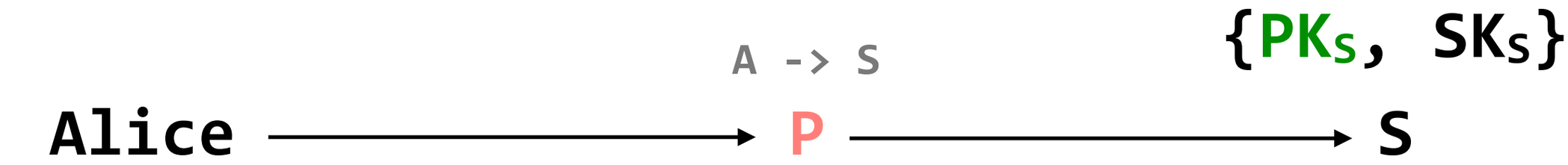
things to avoid: no packet should say “from: Alice, to: S”

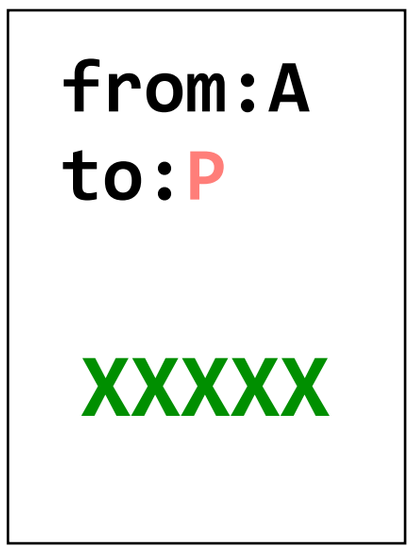


things to avoid: no packet should say “from: Alice, to: S”

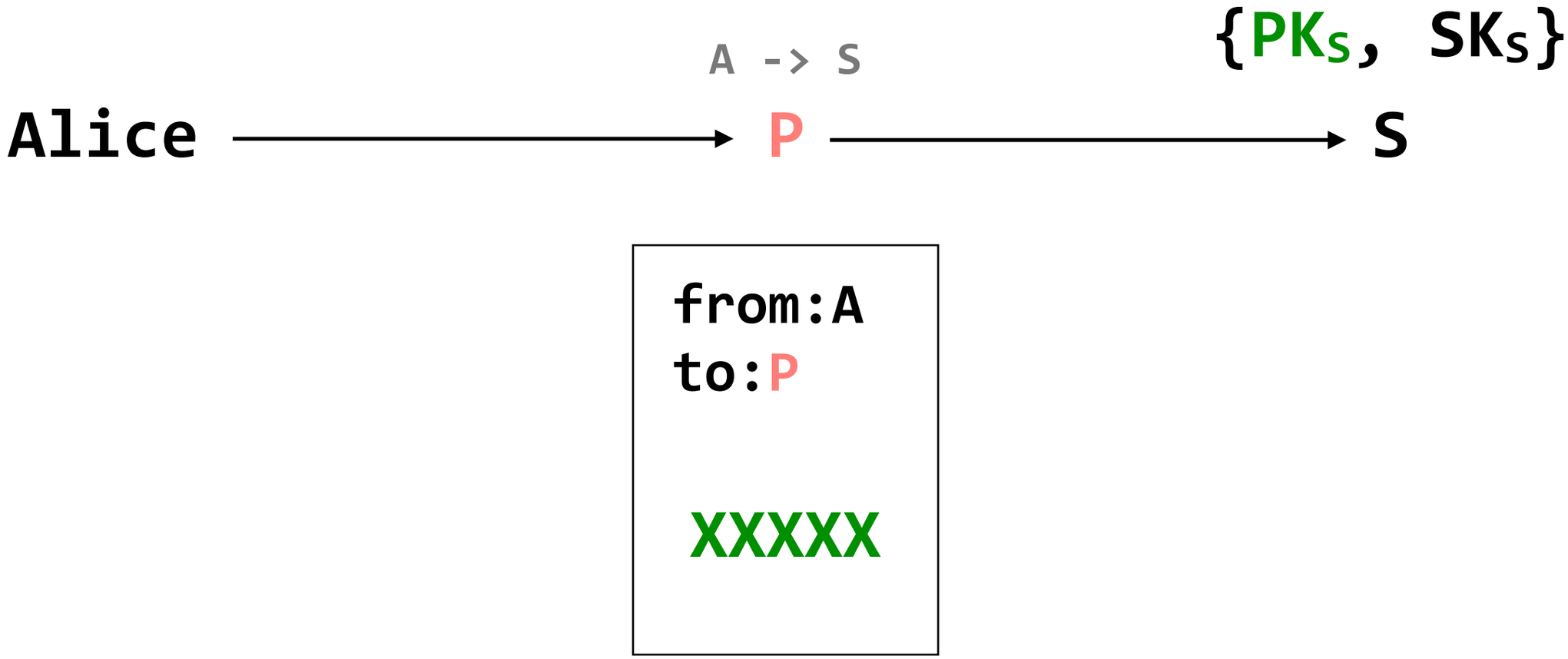


things to avoid: no packet should say “from: Alice, to: S”

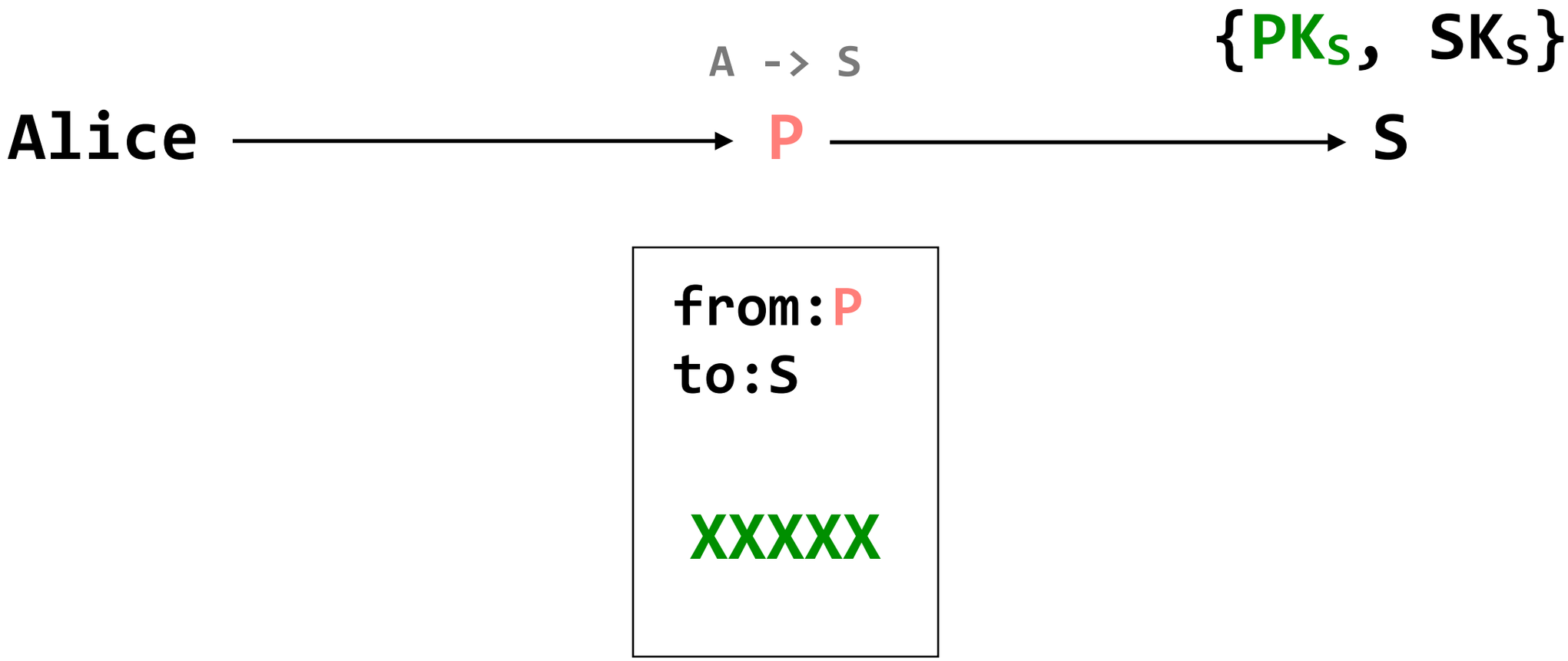




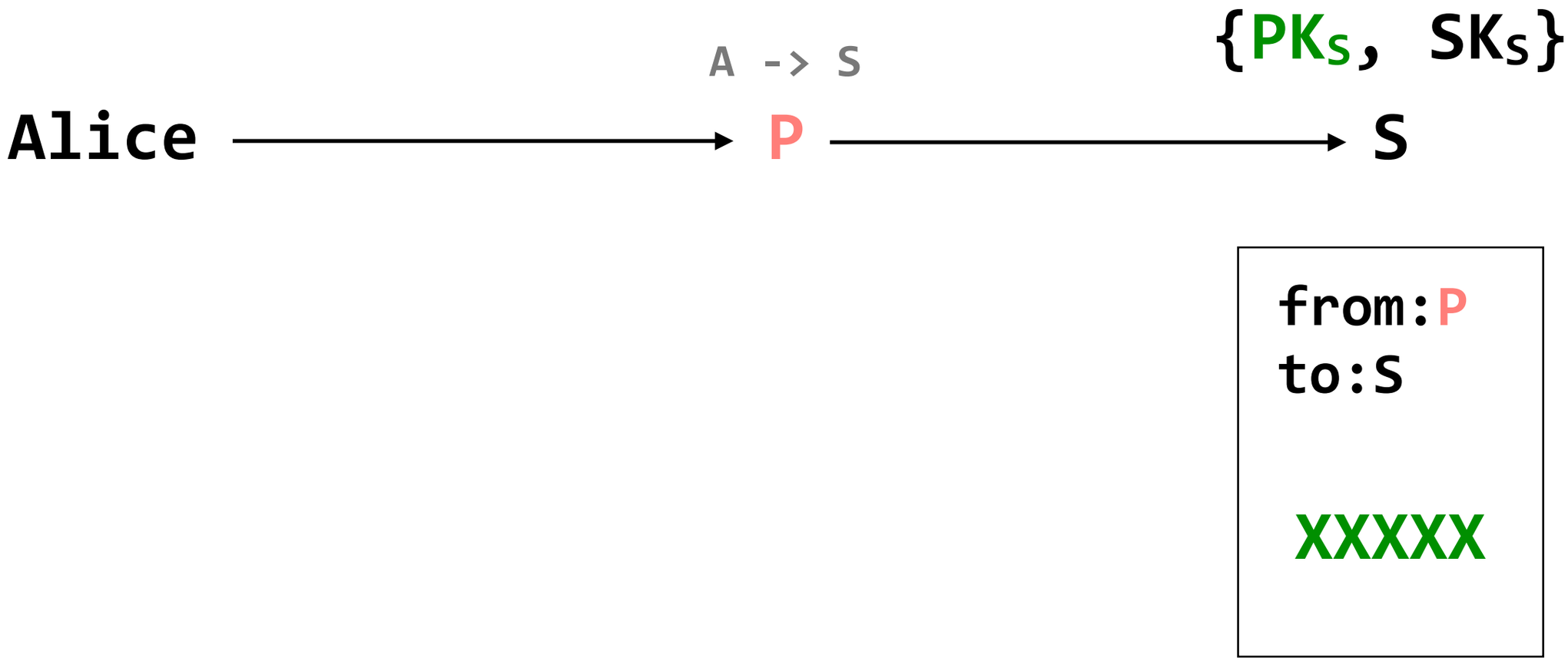
things to avoid: no packet should say “from: Alice, to: S”



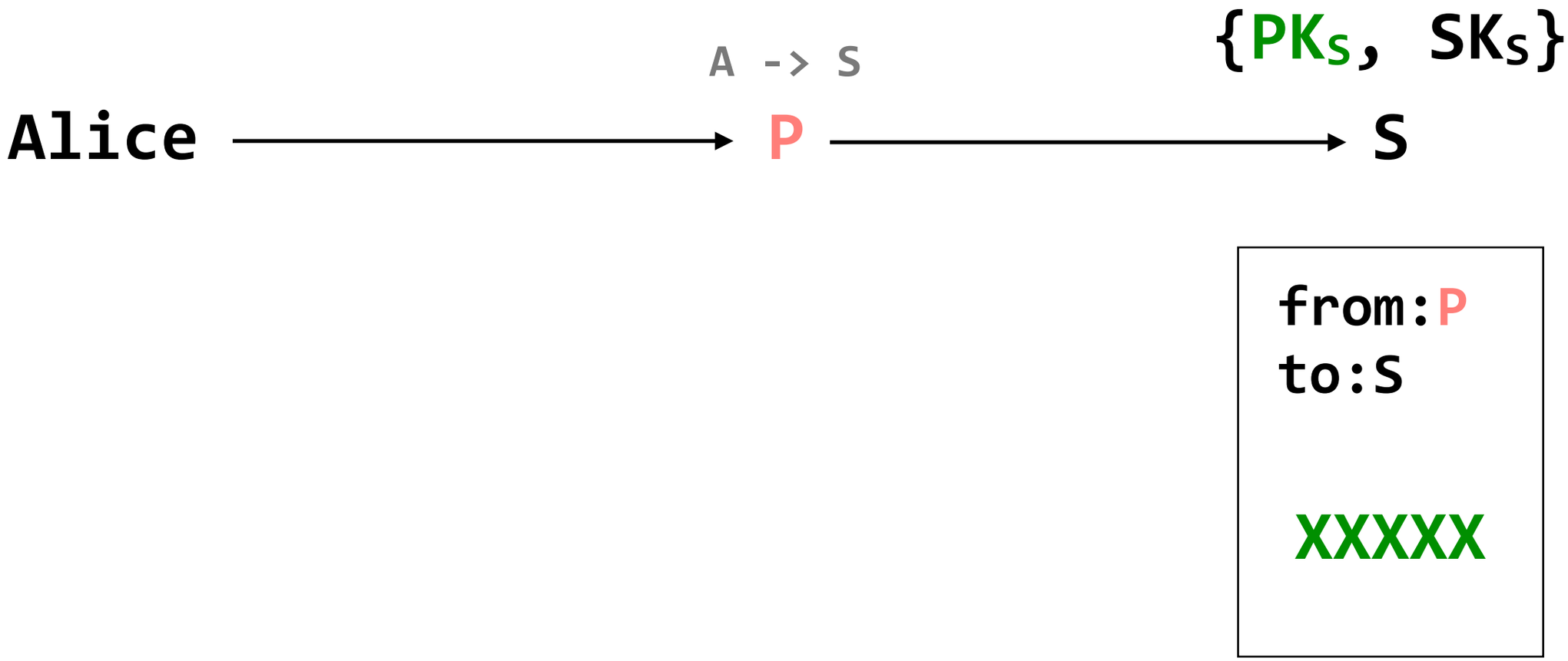
things to avoid: no packet should say “from: Alice, to: S”



things to avoid: no packet should say “from: Alice, to: S”



things to avoid: no packet should say “from: Alice, to: S”

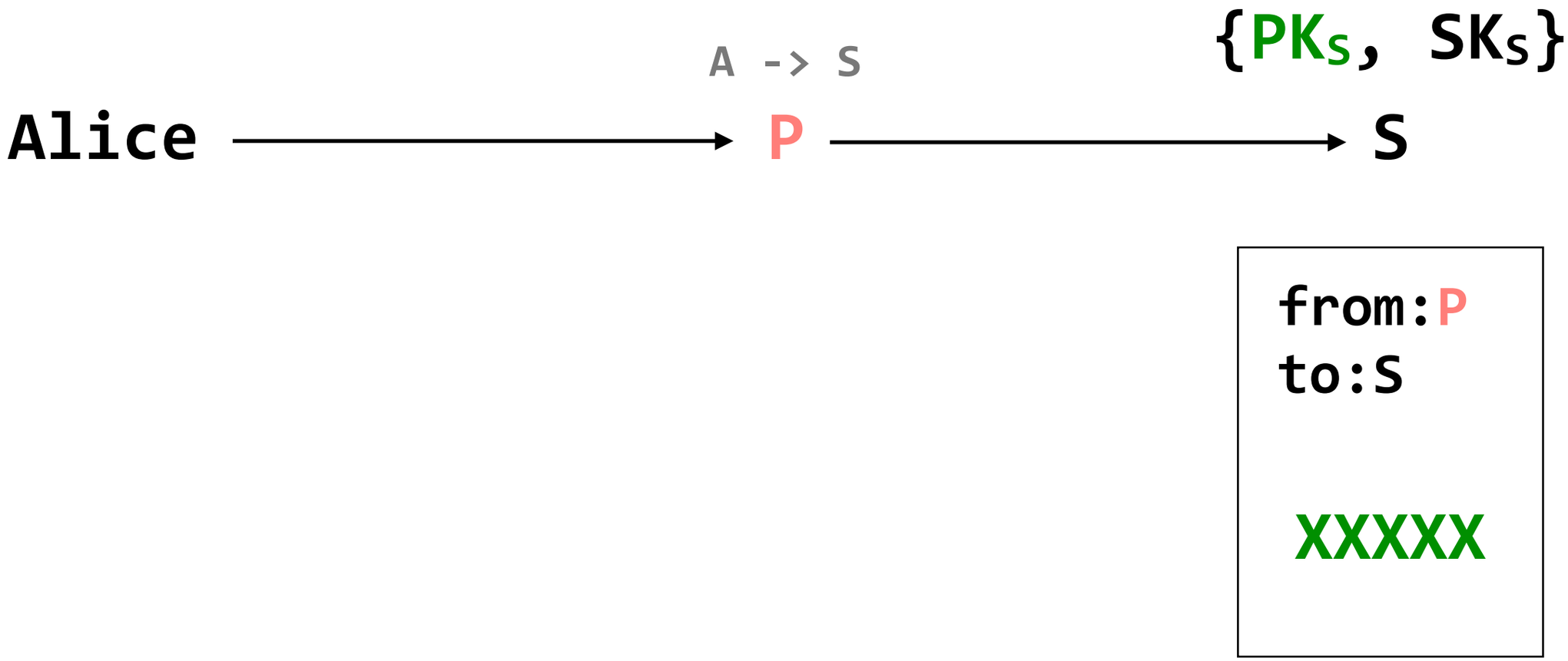


problem: `P` knows that `Alice` is communicating with `S`

things to avoid: no packet should say “from: Alice, to: S”

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

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

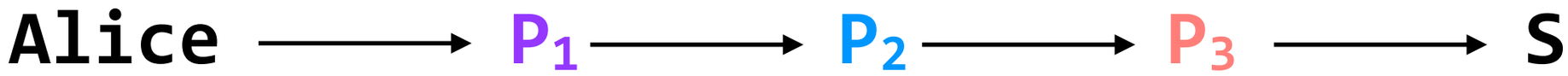


problem: P knows that `Alice` is communicating with `S`

things to avoid: no packet should say “from: Alice, to: S”

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

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



XXXXX

things to avoid: no packet should say “from: Alice, to: S”

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

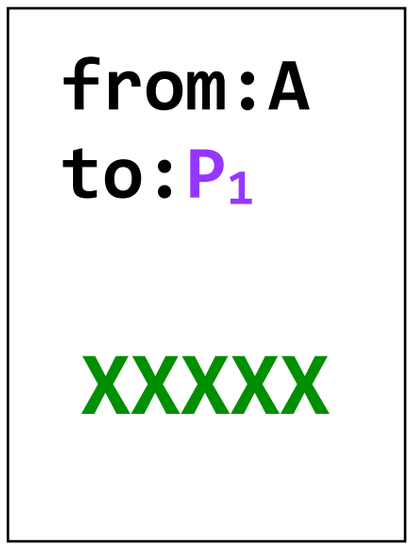
no entity in the network should keep state that links Alice to S

XXXXX

things to avoid: no packet should say “from: Alice, to: S”

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

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

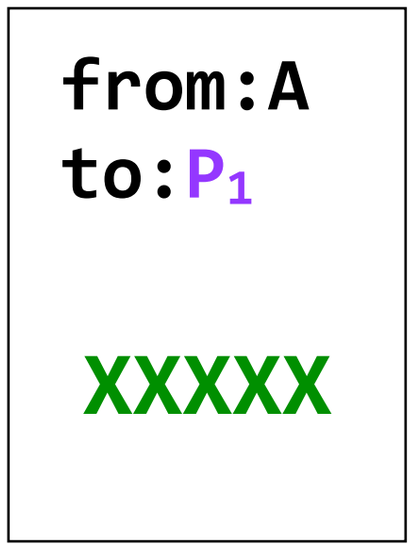


(the circuit ID would also be included and encrypted, perhaps with some shared public key)

things to avoid: no packet should say “from: Alice, to: S”

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

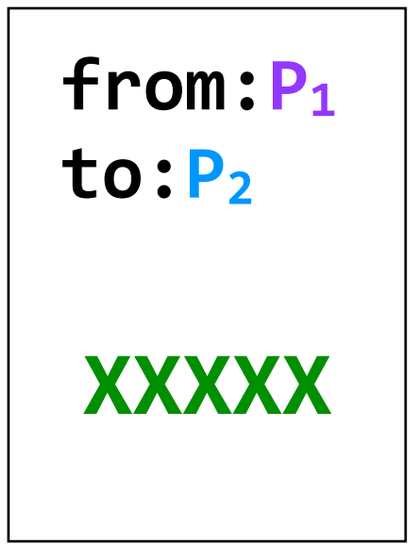
no entity in the network should keep state that links Alice to S



things to avoid: no packet should say “from: Alice, to: S”

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

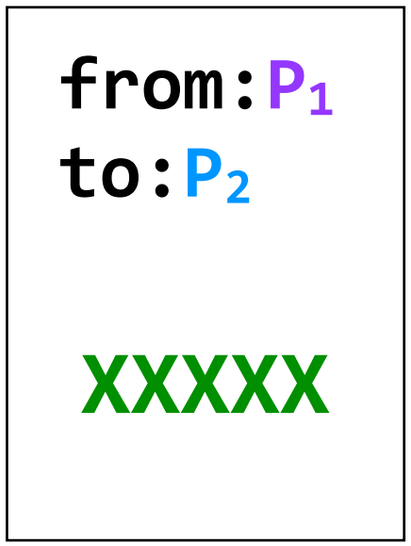
no entity in the network should keep state that links Alice to S



things to avoid: no packet should say “from: Alice, to: S”

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

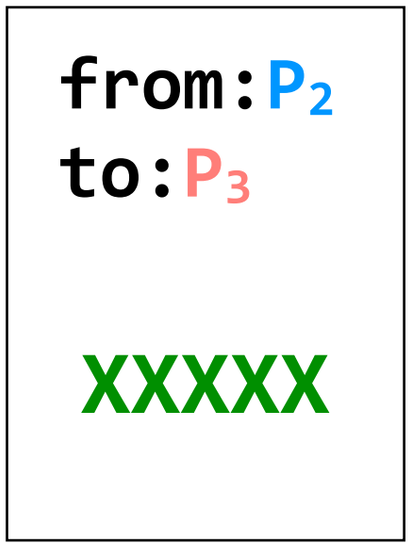
no entity in the network should keep state that links Alice to S



things to avoid: no packet should say “from: Alice, to: S”

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

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



things to avoid: no packet should say “from: Alice, to: S”

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

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

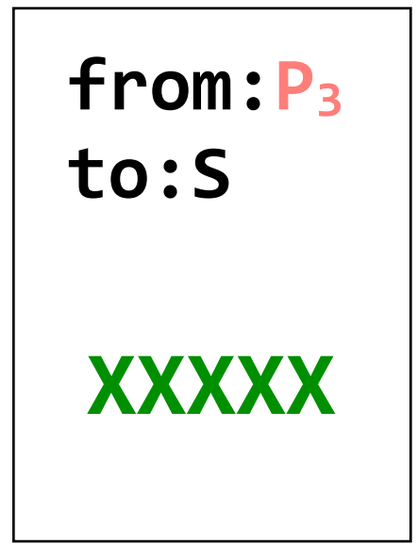
from:P_2
to:P_3

XXXXX

things to avoid: no packet should say “from: Alice, to: S”

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

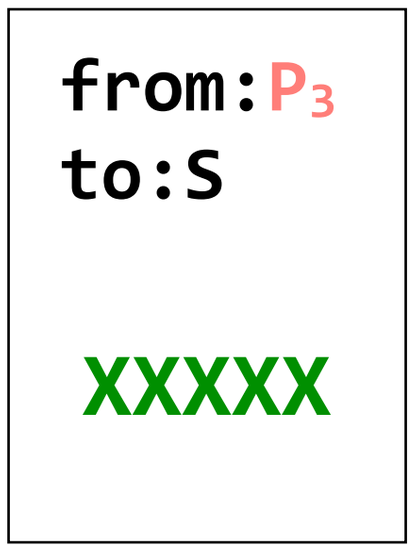
no entity in the network should keep state that links Alice to S



things to avoid: no packet should say “from: Alice, to: S”

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

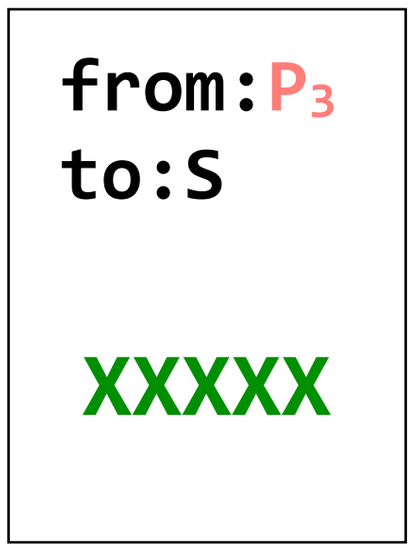
no entity in the network should keep state that links Alice to S



things to avoid: no packet should say “from: Alice, to: S”

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

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



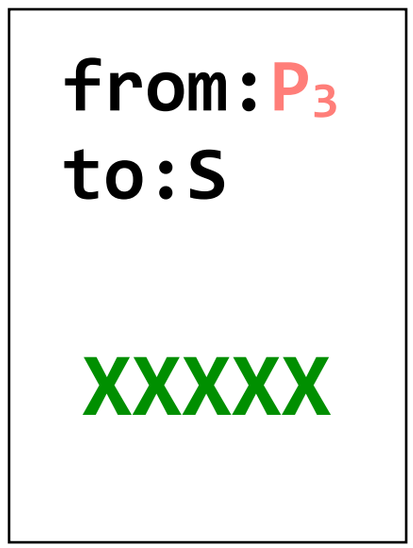
problem: an adversary with multiple vantage points can observe the same data traveling from Alice to S

things to avoid: no packet should say “from: Alice, to: S”

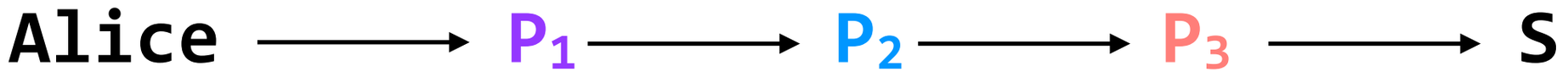
no entity in the network should receive a packet from Alice and send it directly to S

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

data should not appear the same across multiple packets



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



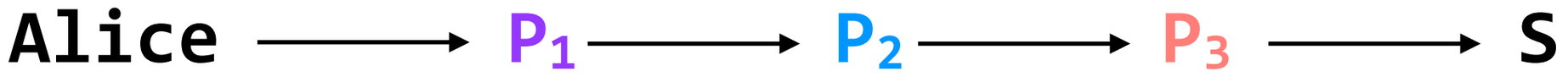
1. Alice adds layers of encryption to her packet

□ = encrypted with P_3's public key, etc.

XXX

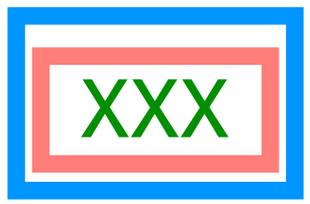
1. Alice adds layers of encryption to her packet

□ = encrypted with P_3's public key, etc.



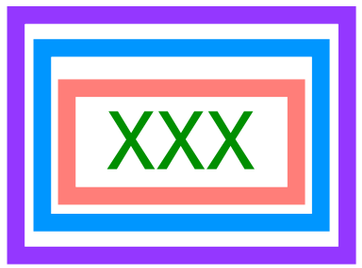
1. Alice adds layers of encryption to her packet

 = encrypted with P_3's public key, etc.



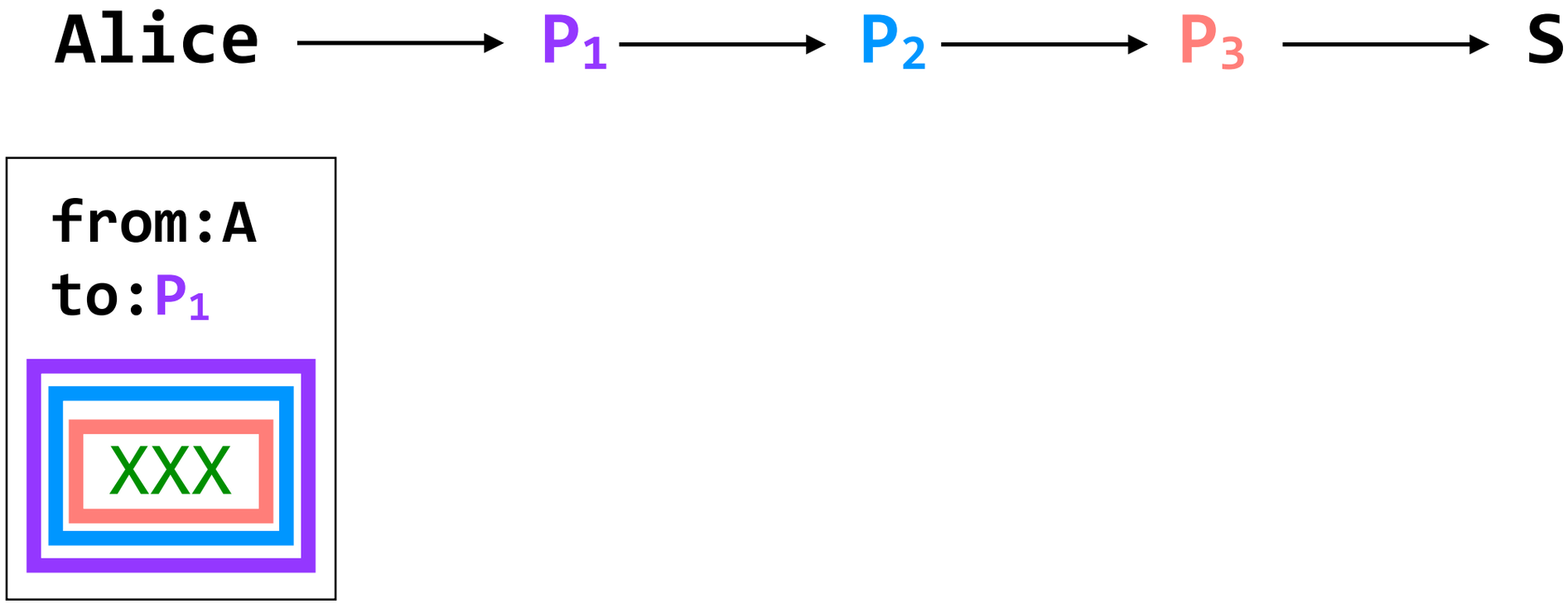
1. Alice adds layers of encryption to her packet

□ = encrypted with P_3's public key, etc.



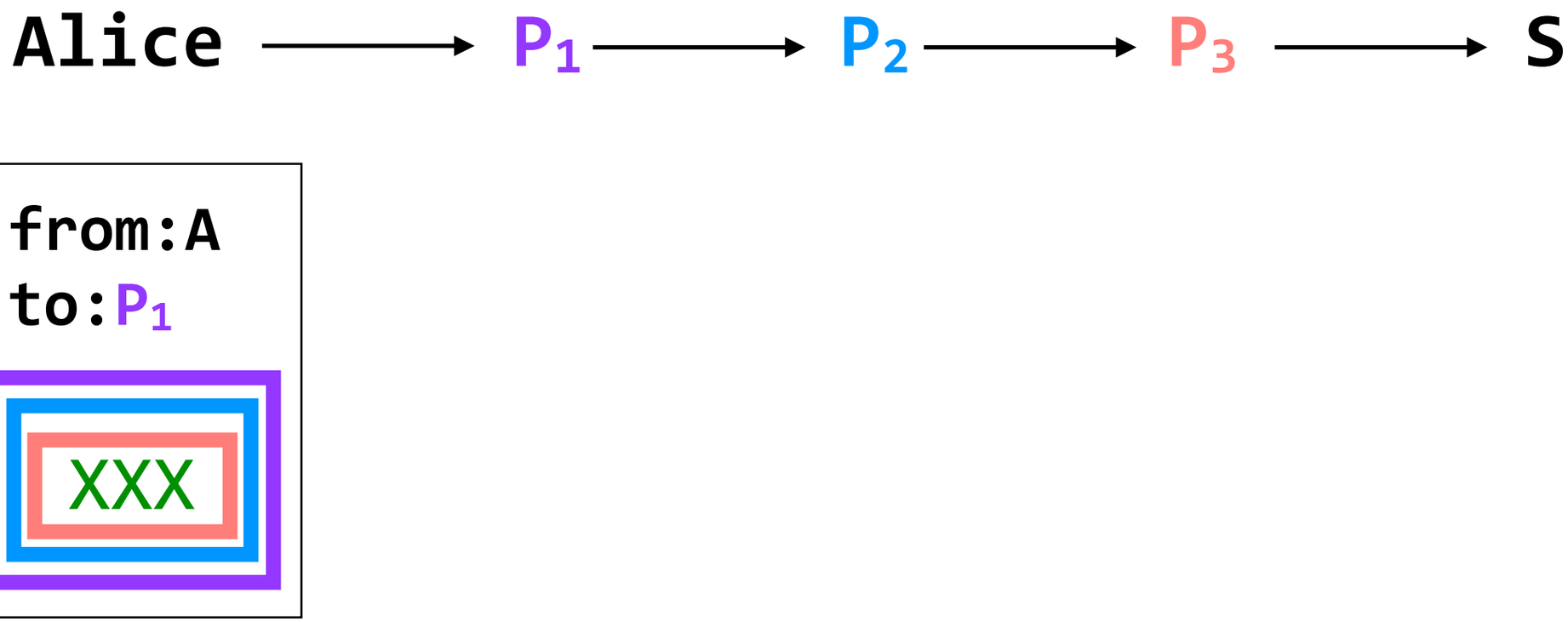
1. Alice adds layers of encryption to her packet

= encrypted with P_3's public key, etc.



1. Alice adds layers of encryption to her packet

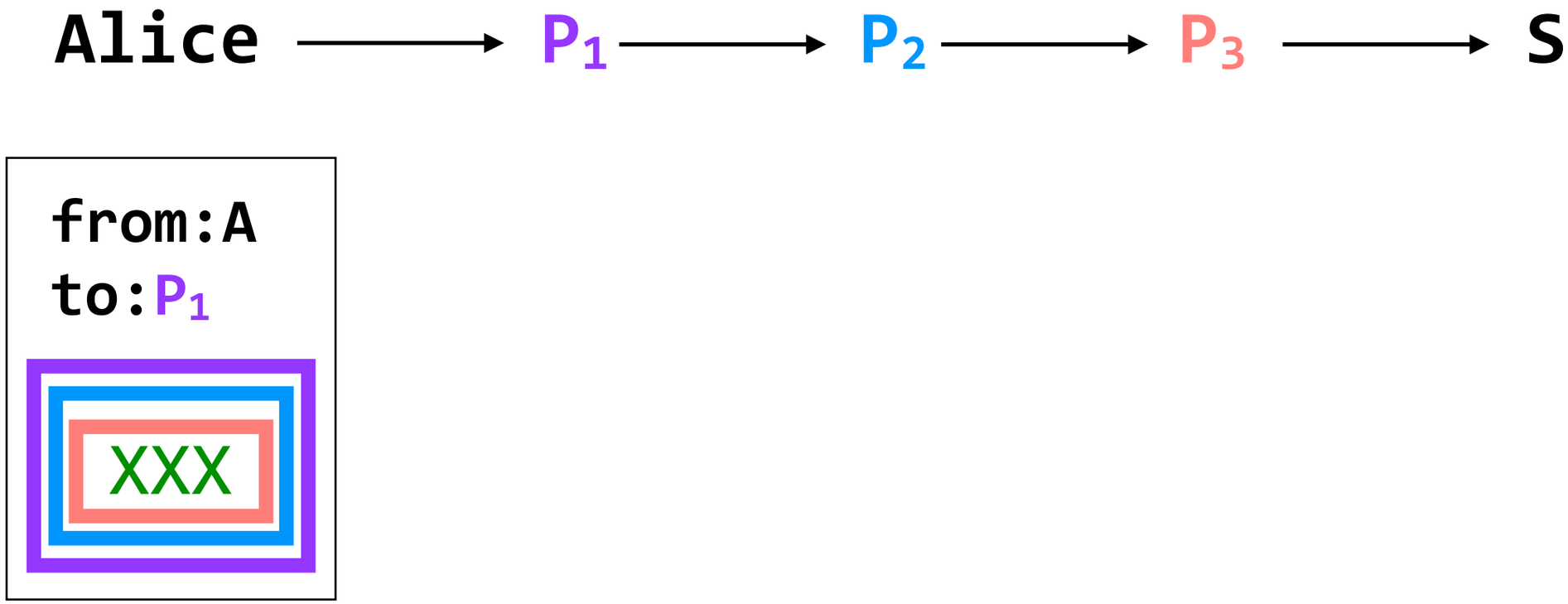
= encrypted with P_3's public key, etc.



(the circuit ID would also be included and encrypted)

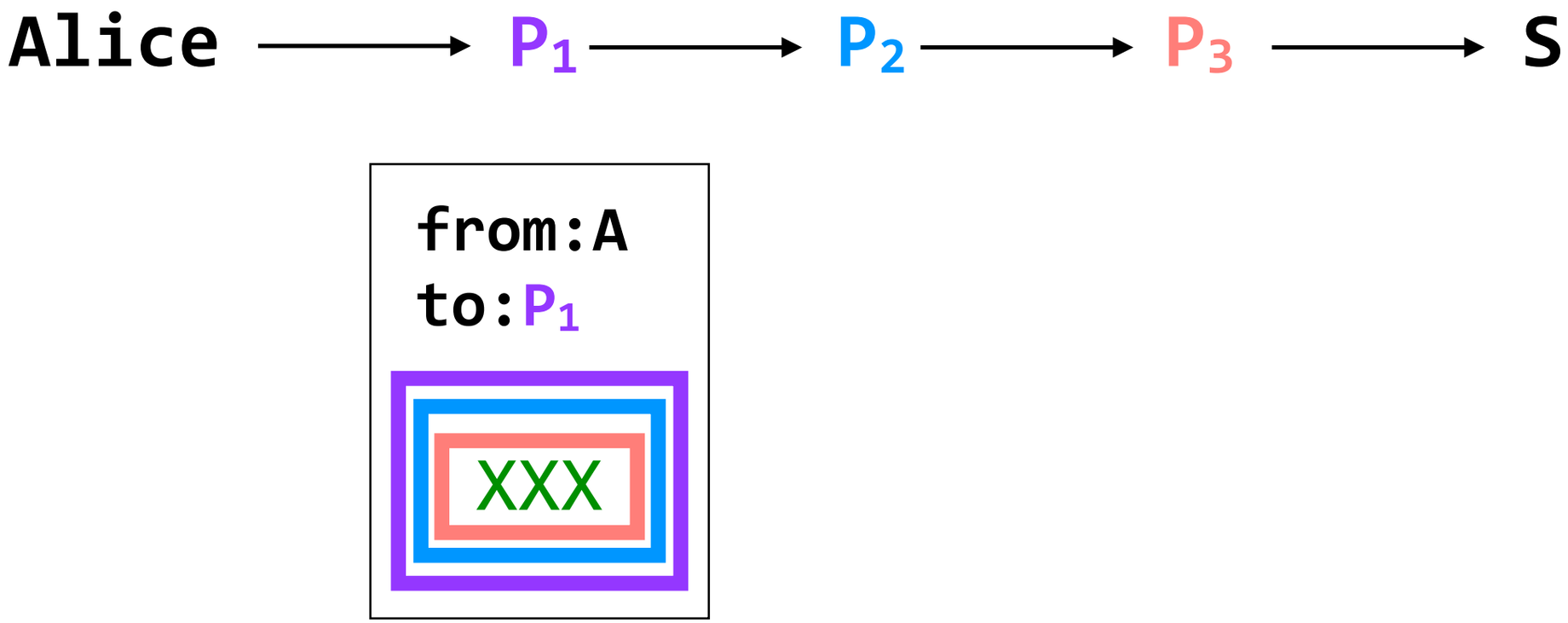
1. Alice adds layers of encryption to her packet

= encrypted with P_3's public key, etc.



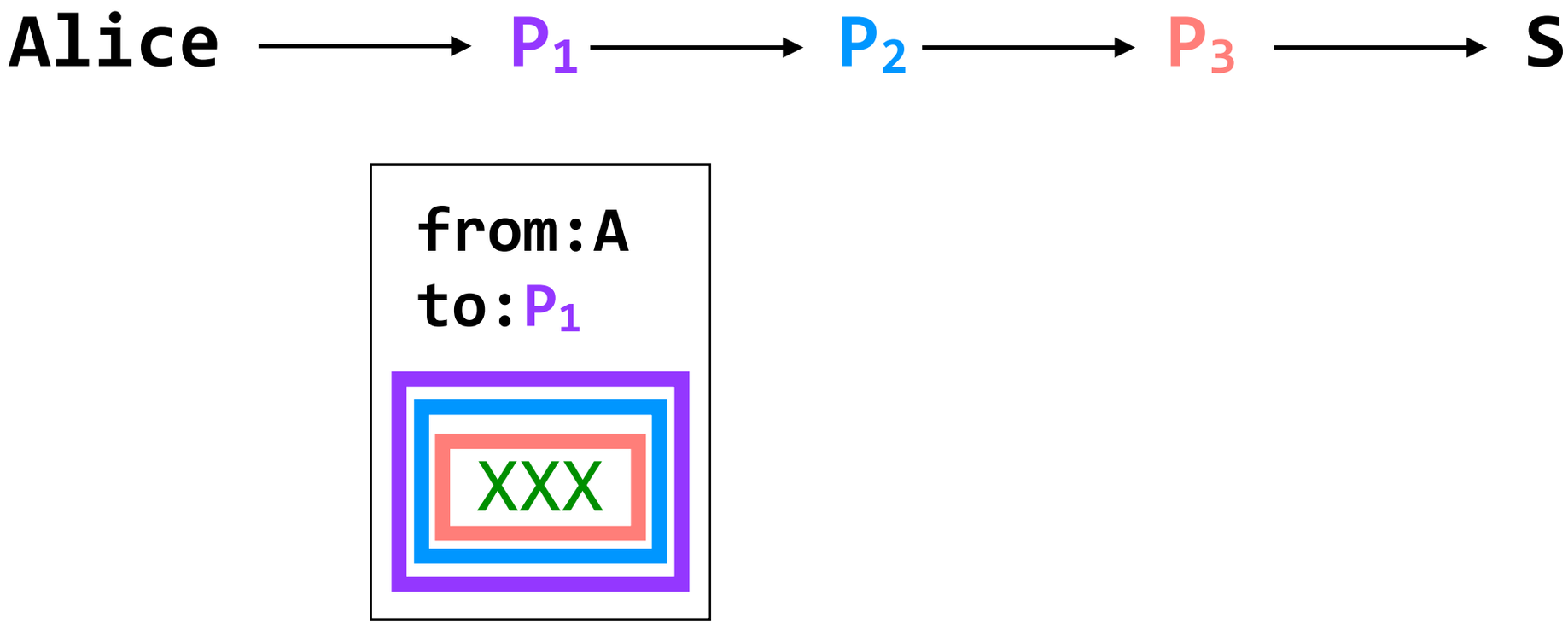
1. Alice adds layers of encryption to her packet

= encrypted with P_3's public key, etc.

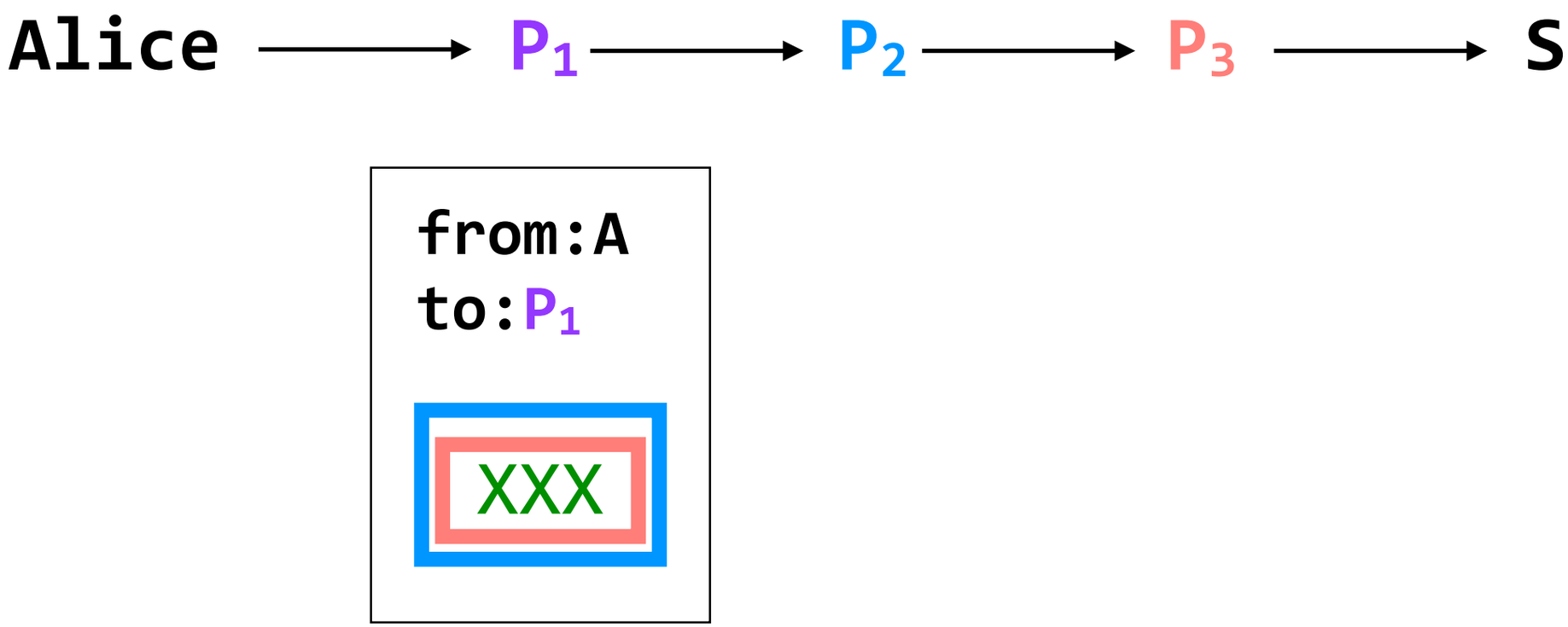


1. Alice adds layers of encryption to her packet

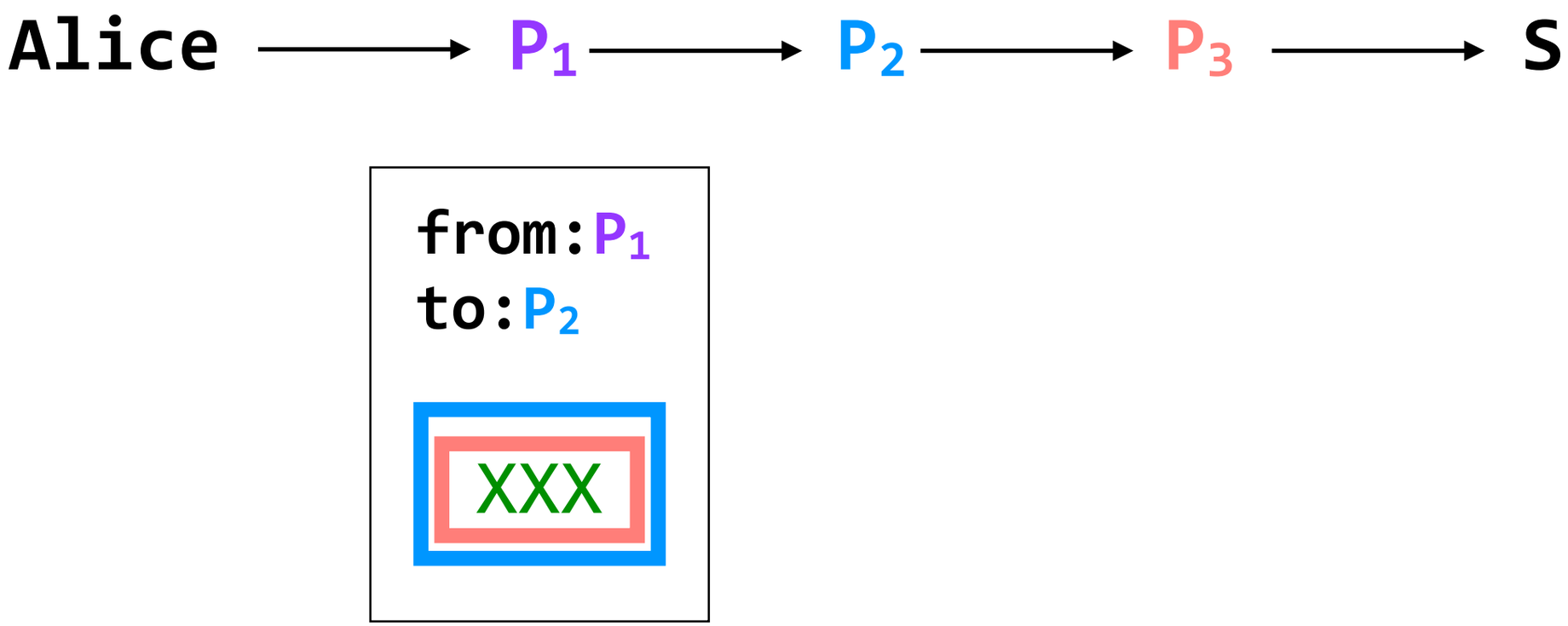
= encrypted with P_3's public key, etc.



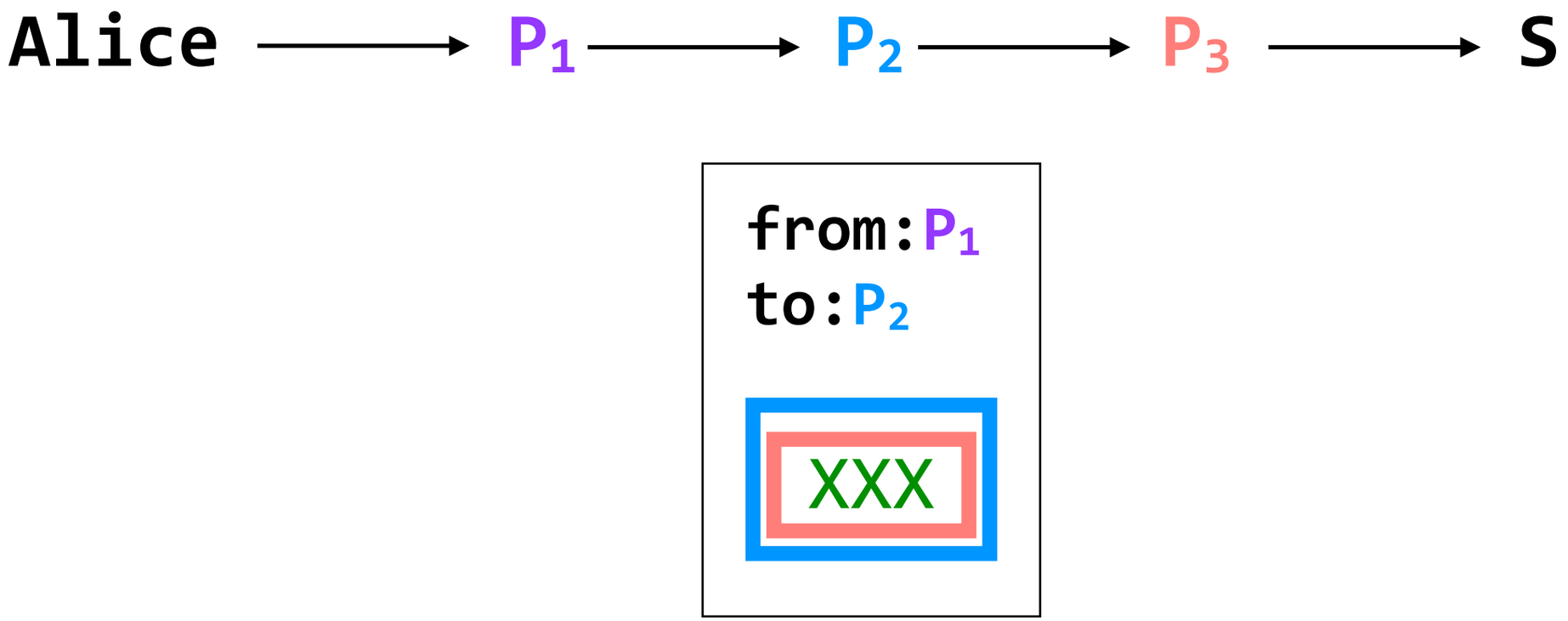
2. P_1 strips off one layer of encryption and edits the header



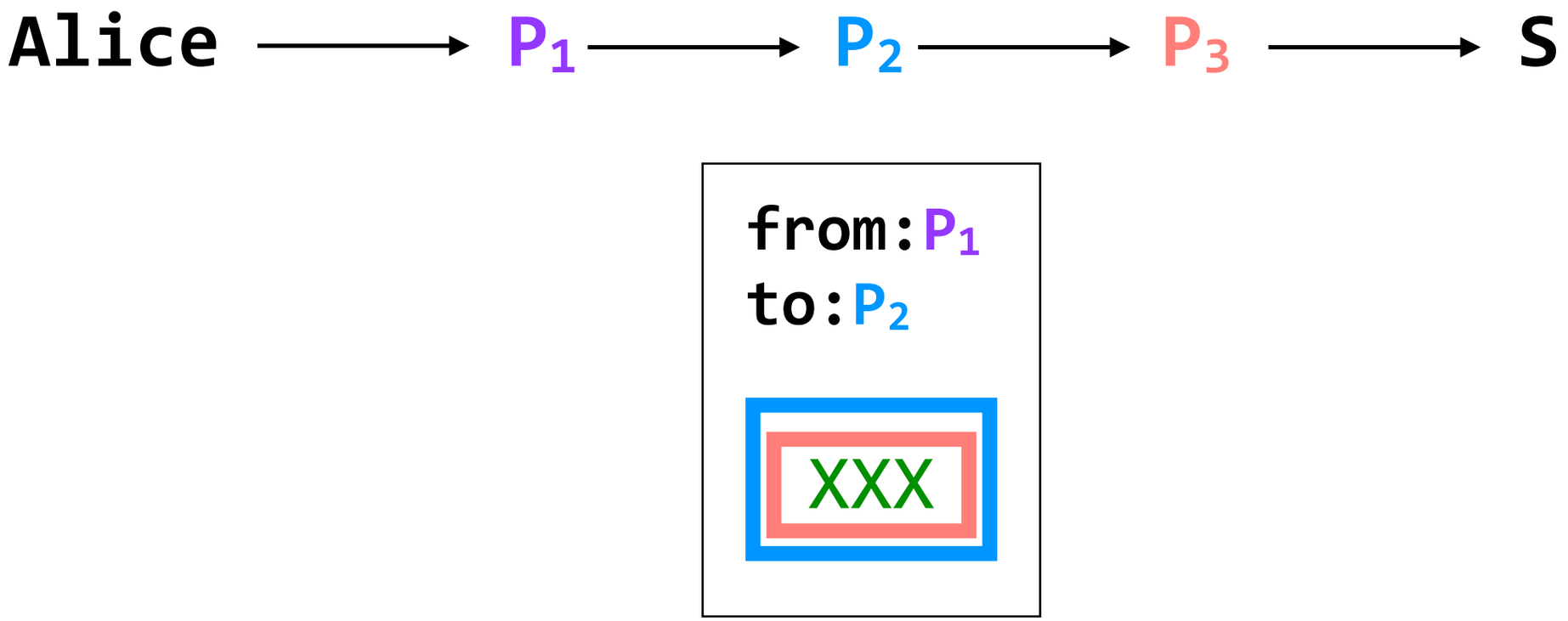
2. P_1 strips off one layer of encryption and edits the header



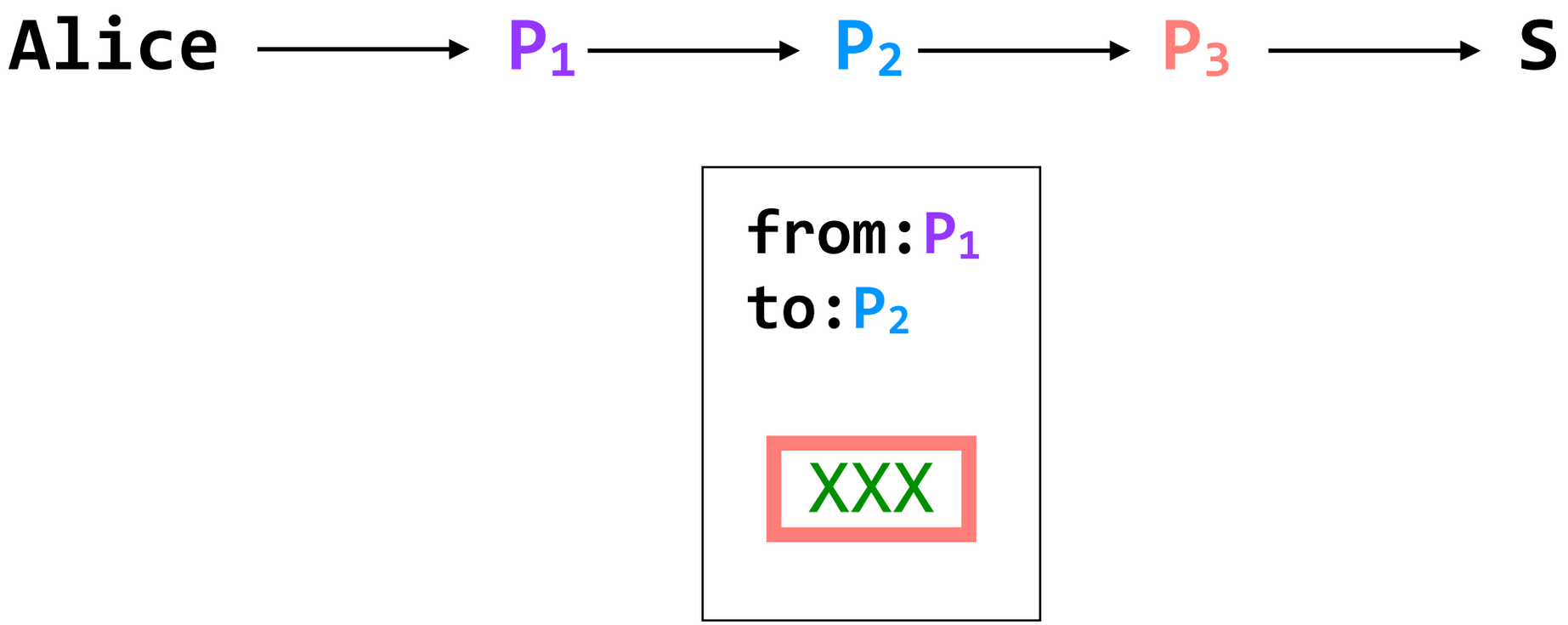
2. P_1 strips off one layer of encryption and edits the header



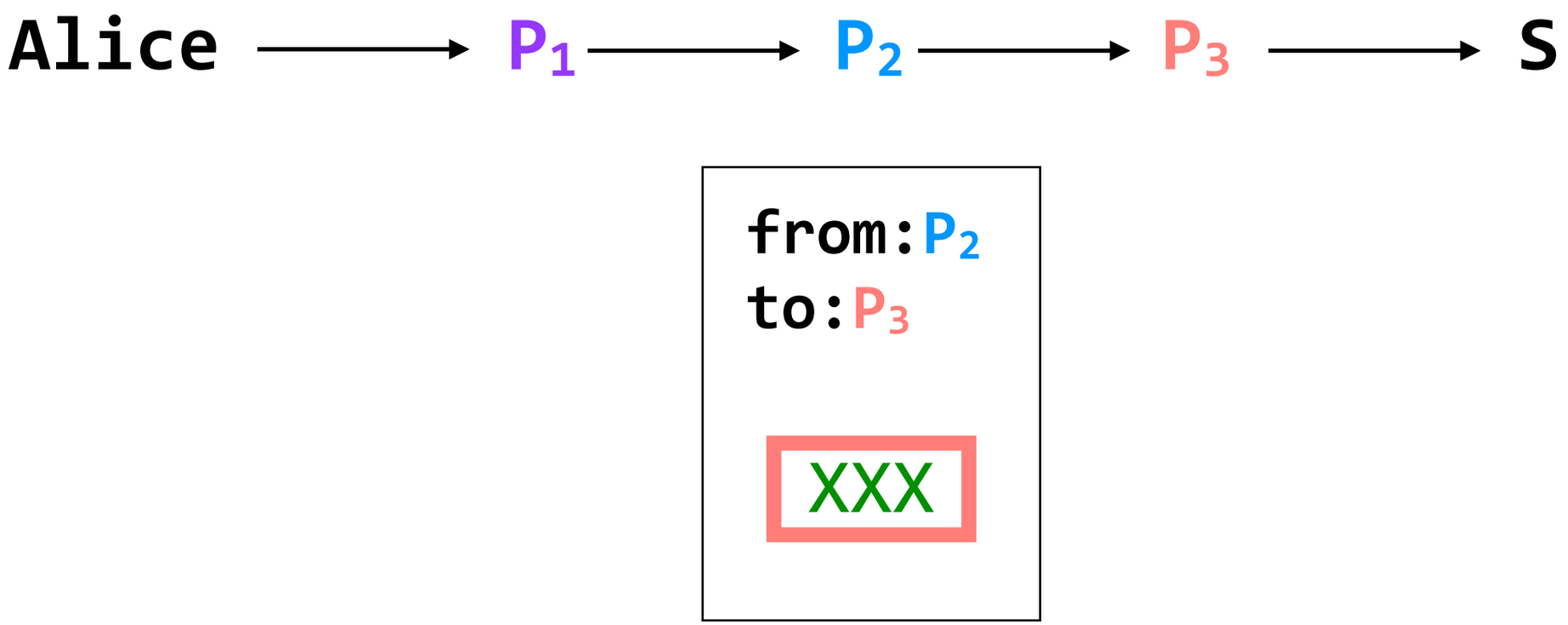
2. P_1 strips off one layer of encryption and edits the header



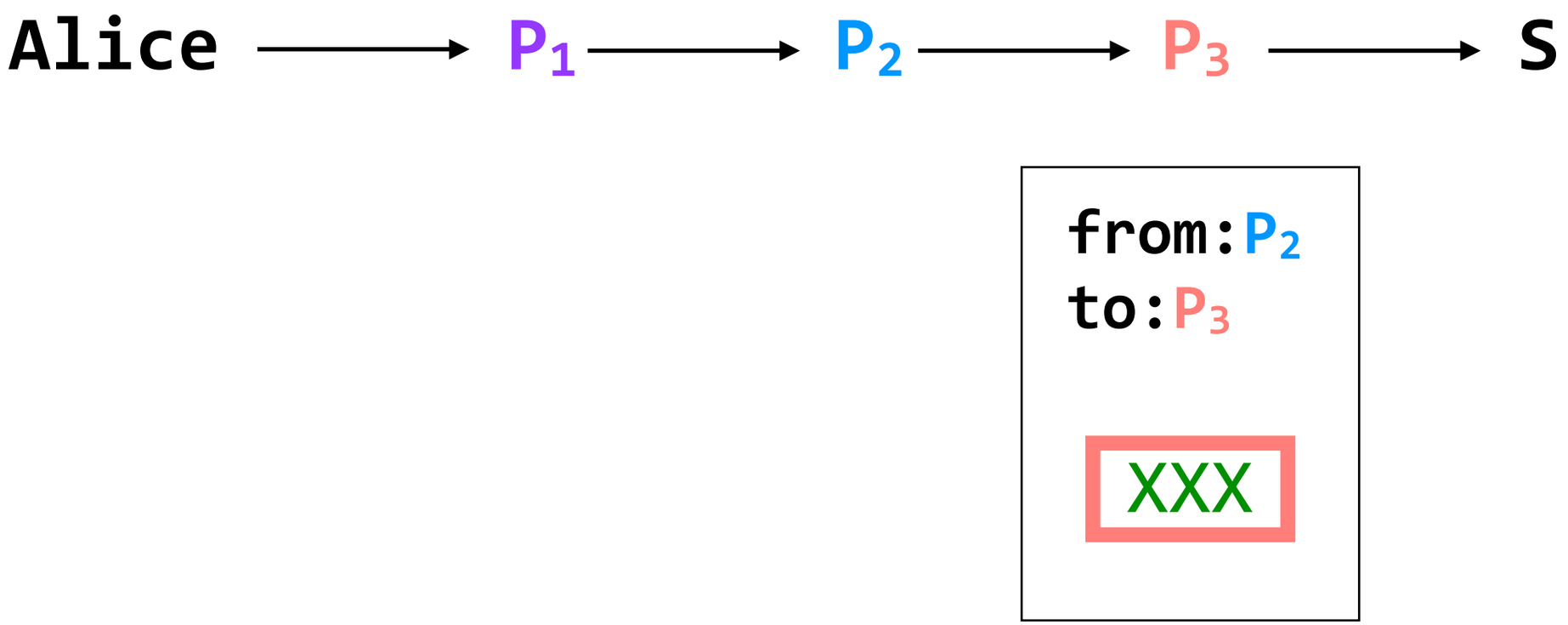
3. P_2 strips off one layer of encryption and edits the header



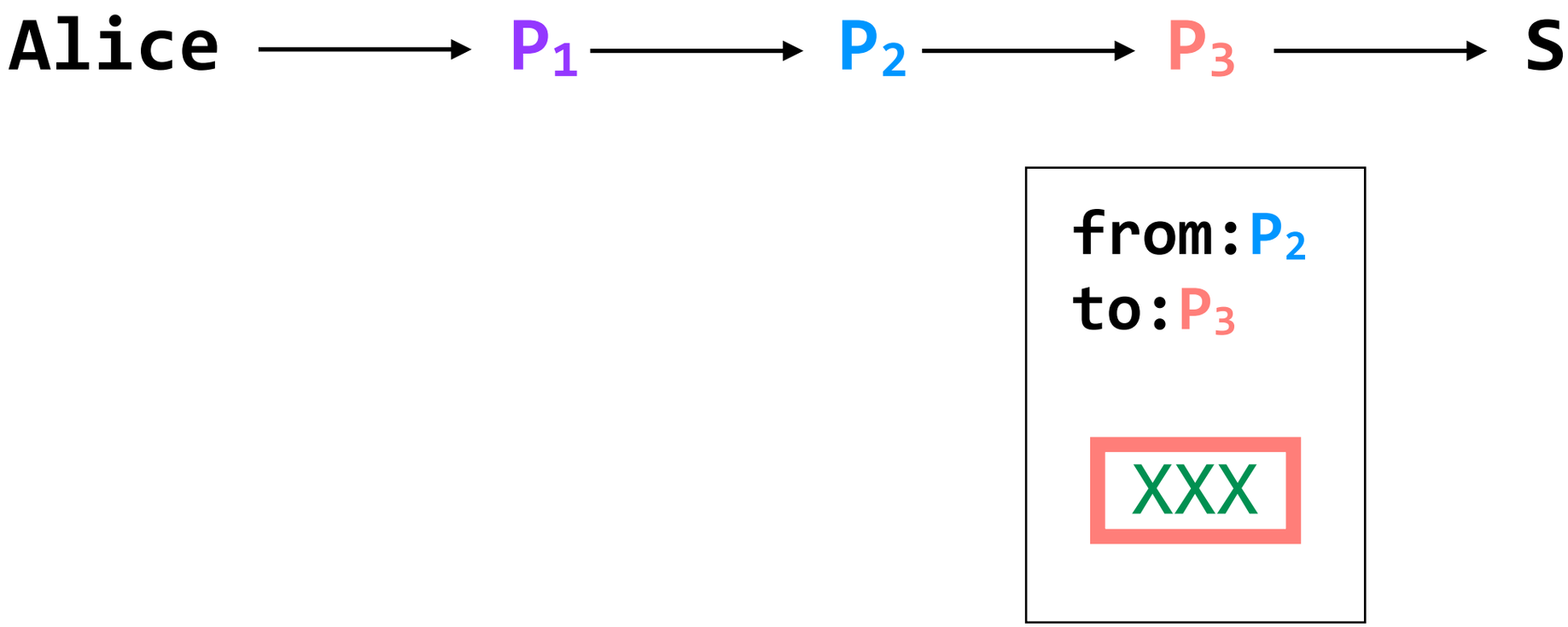
3. P_2 strips off one layer of encryption and edits the header



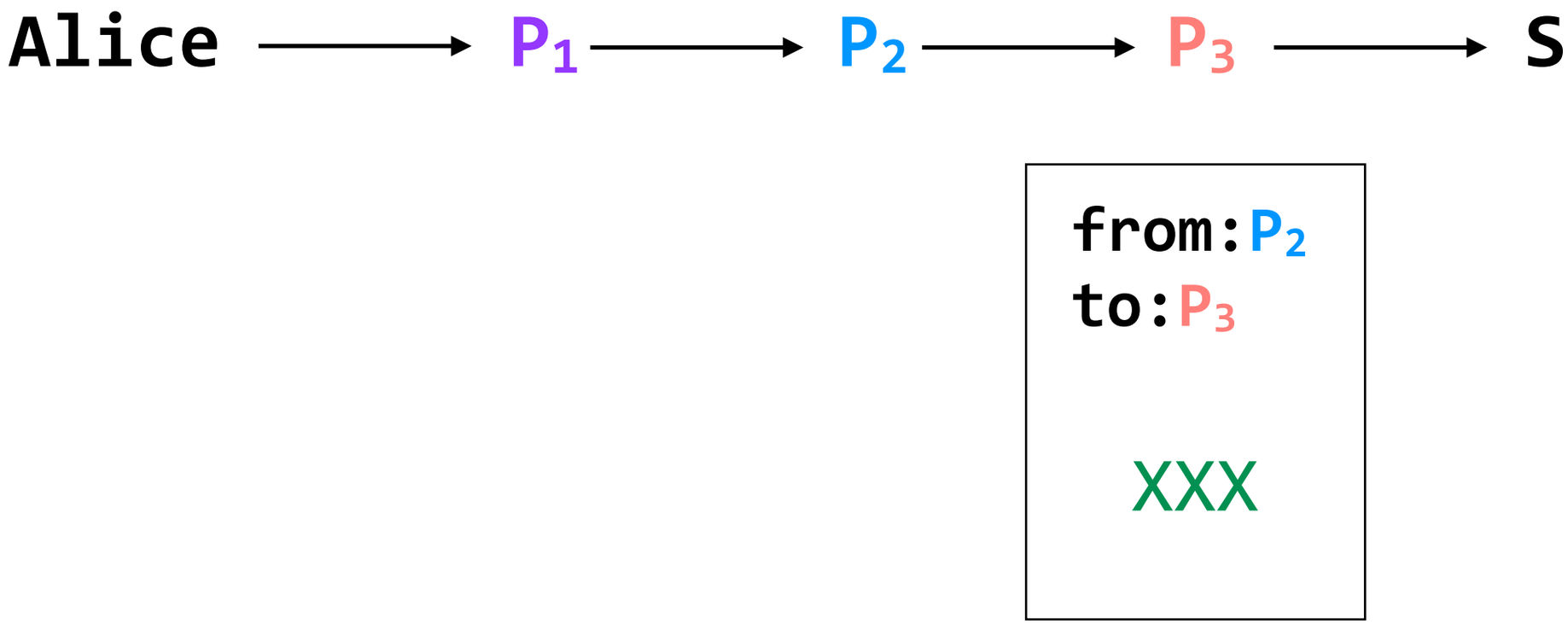
3. P_2 strips off one layer of encryption and edits the header



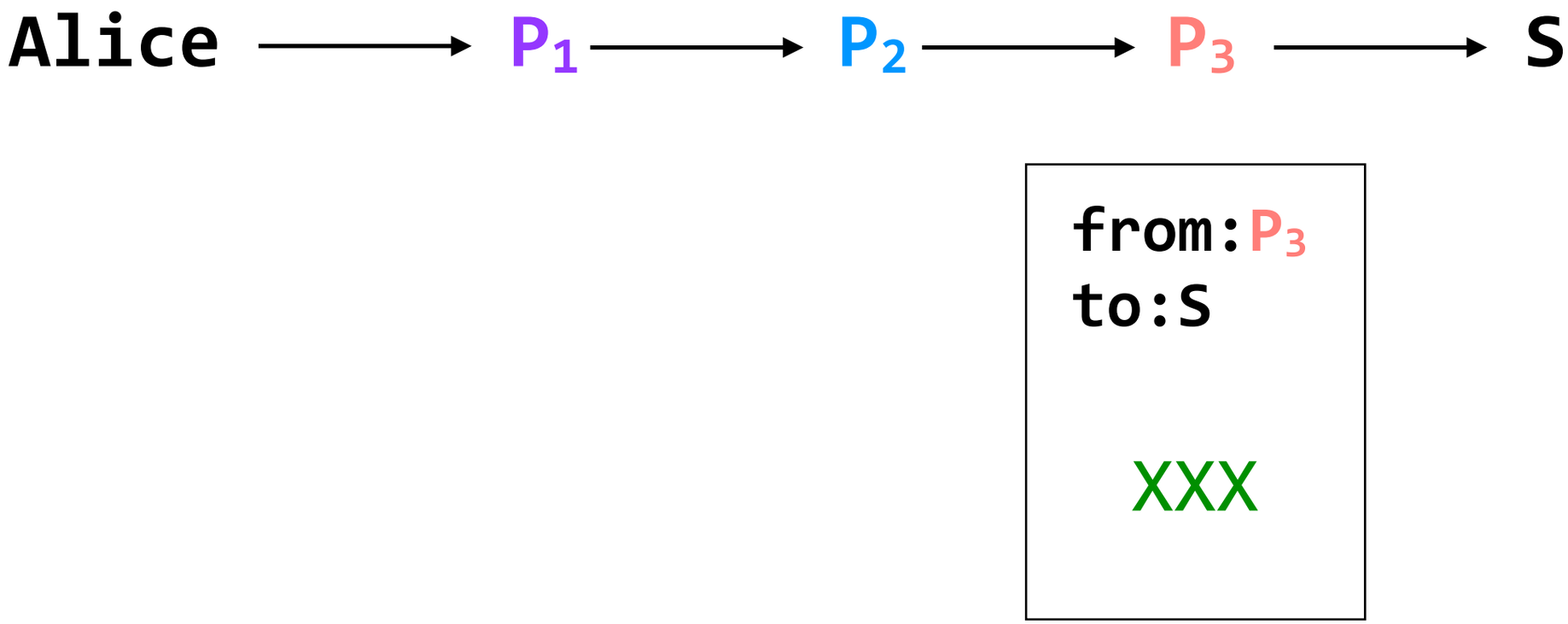
3. P_2 strips off one layer of encryption and edits the header



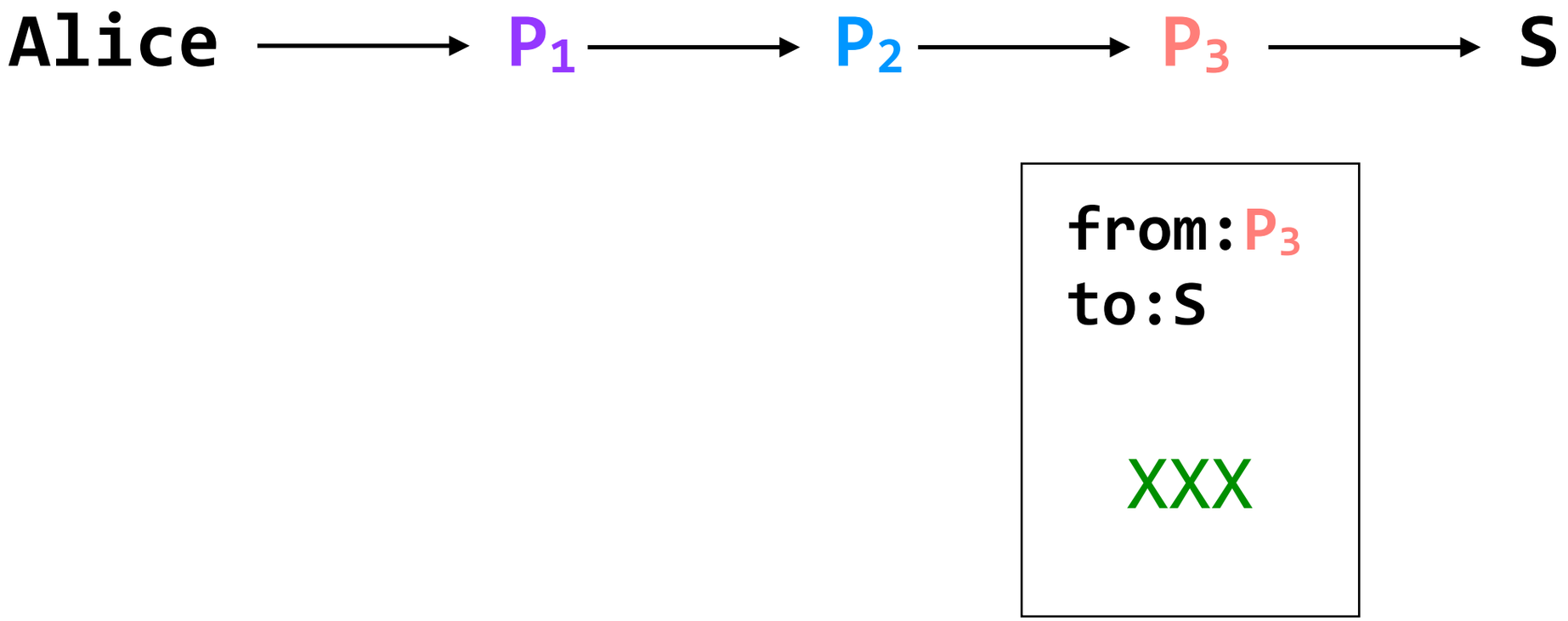
4. P_3 strips off one layer of encryption and edits the header



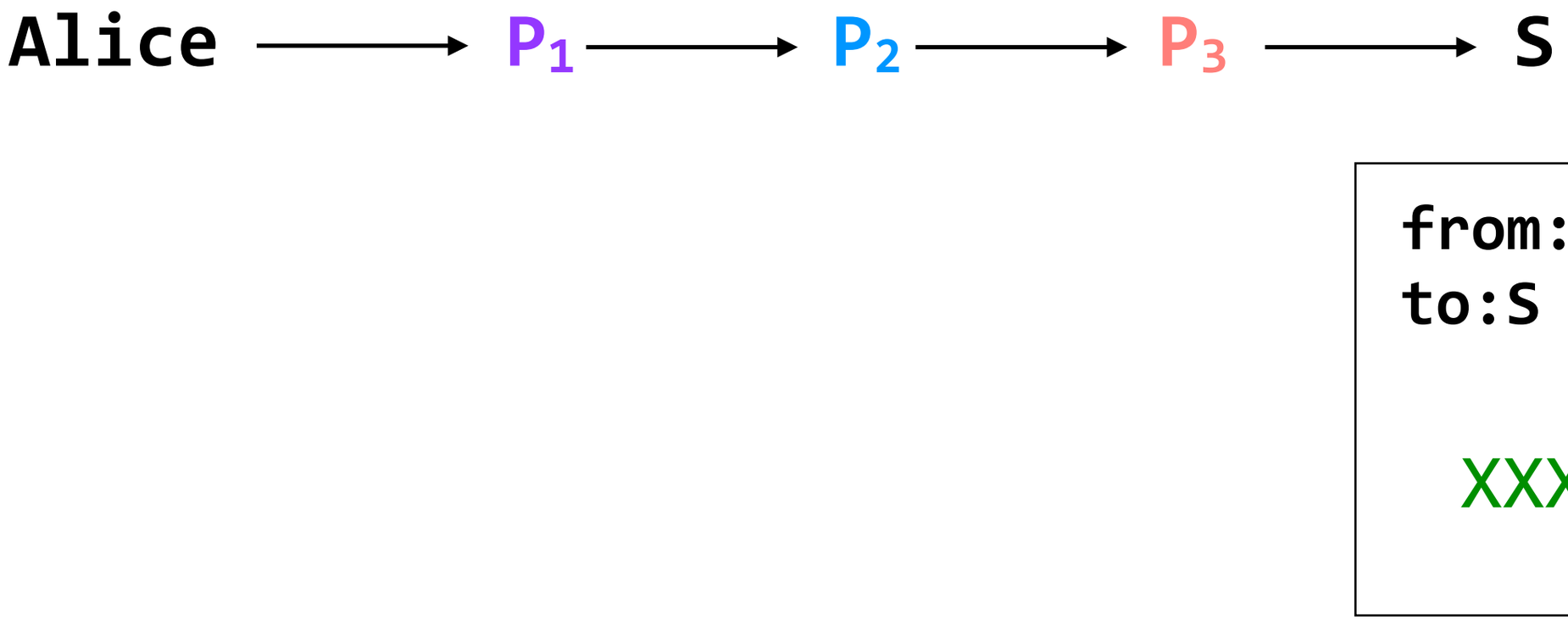
4. P_3 strips off one layer of encryption and edits the header



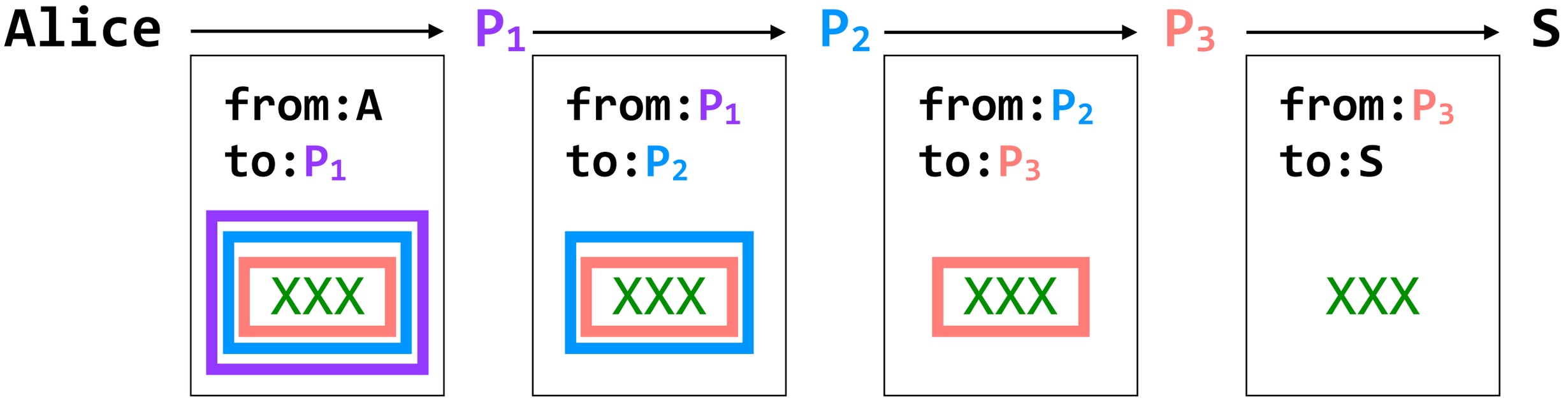
4. P_3 strips off one layer of encryption and edits the header



5. P_3 sends the packet to S



5. P_3 sends the packet to S



things to avoid

- 👍 no packet should say “from: Alice, to: S”
- 👍 no entity in the network should receive a packet from Alice and send it directly to S
- 👍 no entity in the network should keep state that links Alice to S
- 👍 data should not appear the same across multiple packets

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.

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.

- **Tor** provides anonymity for users, preventing attackers from linking a sender to its receiver.

- There are still ways to attack Tor, namely by **correlating traffic** from various points in the network.

- Both Tor and Bitcoin (last lecture) deal, at least somewhat, with **anonymity**. But more importantly, they solve interesting technical problems and use cryptography (and other techniques) in clever ways. Understanding how they work and why they're used will give you a better sense of how secure you are online.