Hi!

IM2000

IM2000
RSS/Email

IM2000
RSS/Email

Hypertext Mail Transport Protocol

IM2000
RSS/Email
HTMP

IM2000
RSS/Email
HTMP
Stubmail

Stubbymail

Stubbymail

with your hosts

Stubbymail

with your hosts

Julian Haight

Meng Weng Wong

Stubbymail

with your hosts

Julian Haight

(Spamcop)

Meng Weng Wong

Stubbymail

with your hosts

Julian Haight

(Spamcop)

Meng Weng Wong

(Pobox.com)

Stubbymail

with your hosts

Julian Haight

(Spamcop)

Meng Weng Wong

(SPF)

The opposite of every great idea is another great idea. – Niels Bohr

SMTP push

SMTP	**push**
Stubbymail	**pull**

SMTP	**push**
Stubbymail	**pull**
The Web	**pull**

SMTP	**push**
Stubbymail	**pull**
The Web	**pull**
RSS	**pull**

spam is an unsubscribe problem

“omg this blog won’t stop sending me new posts”

“i keep going to this website against my will!”

Dear Mailing List,

please stop sending me stuff

Dear Mailing List,

please stop sending me stuff

…WRONG!!!

Please to be putting power back in the hands of the user

djb proposed IM2000 about ten years ago

It'll never work!

Science

Science

=

Disprove Hypothesis

The Plan:

1. Build it.
2. Oh, it doesn’t work.
3. Fine, you win.

The Goal:

Meng and Julian can email each other without worrying about spam filtering.

Not The Goal:

Convince stubborn grouchy old-timers to stop using email.

Not The Goal:

Get everybody in the world to switch.

Email will go on.

This is just another thing.

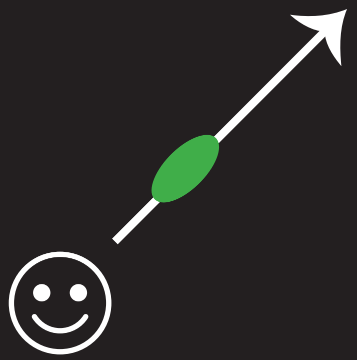
Email will go on.

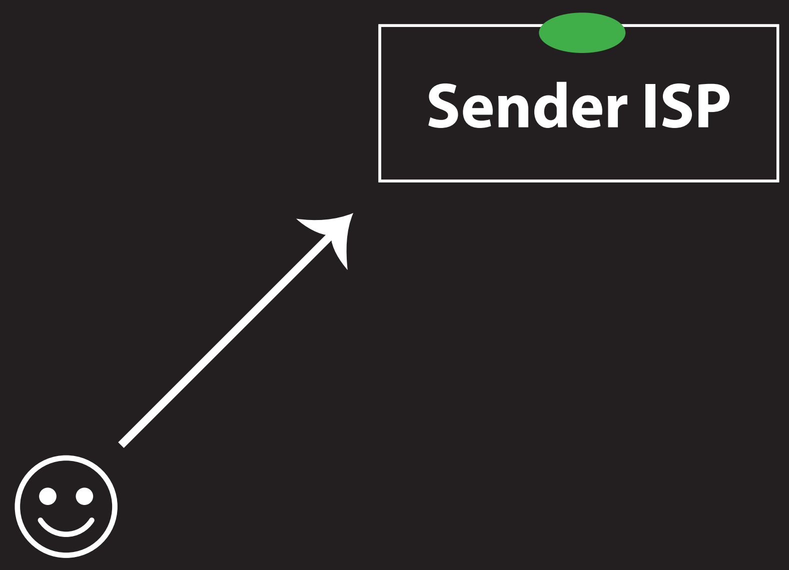
Spam couldn’t kill email.

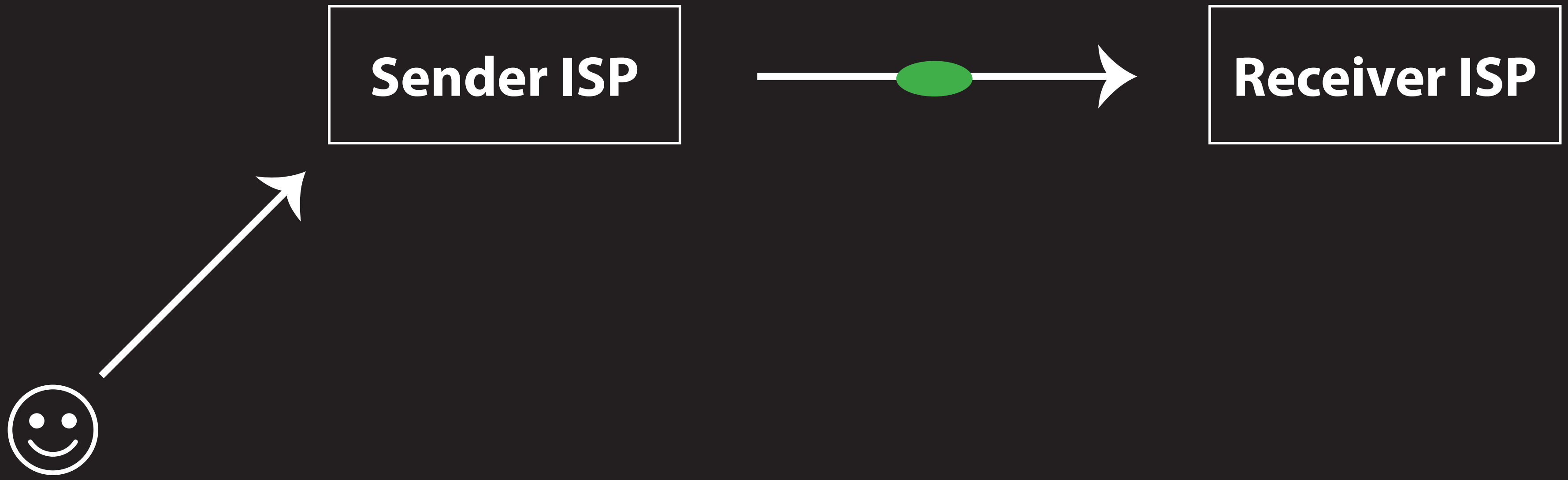
Email will go on.

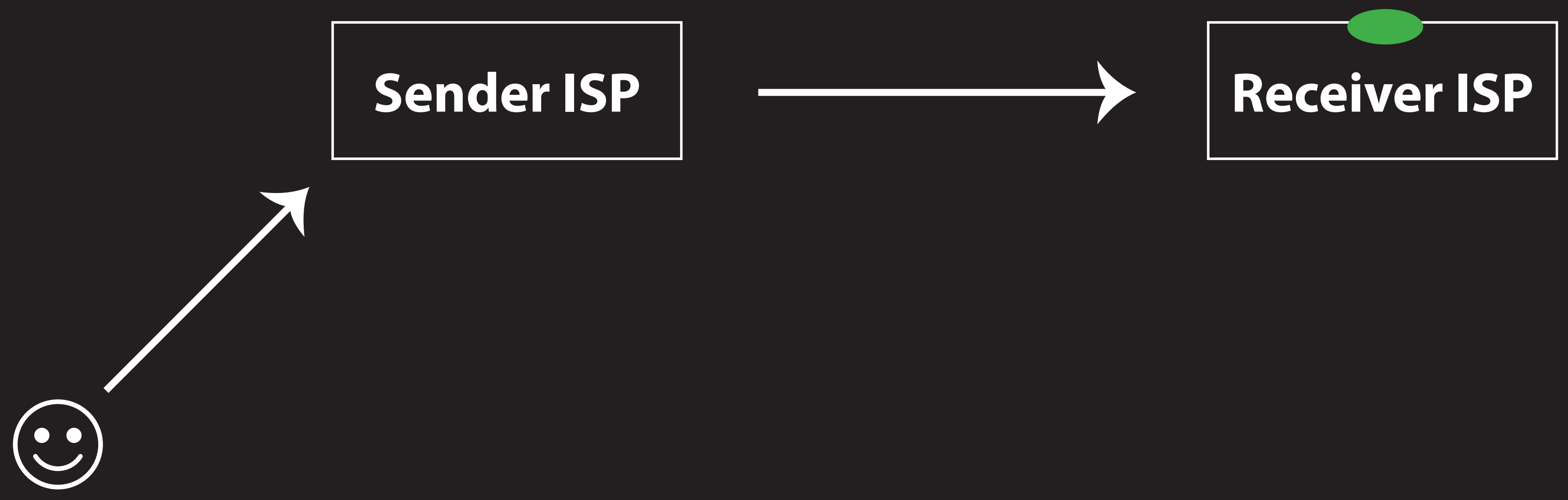
Spam couldn't kill email.
No way we can.

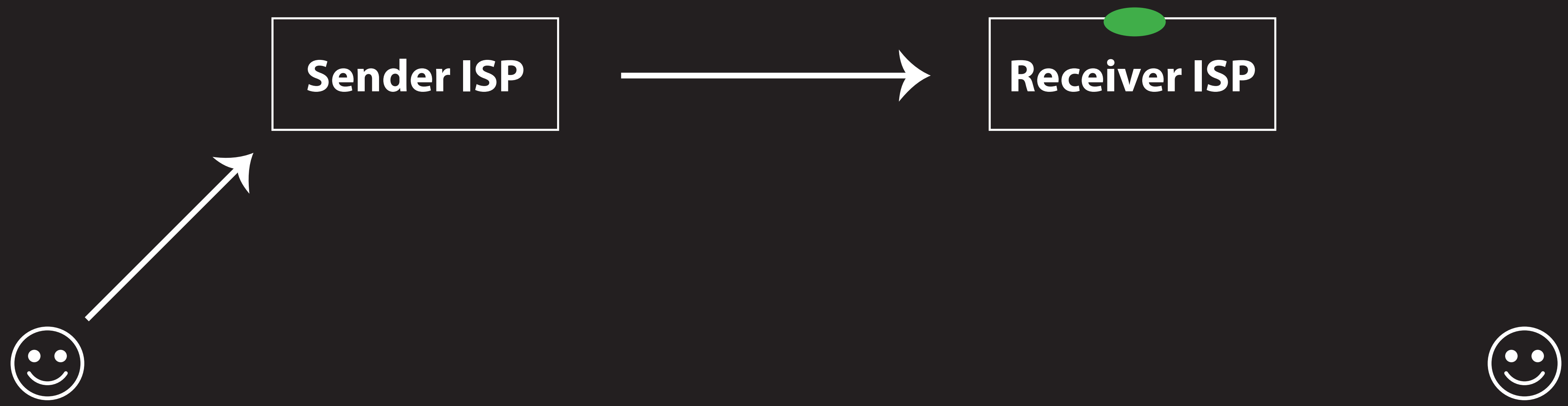
☺

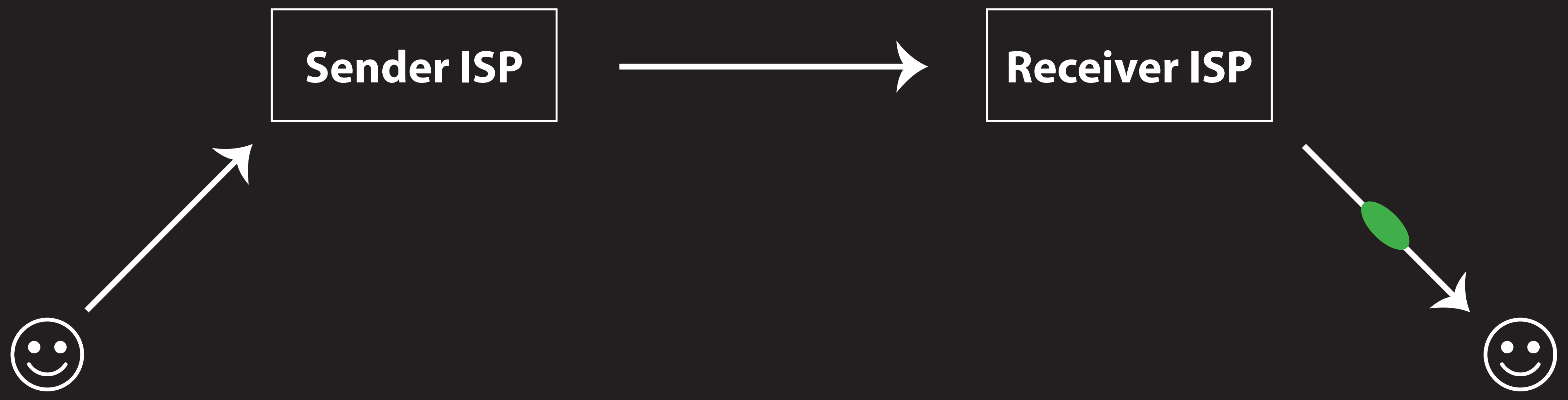


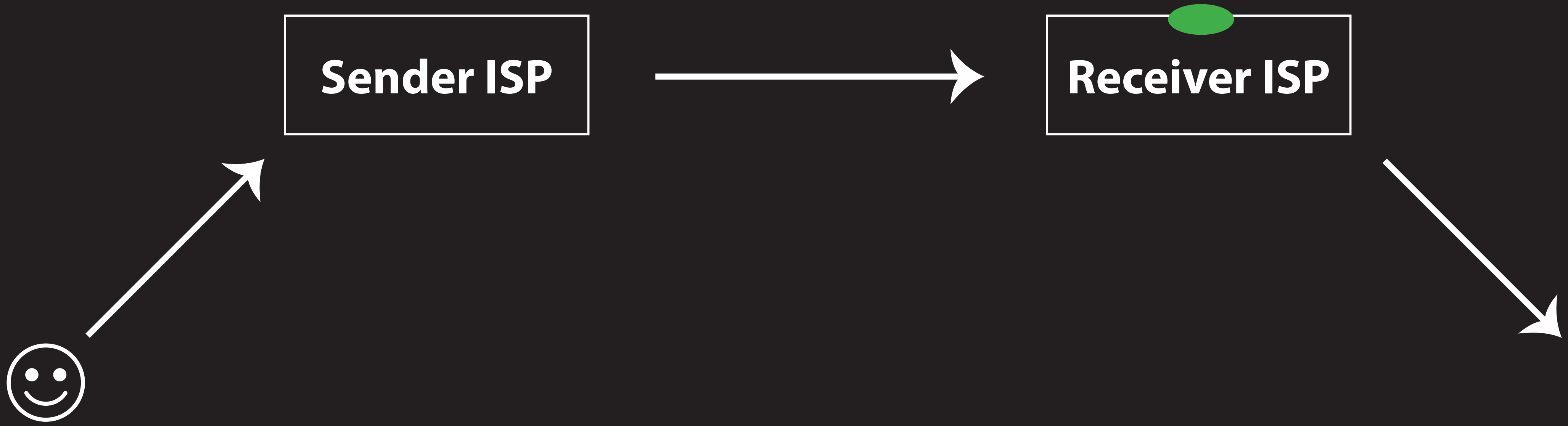

Sender ISP

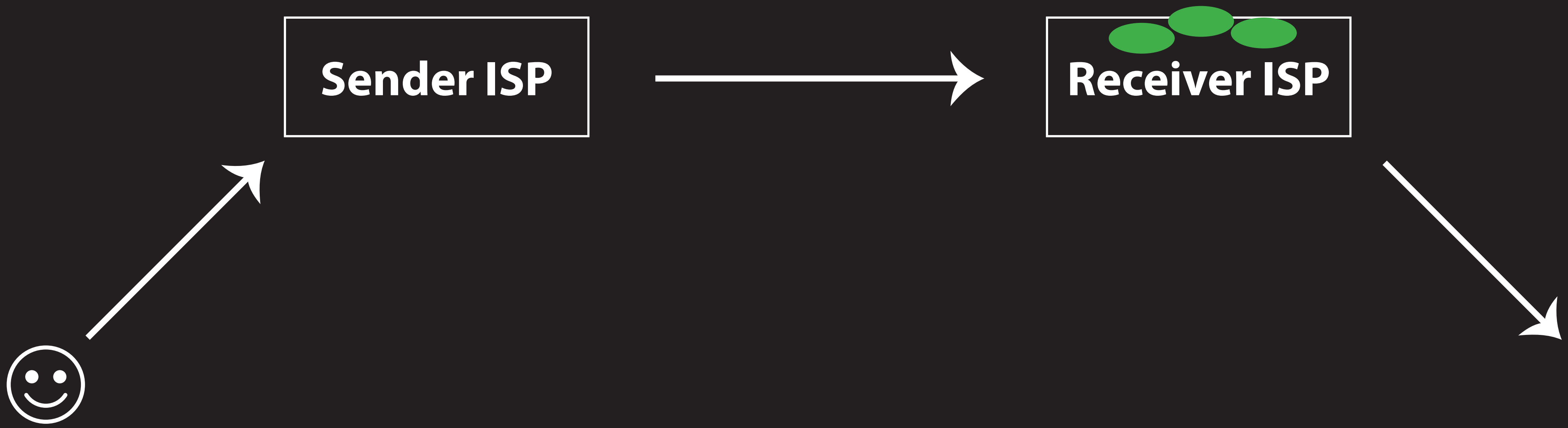

Sender ISP
Receiver ISP

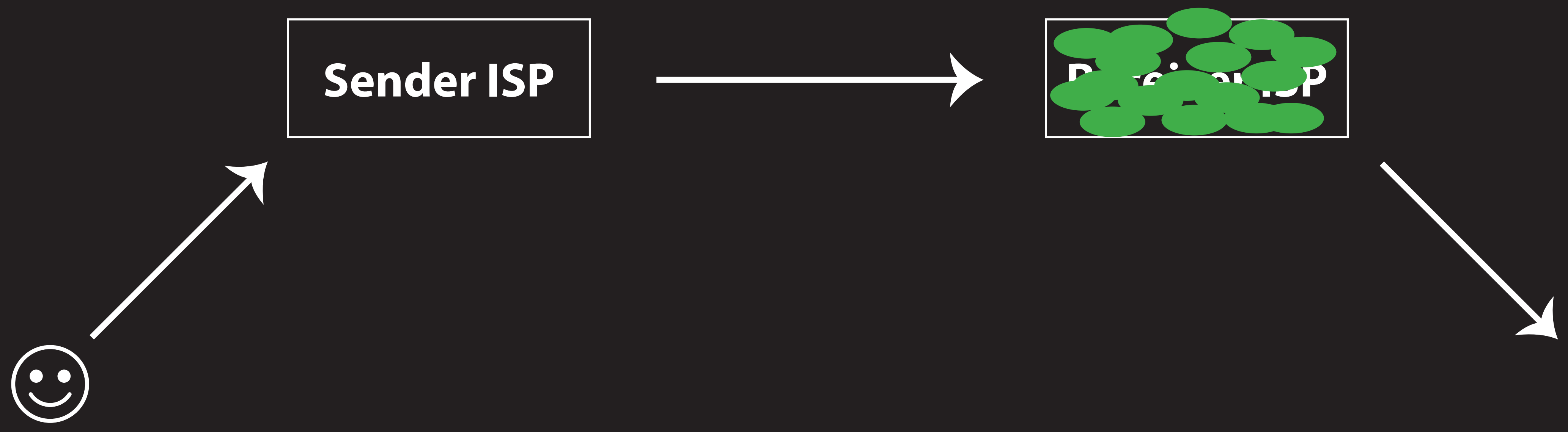

Sender ISP
Receiver ISP

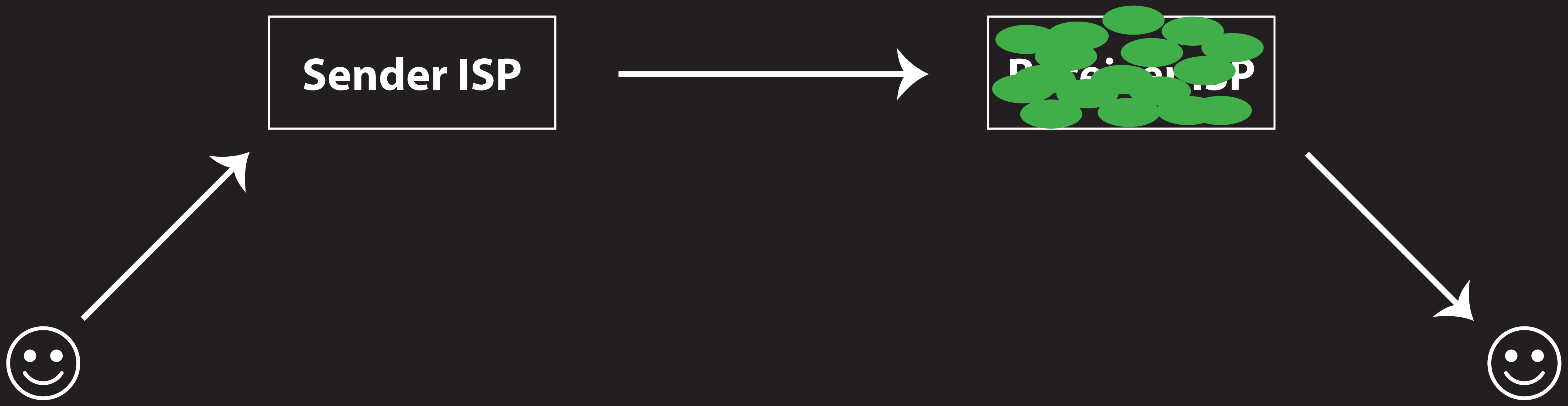

Sender ISP
Receiver ISP

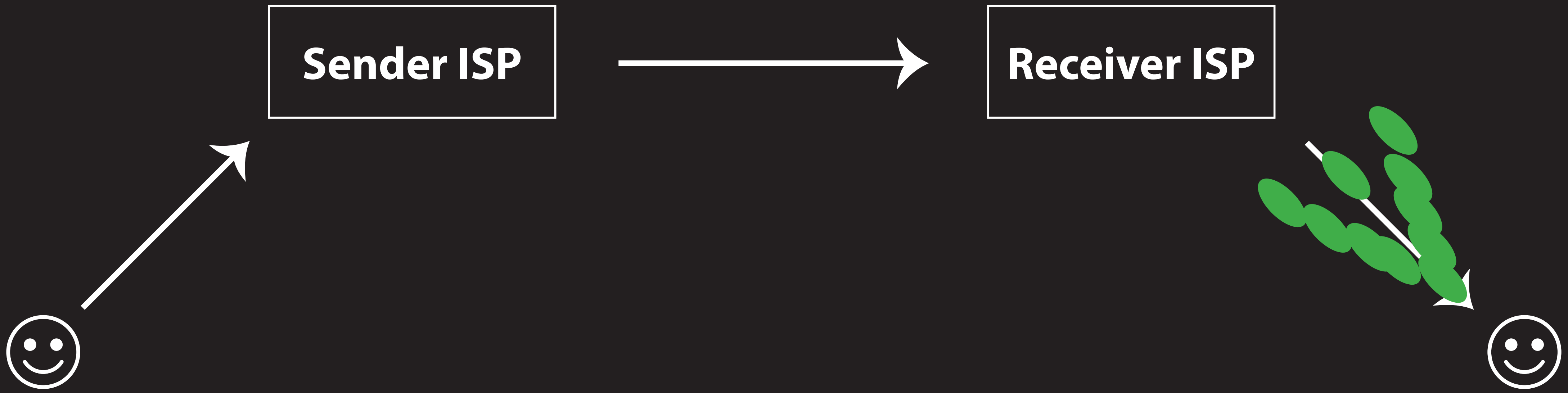

Sender ISP
Receiver ISP
☺
☺

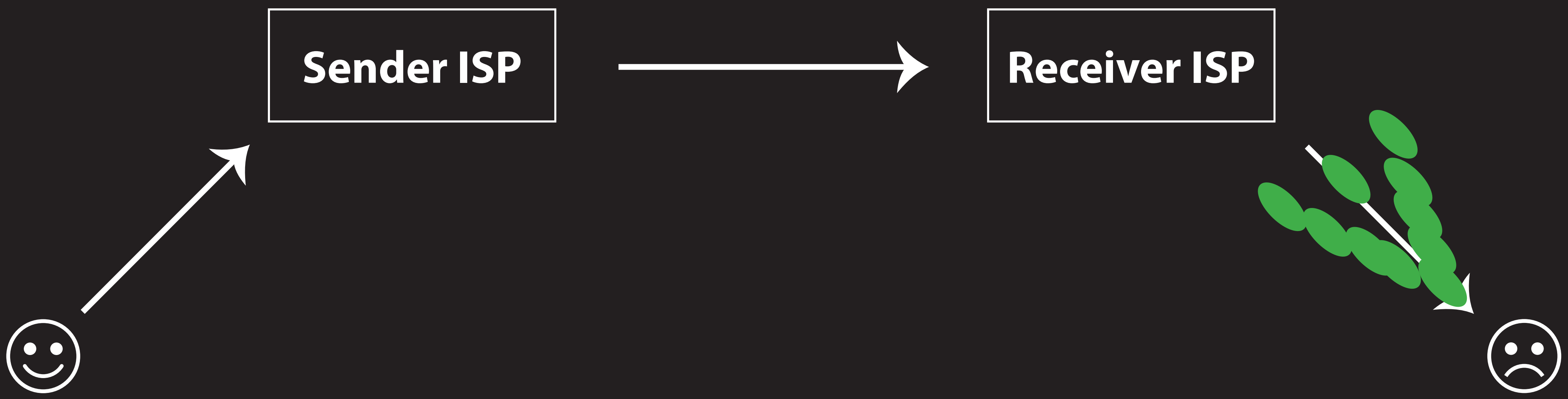

Sender ISP
Receiver ISP

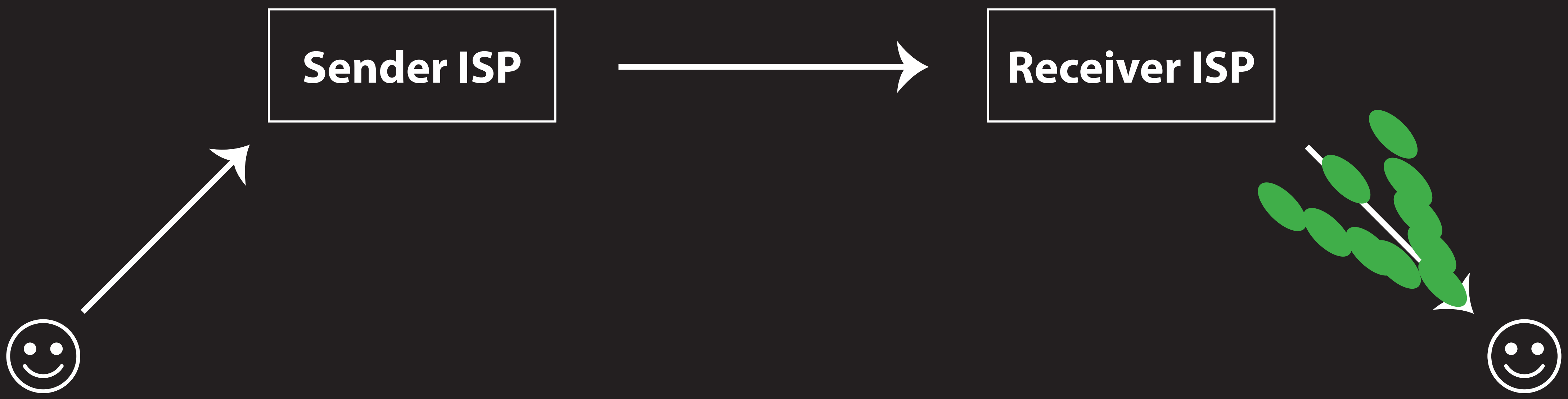

Sender ISP
Receiver ISP

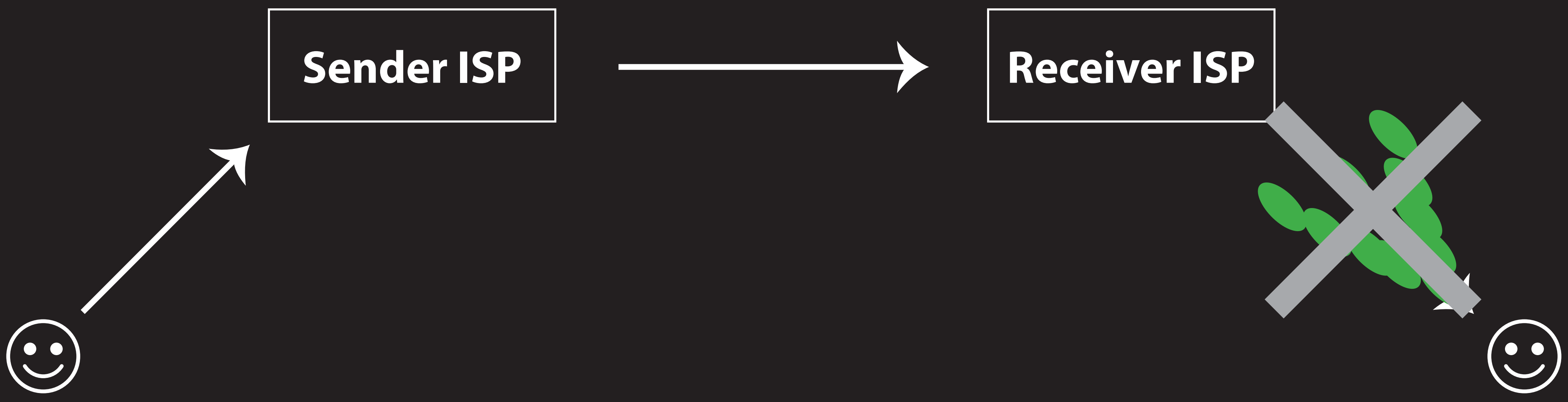

Sender ISP
Receiver ISP

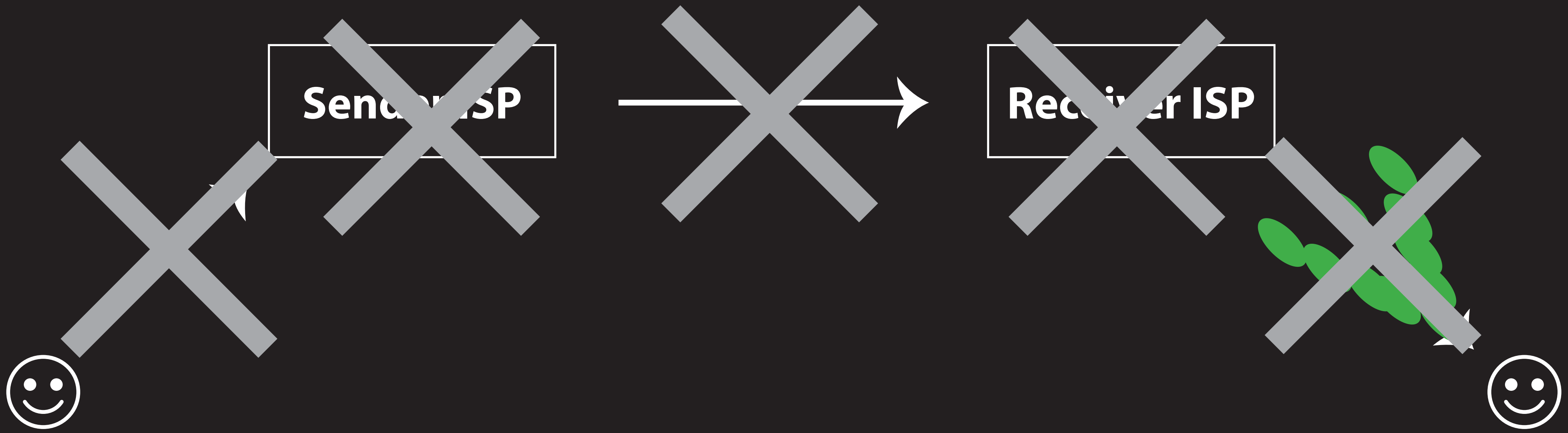

Sender ISP
Receiver ISP
☺
☺

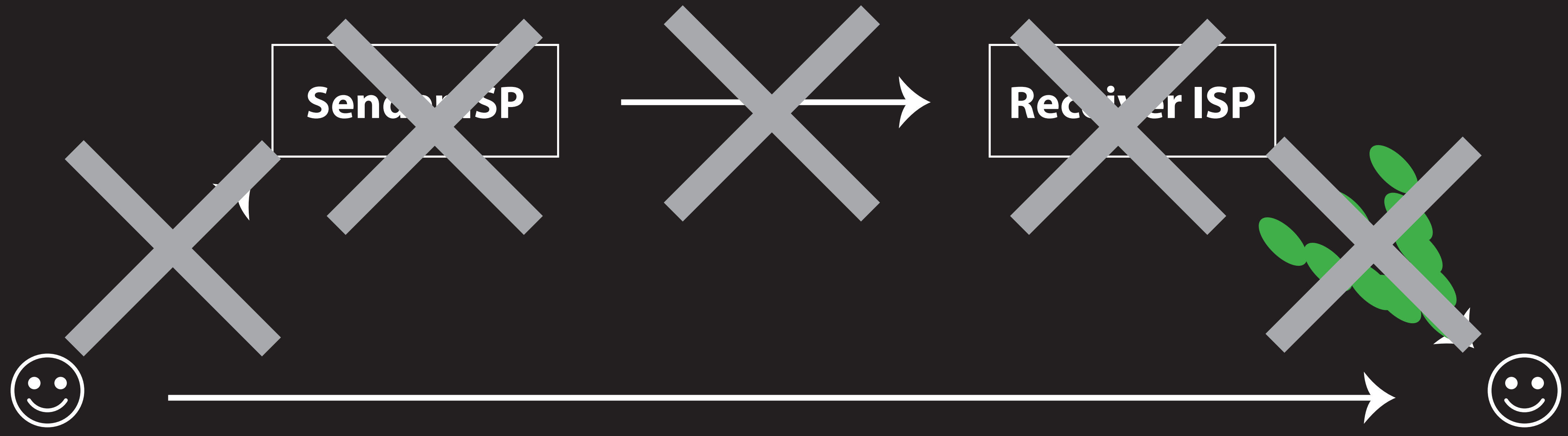

Sender ISP
Receiver ISP


Sender ISP
Receiver ISP

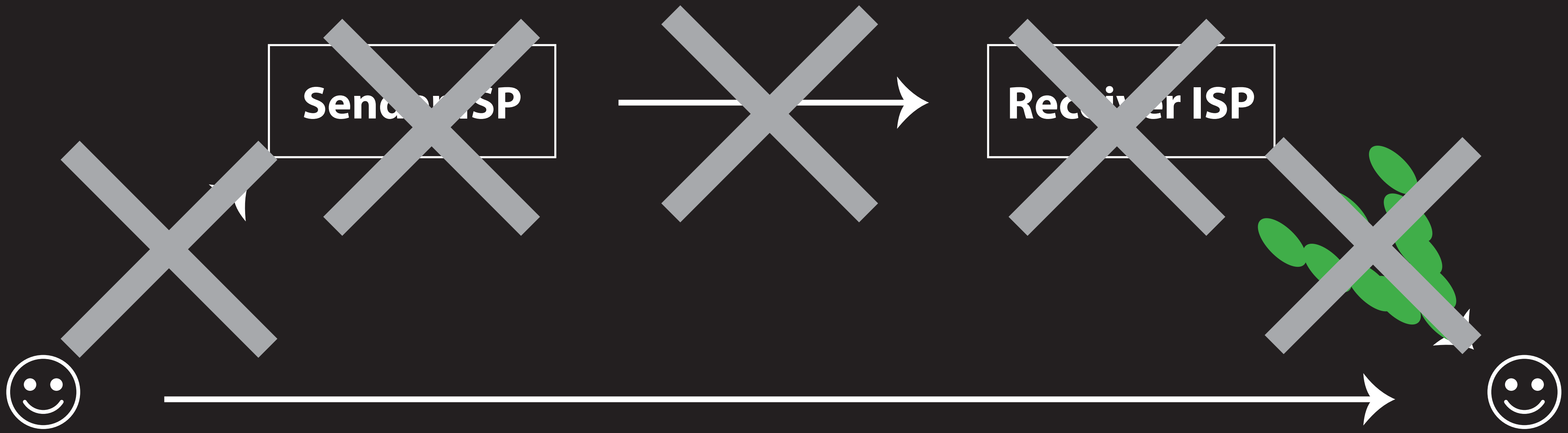

Sender ISP
Receiver ISP
☺
☺


Sender ISP
Receiver ISP


Sender ISP
Receiver ISP



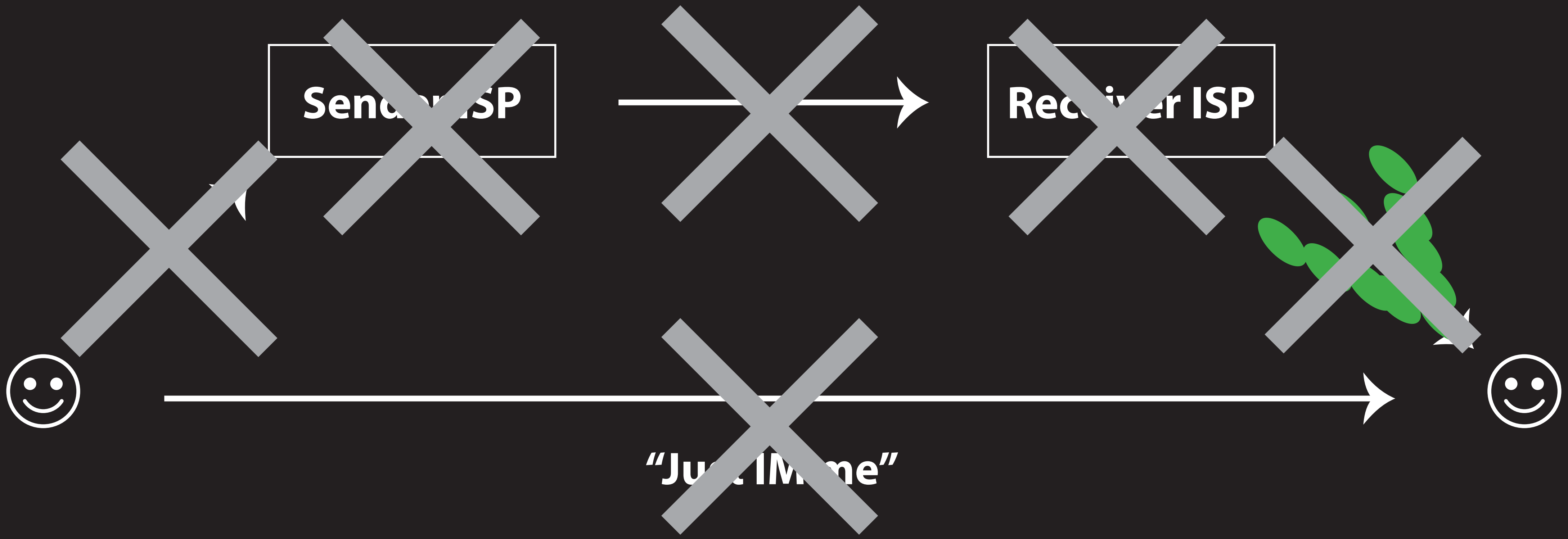
"Just IM me"



"Just IM me"

but!

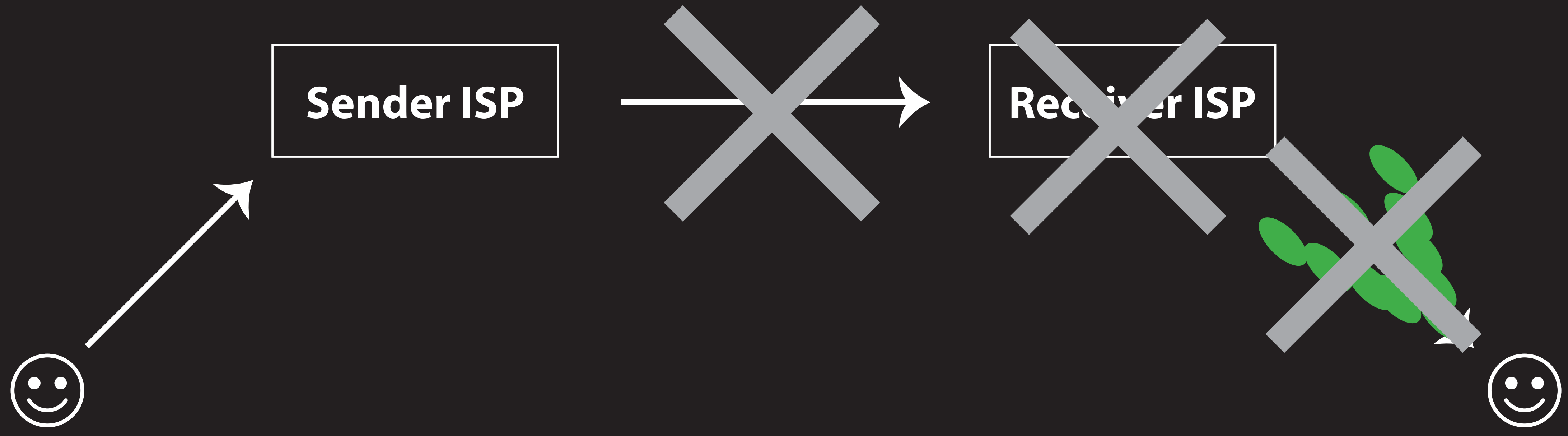
mobile computing => disconnected

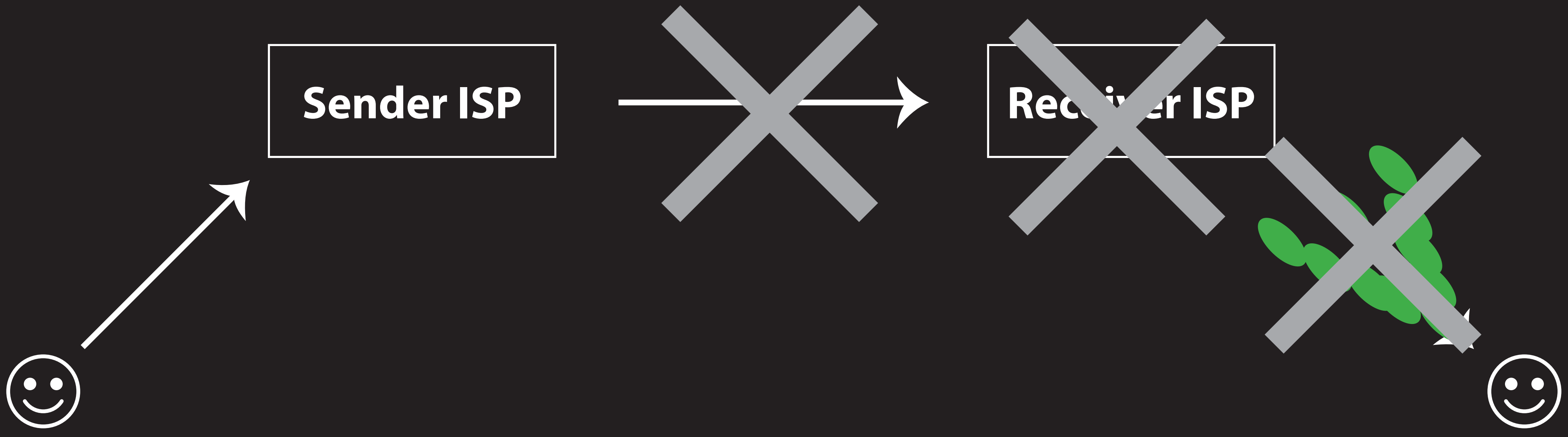


but!

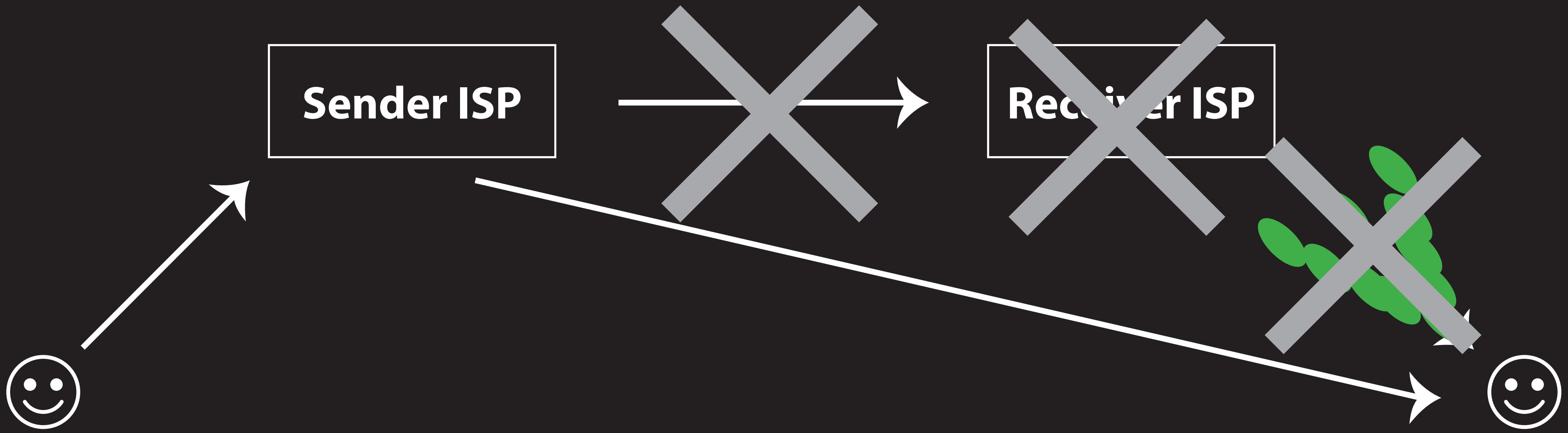
mobile computing => disconnected

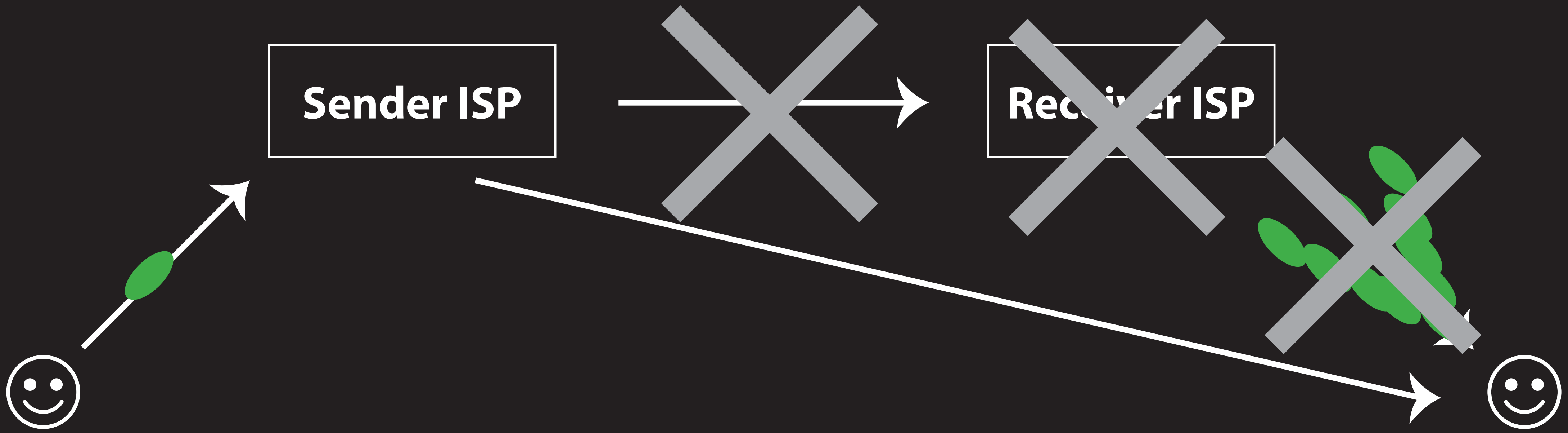
still need asynchronous mode

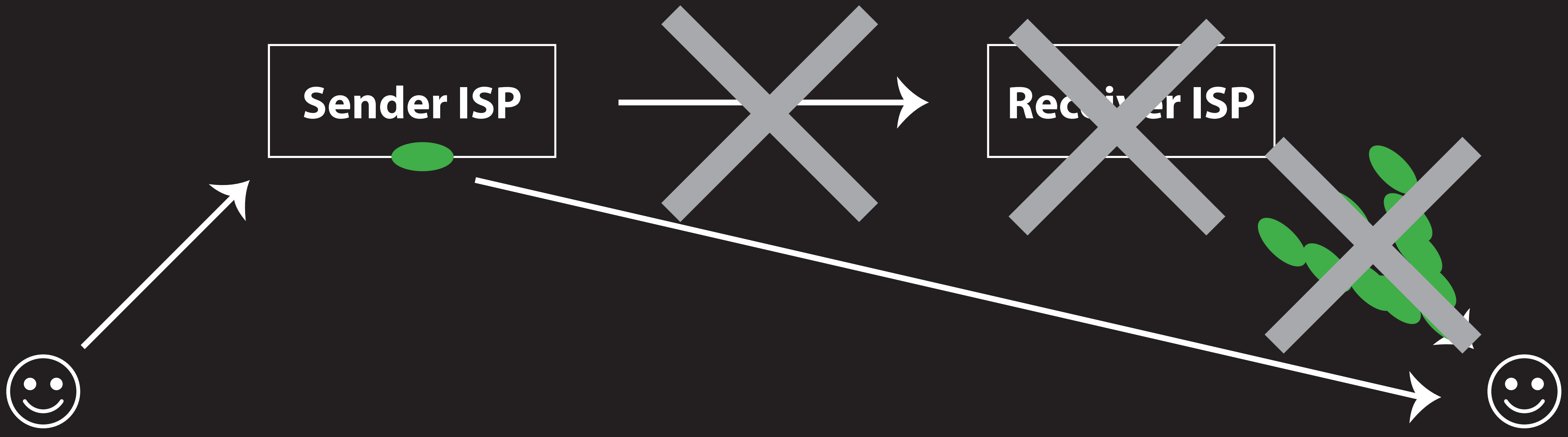

Sender ISP
Receiver ISP

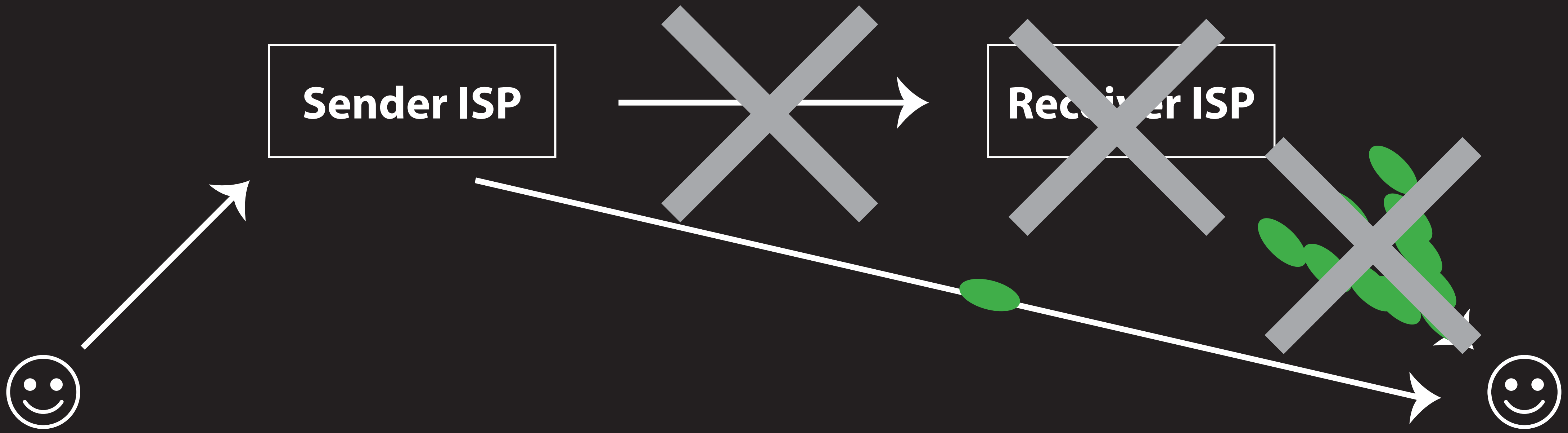


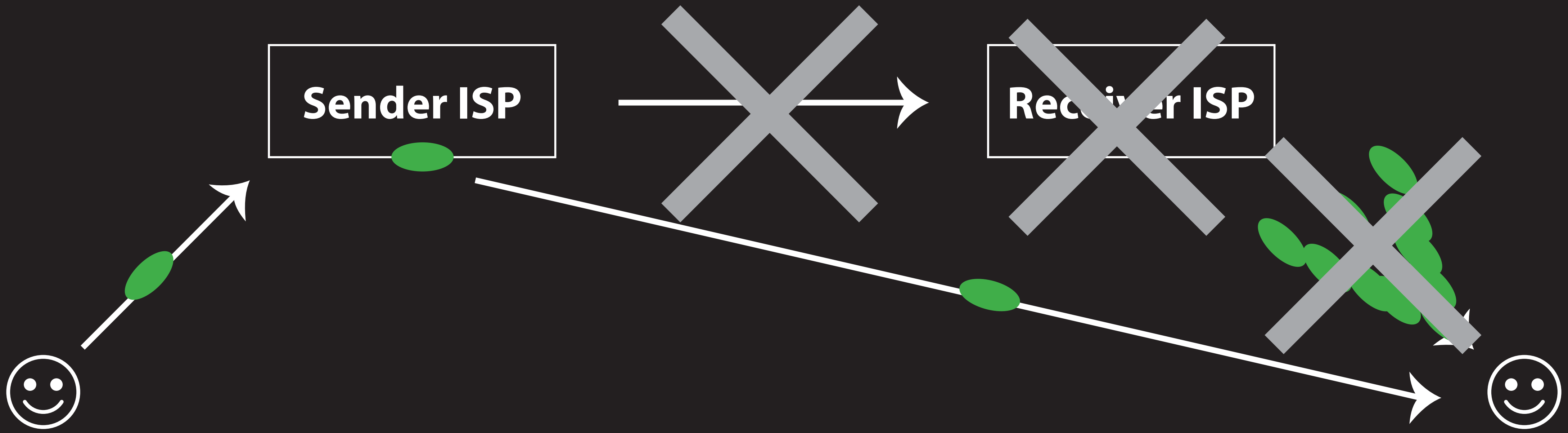
Sender ISP = permanently connected host


Sender ISP
Receiver ISP

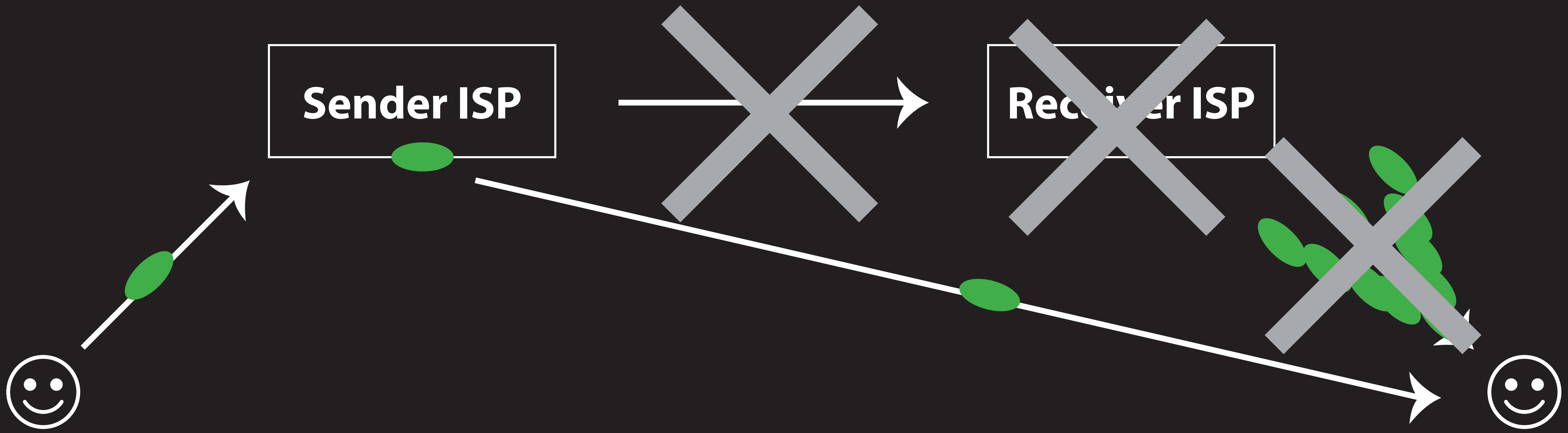

Sender ISP
Receiver ISP
☺
☺


Sender ISP
Receiver ISP


Sender ISP
Receiver ISP

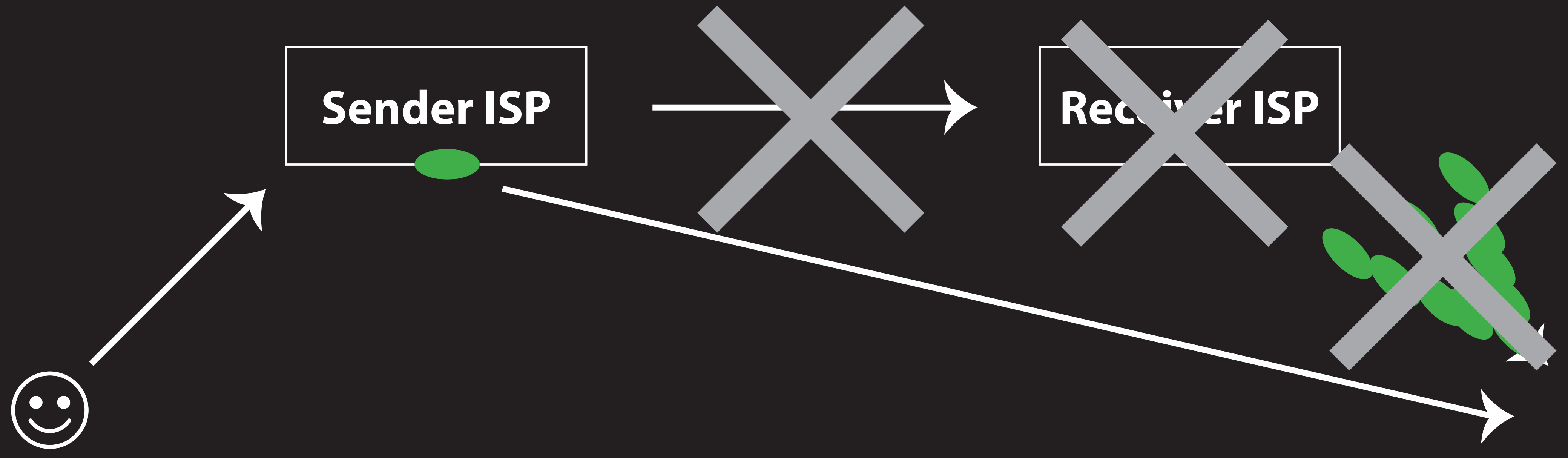


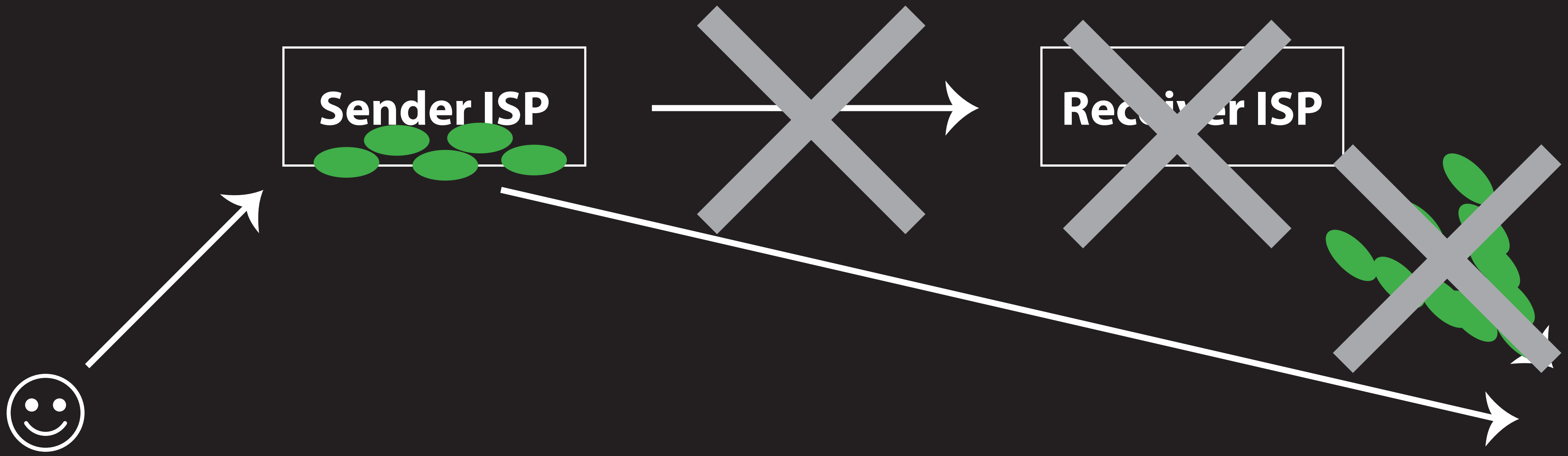
Ten years ago
Store and forward

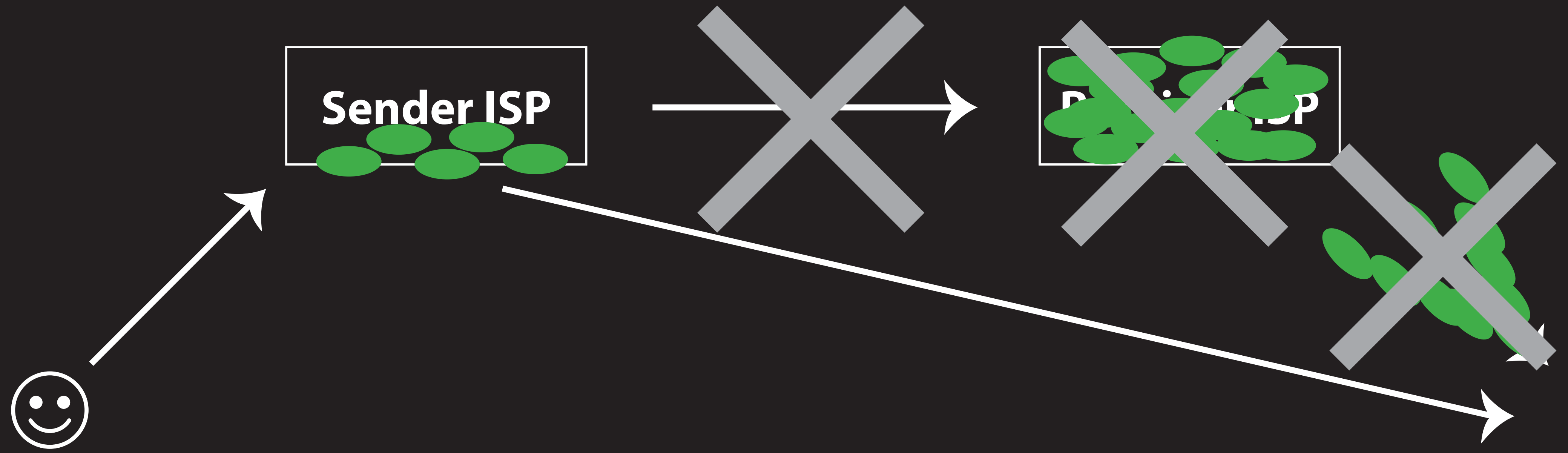


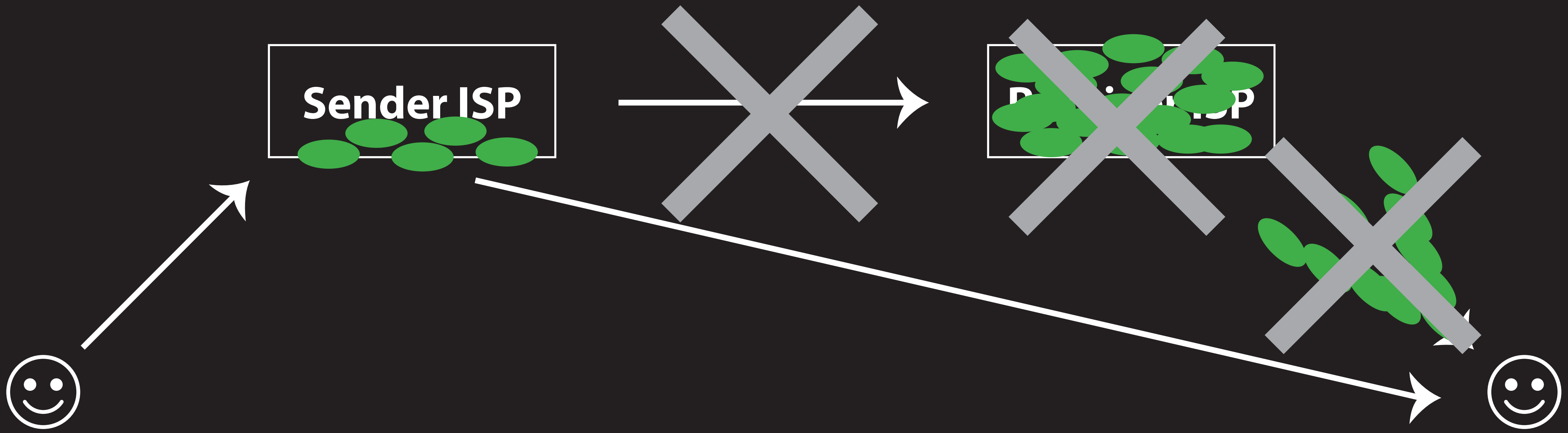
Ten years ago
Store and forward

Today
The Web
Blogging
RSS

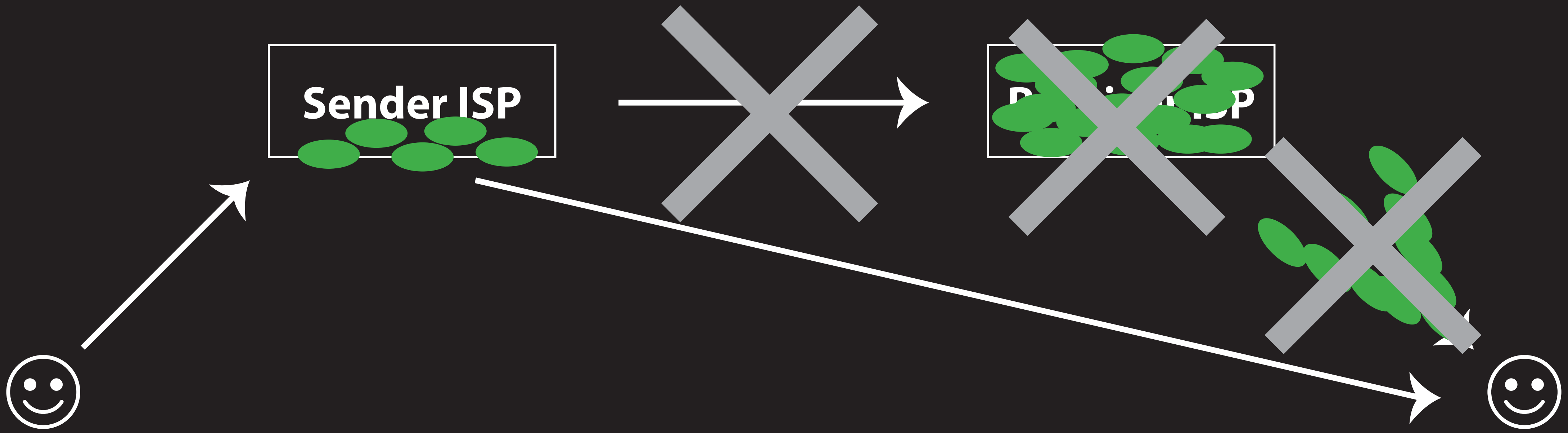

Sender ISP
Receiver ISP


Sender ISP
Receiver ISP


Sender ISP

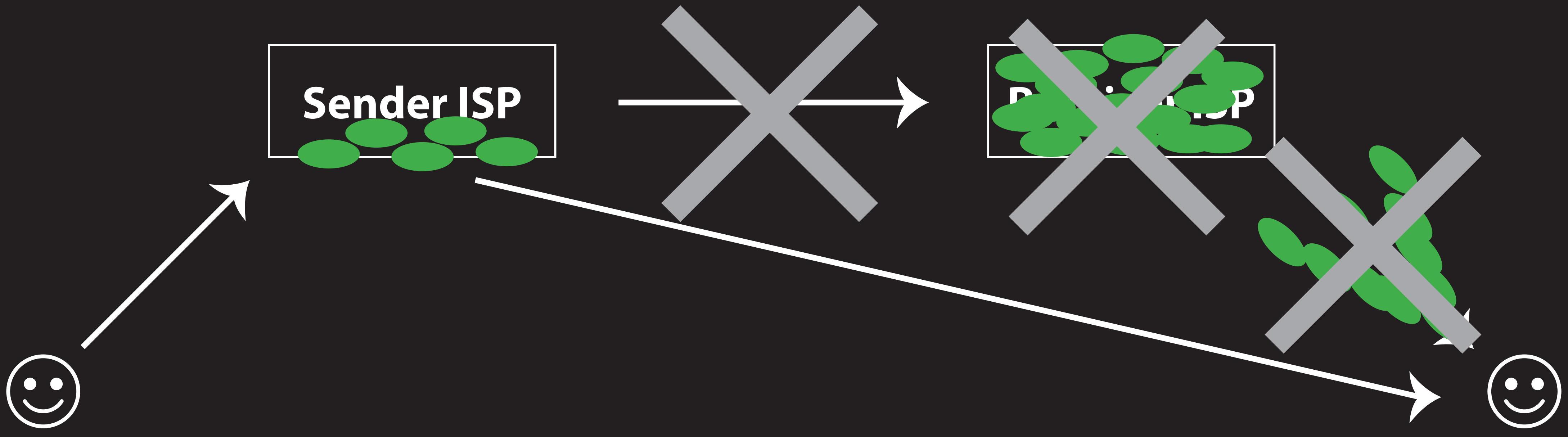


We could solve spam
if only we could shift the burden to the sender



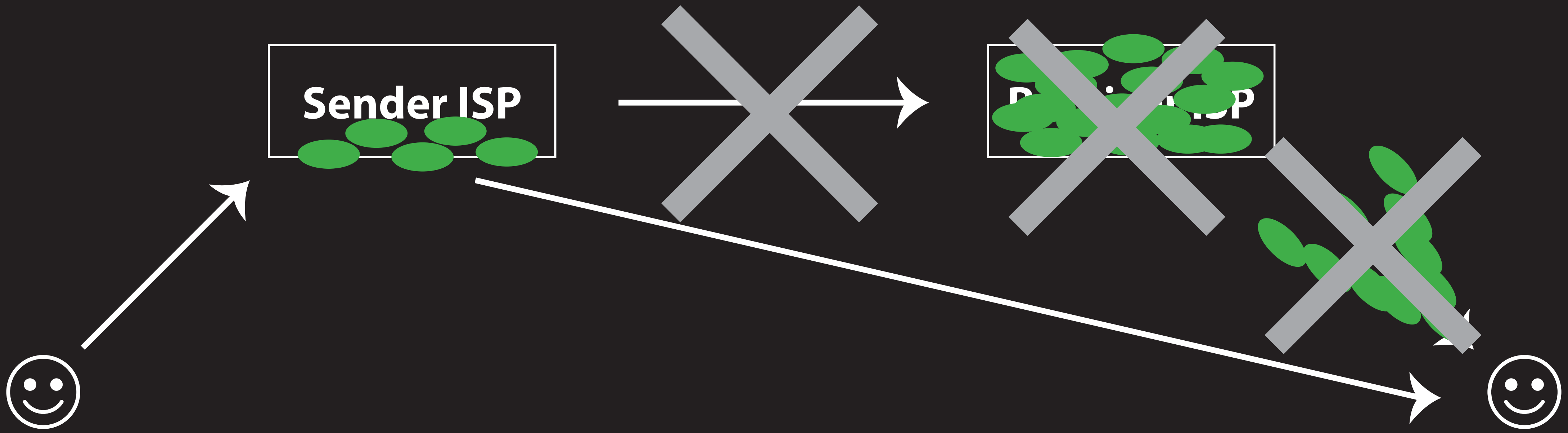
We could solve spam
if only we could shift the burden to the sender

e-Postage = $0.00001 per email



We could solve spam
if only we could shift the burden to the sender

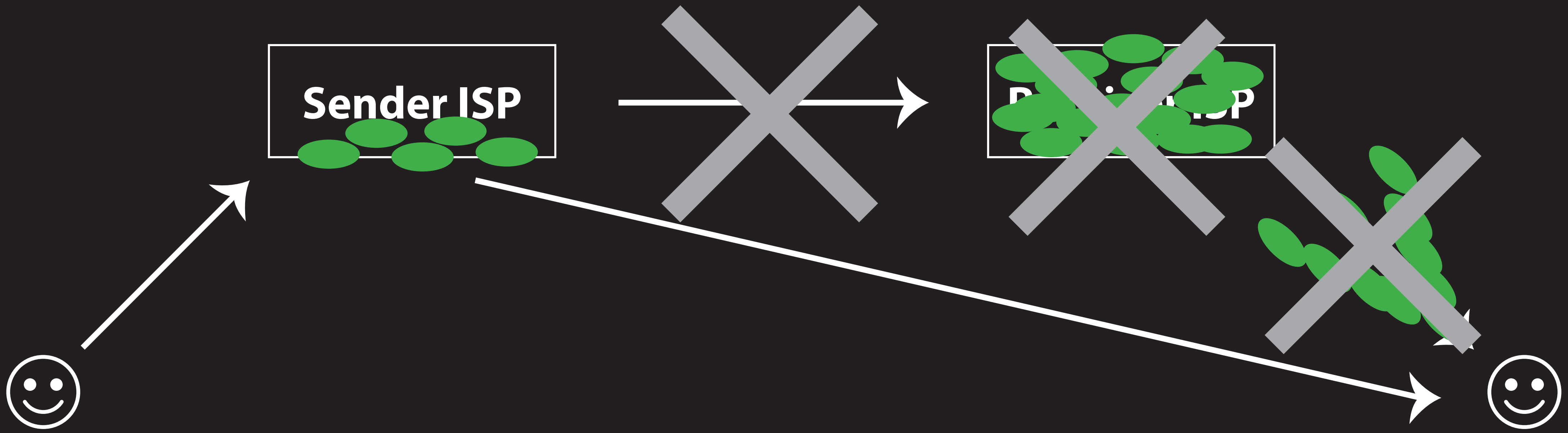
e-Postage = $0.00001 per email
“but what about Africa?”



We could solve spam
if only we could shift the burden to the sender

e-Postage = $0.00001 per email
“but what about Africa?”

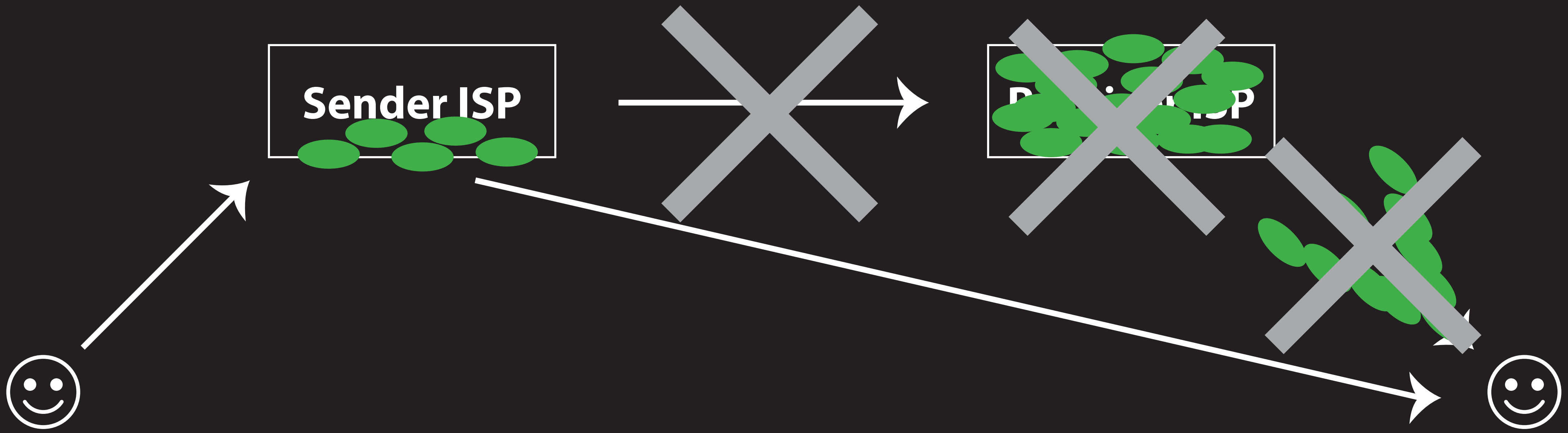
hashcash = make sender burn CPU



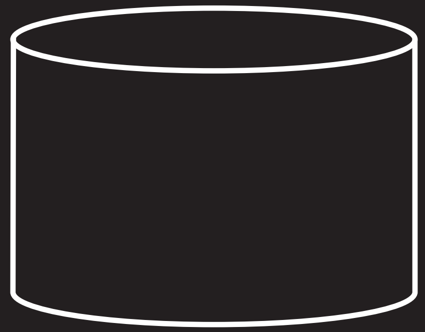
We could solve spam
if only we could shift the burden to the sender

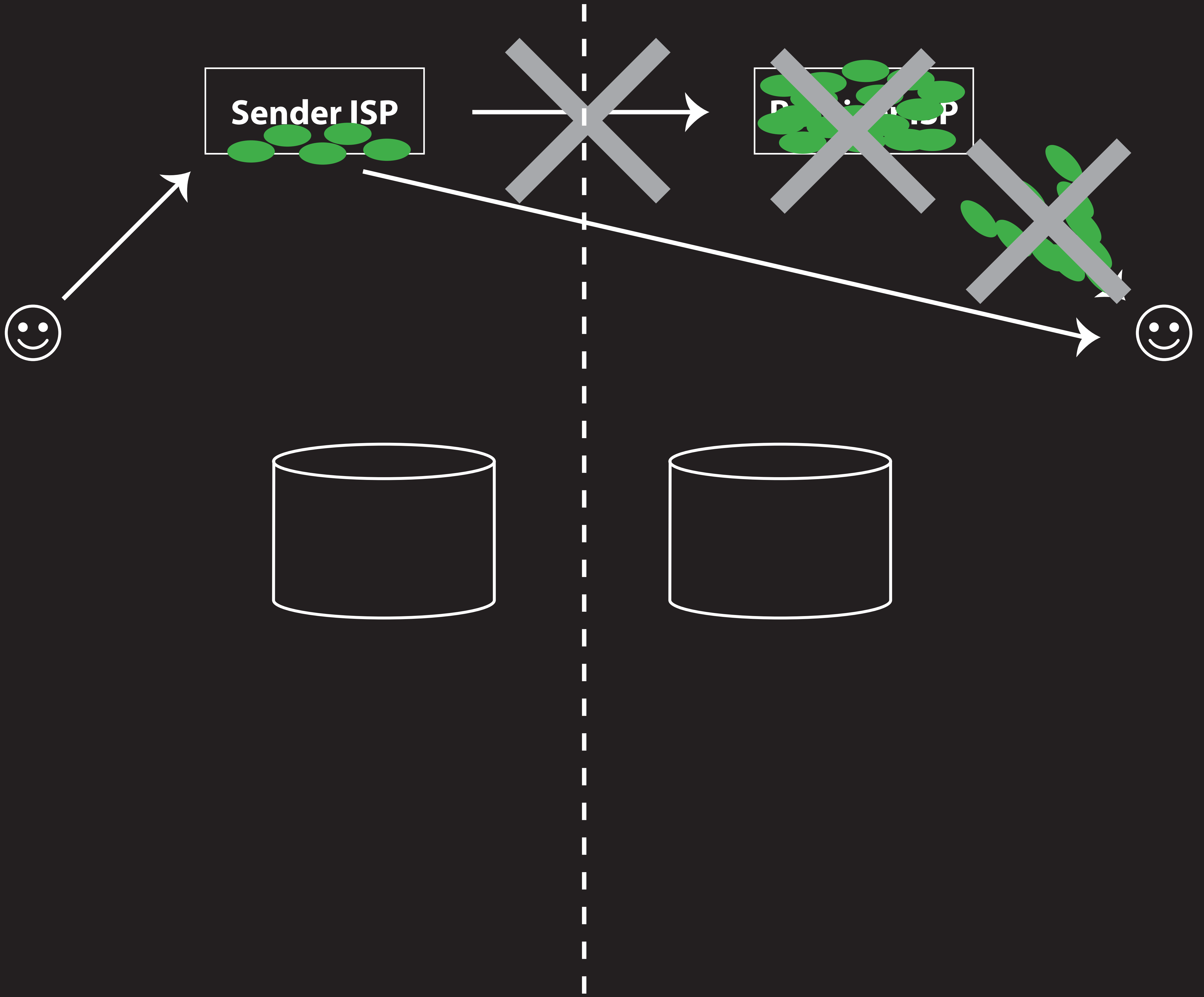
e-Postage = $0.00001 per email
“but what about Africa?”

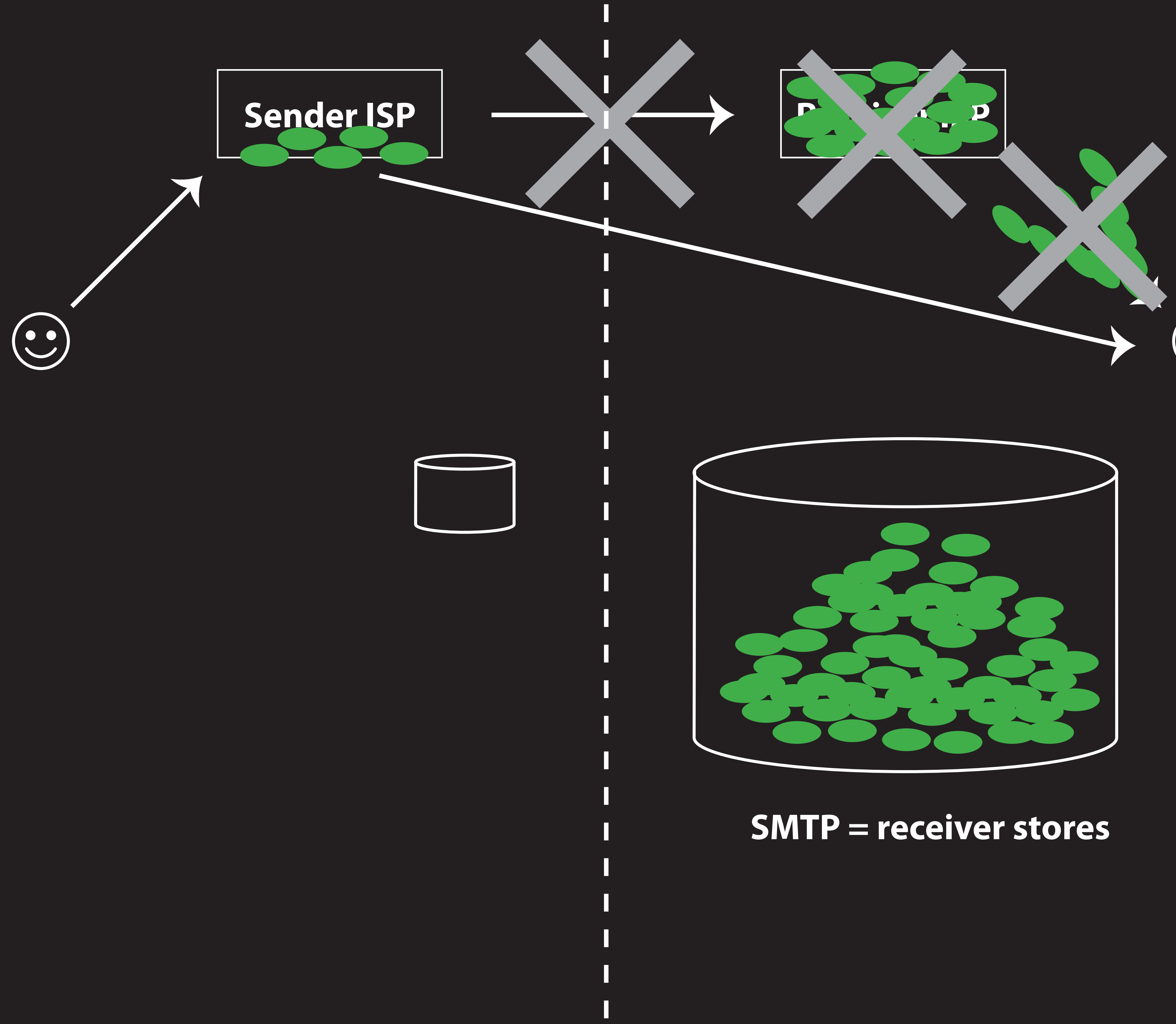
hashcash = make sender burn CPU
but zombies have more CPU than anyone



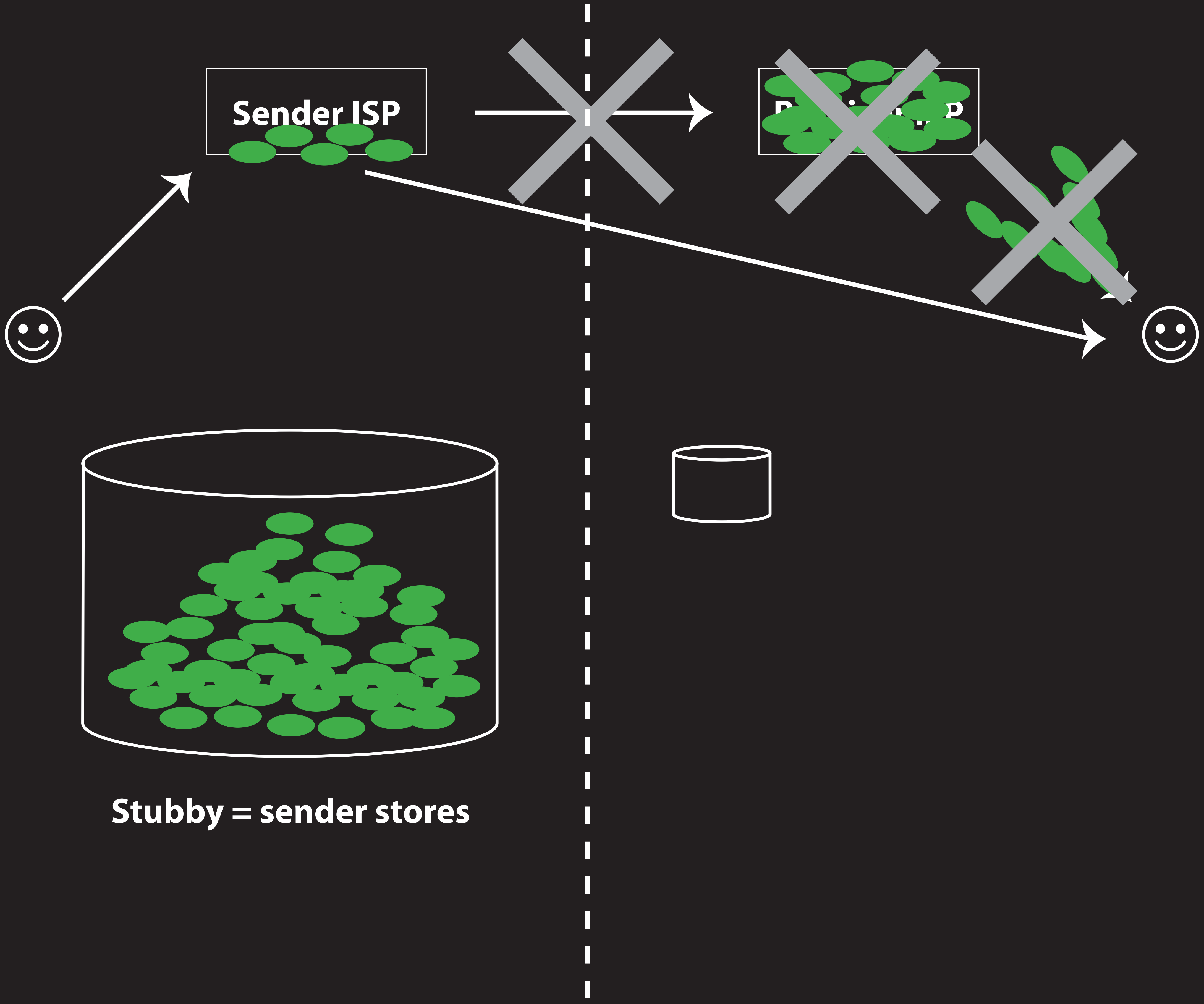
We could solve spam
if only we could shift the burden to the sender



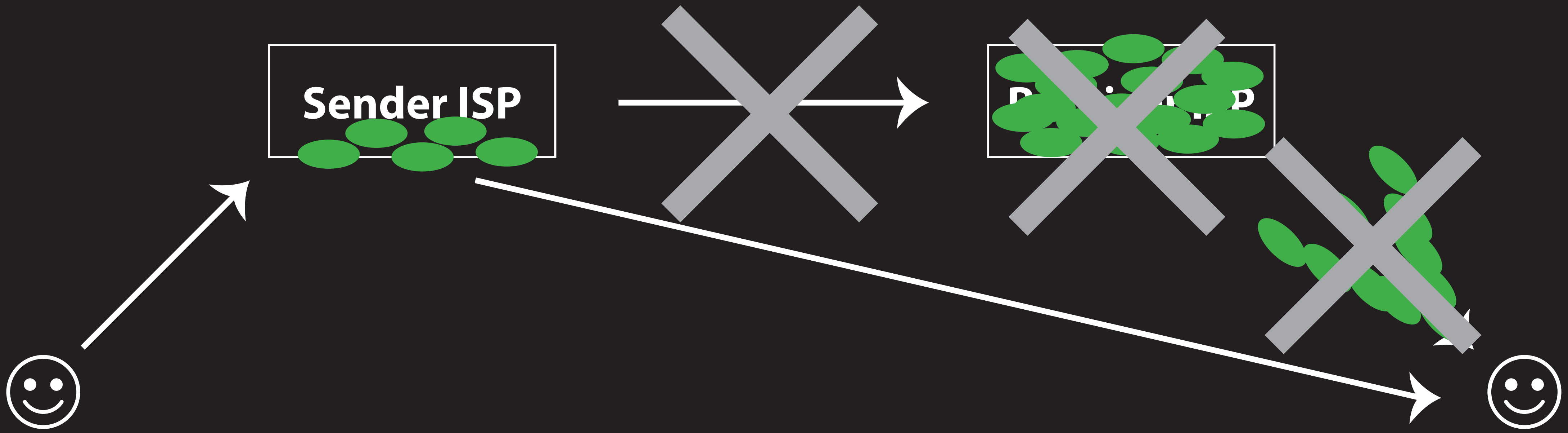

Sender ISP



SMTP = receiver stores

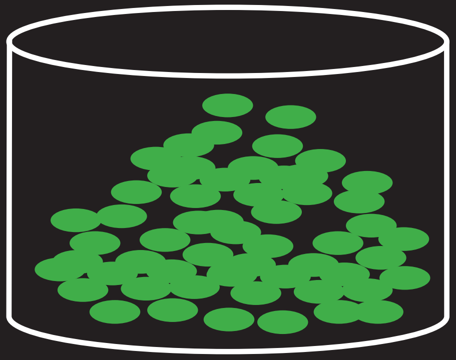


Stubby = sender stores

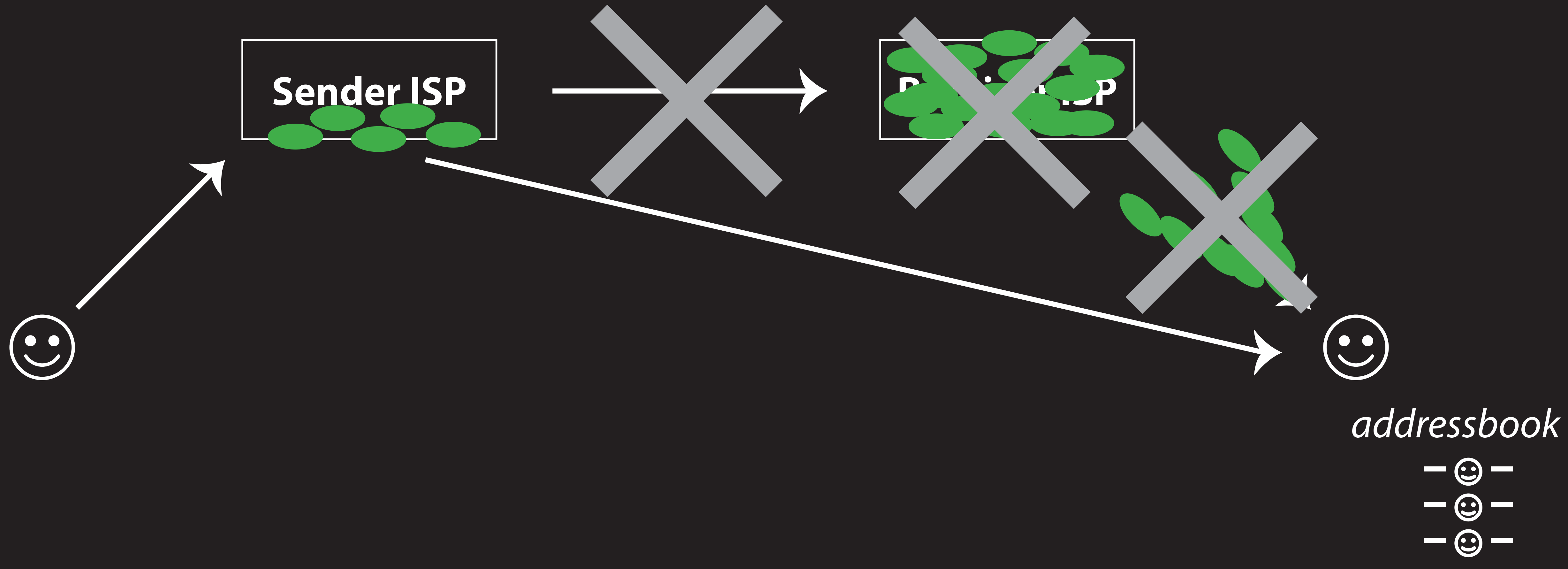


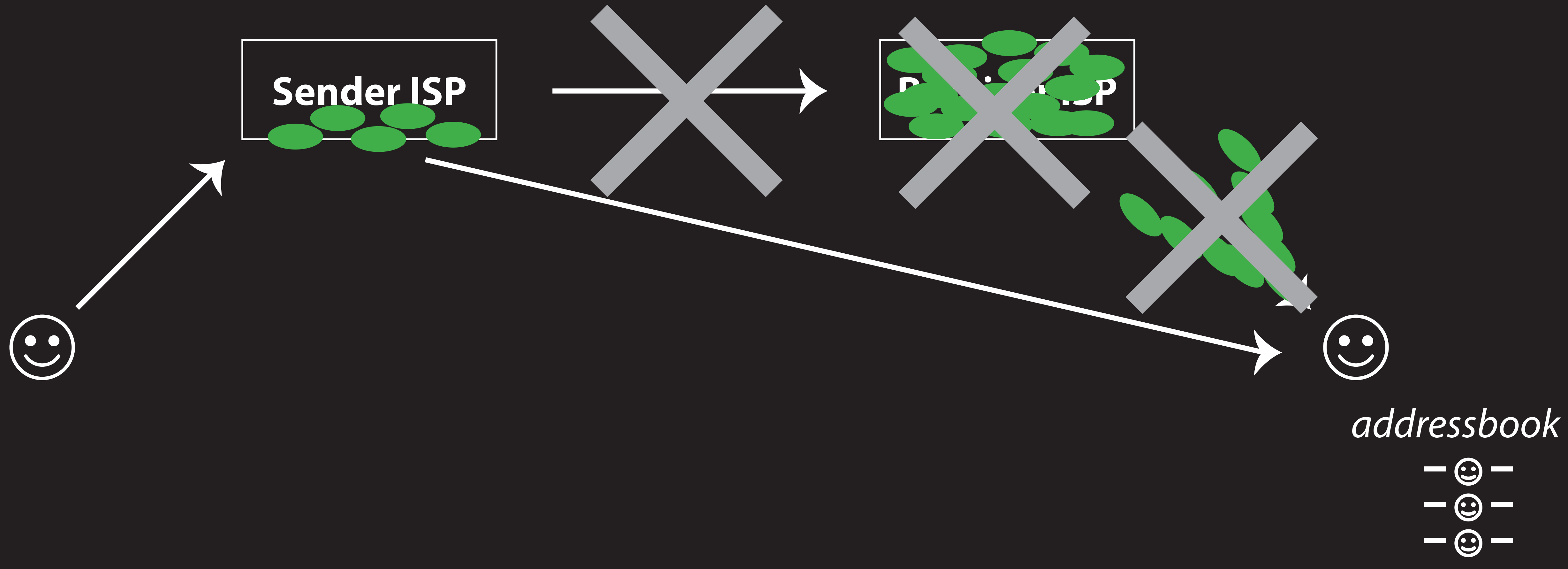
We could solve spam
if only we could shift the burden to the sender

disk + network
done!

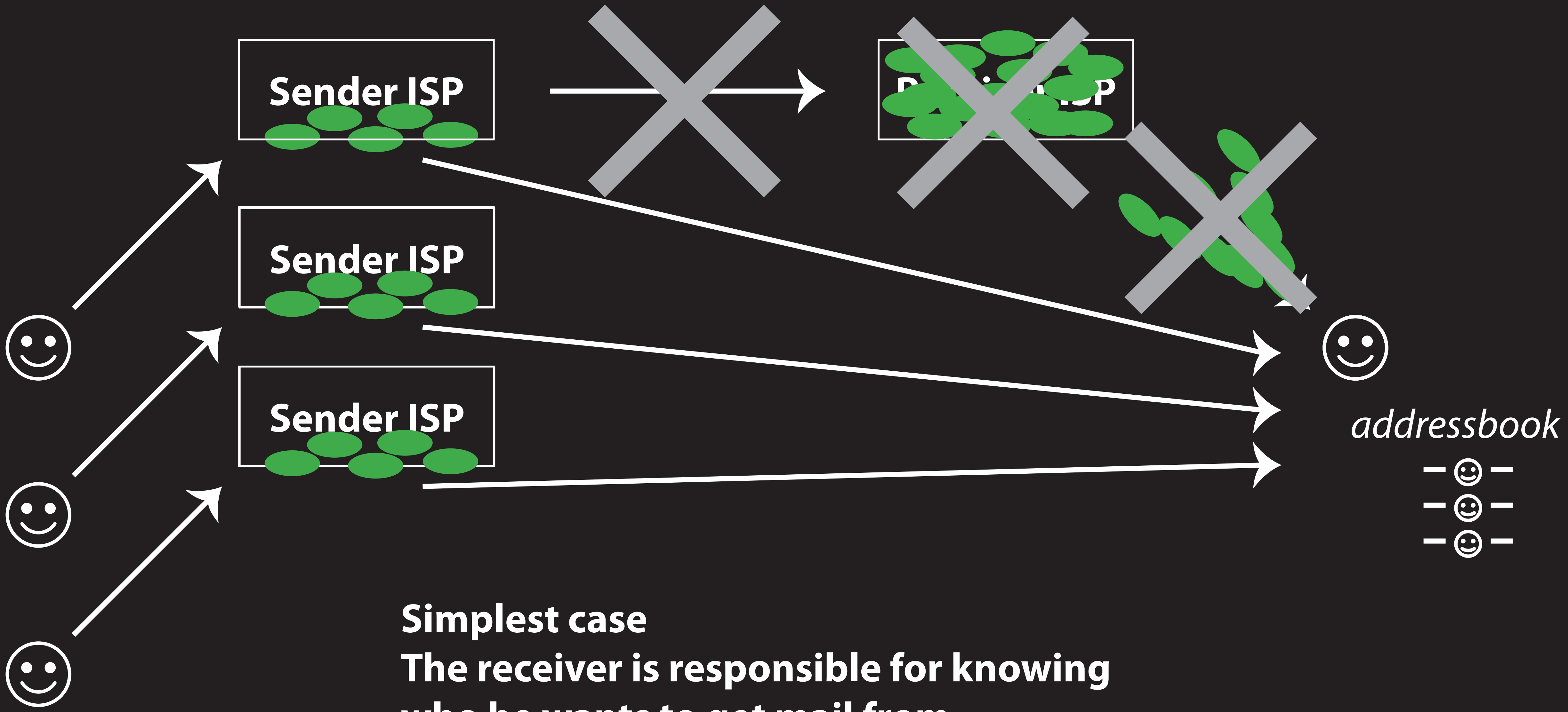


Stubby = sender stores

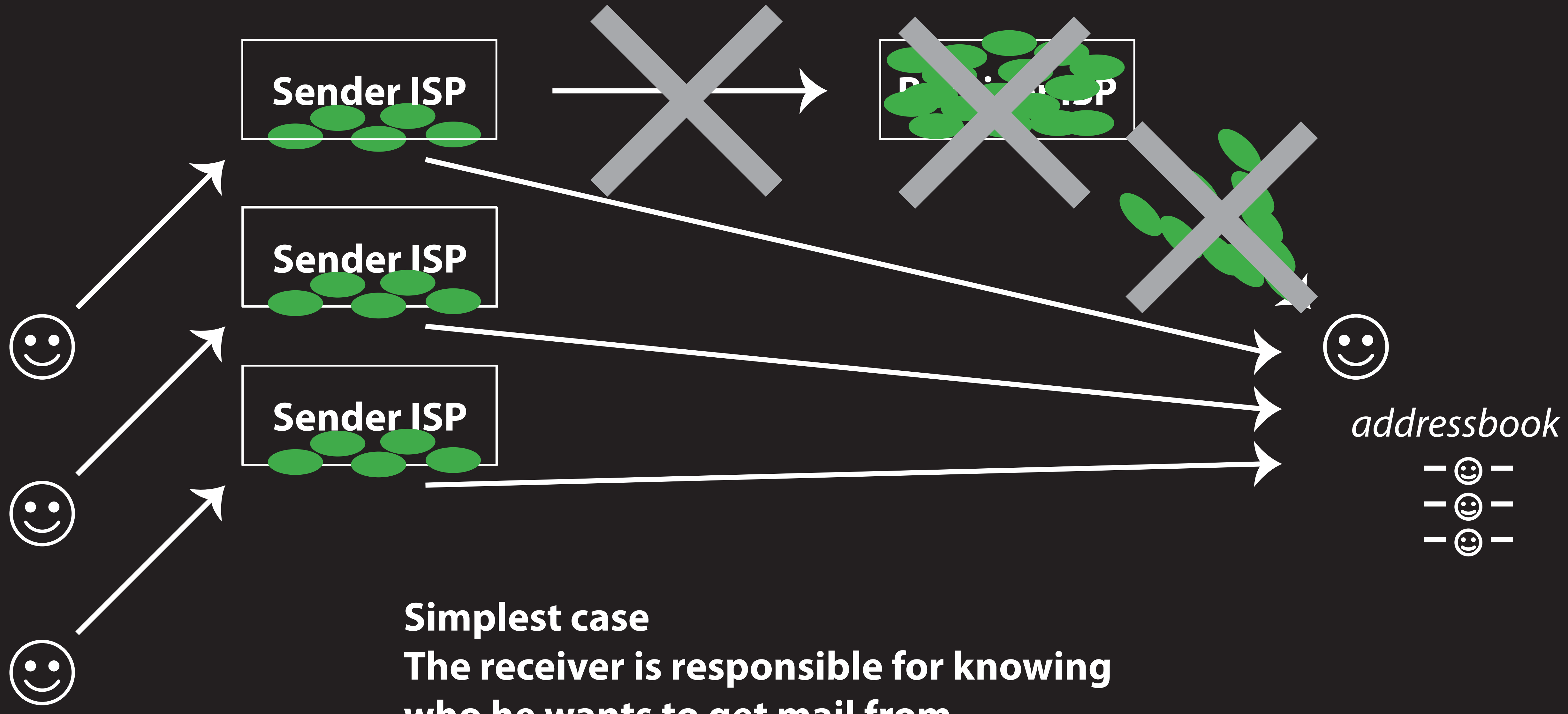

Sender ISP
Recipient ISP
addressbook



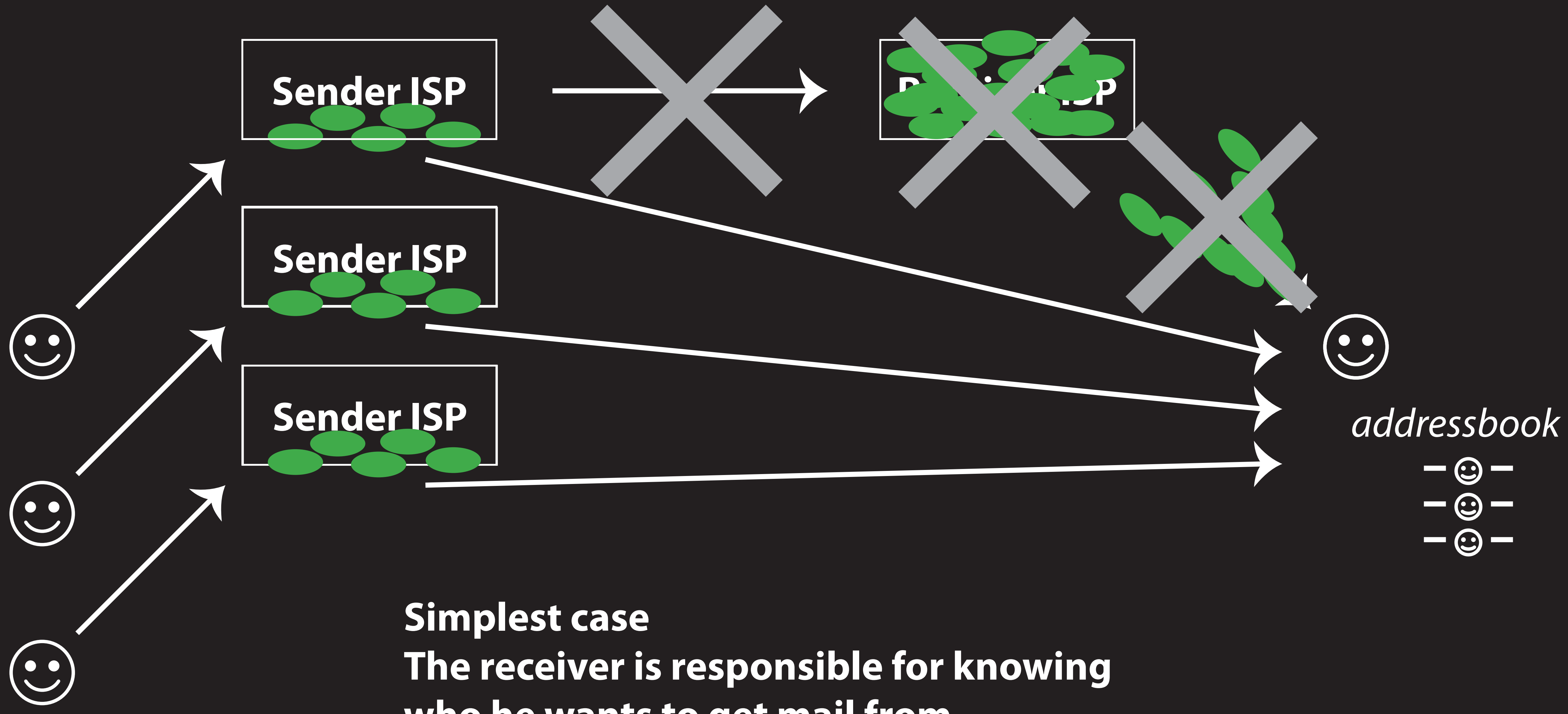
just like ~/.rss/subscribed-blogs



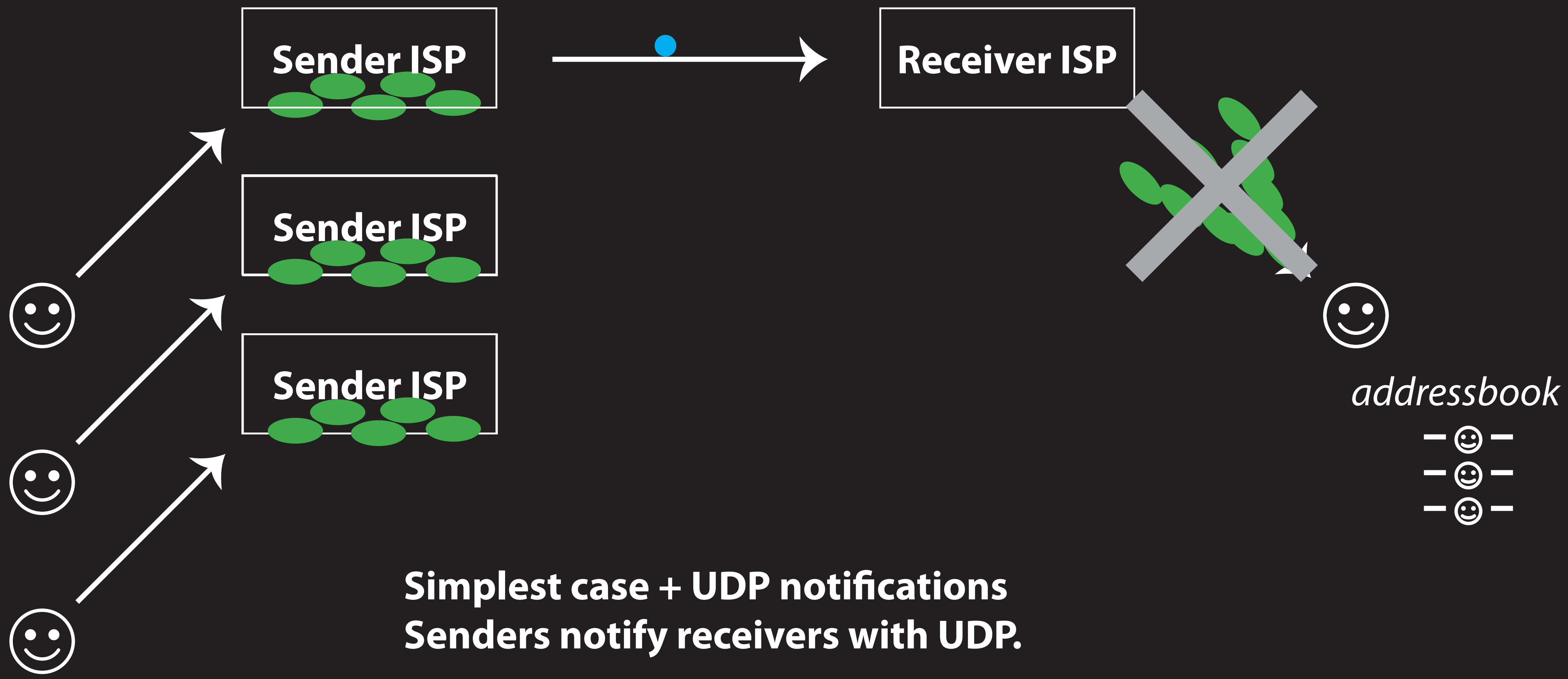
Simplest case
The receiver is responsible for knowing who he wants to get mail from.



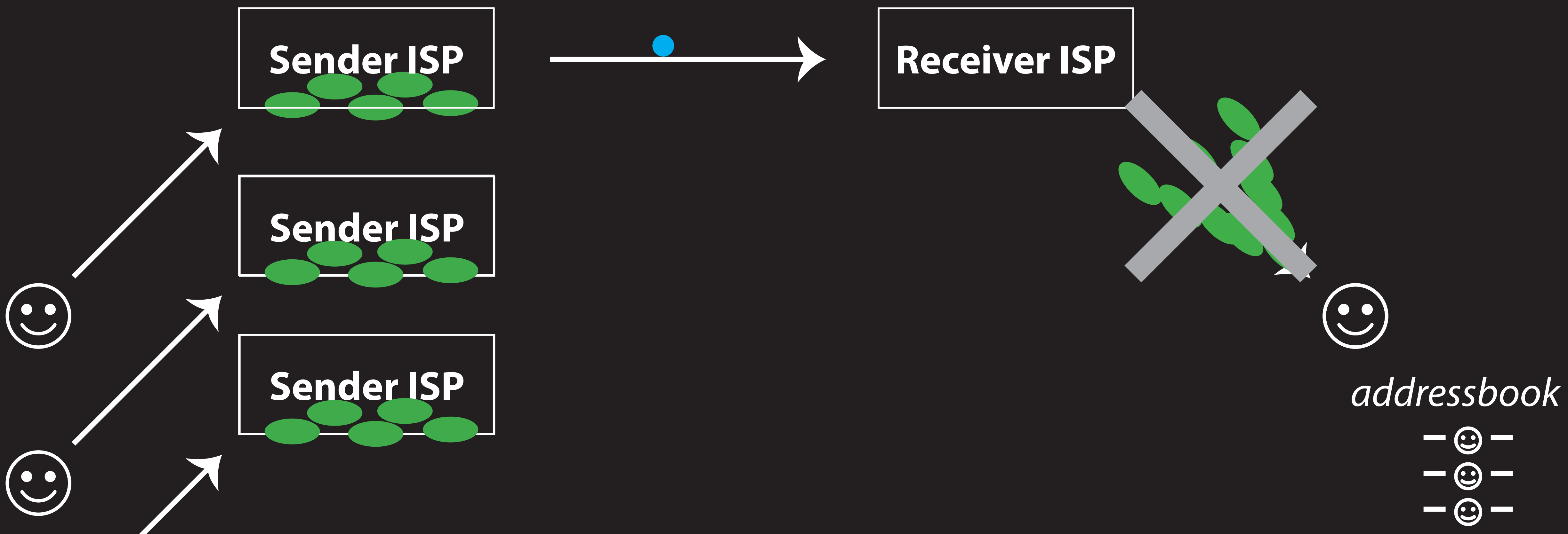
Simplest case
The receiver is responsible for knowing who he wants to get mail from.
The receiver has to keep polling.



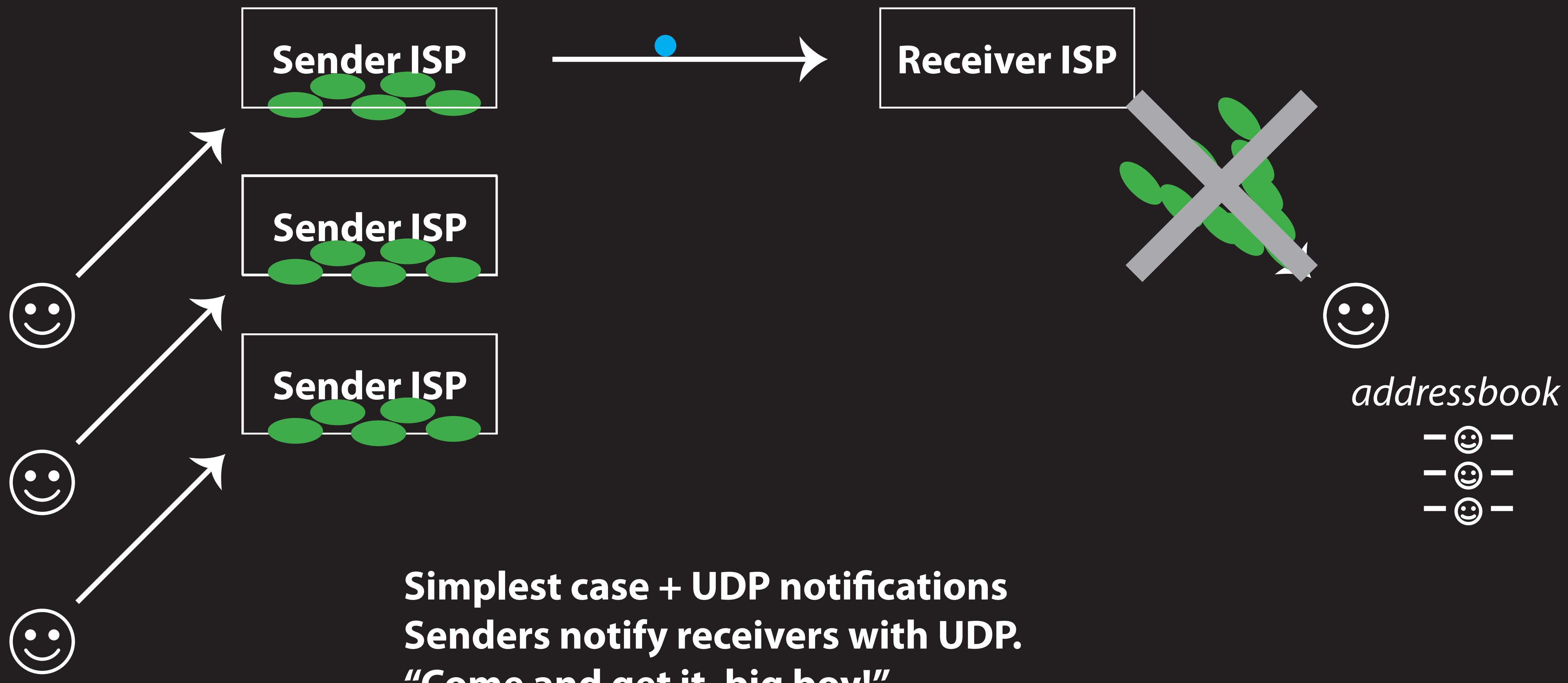
Simplest case
The receiver is responsible for knowing who he wants to get mail from.
The receiver has to keep polling.
“But polling is inefficient! Lag!”



Simplest case + UDP notifications
Senders notify receivers with UDP.

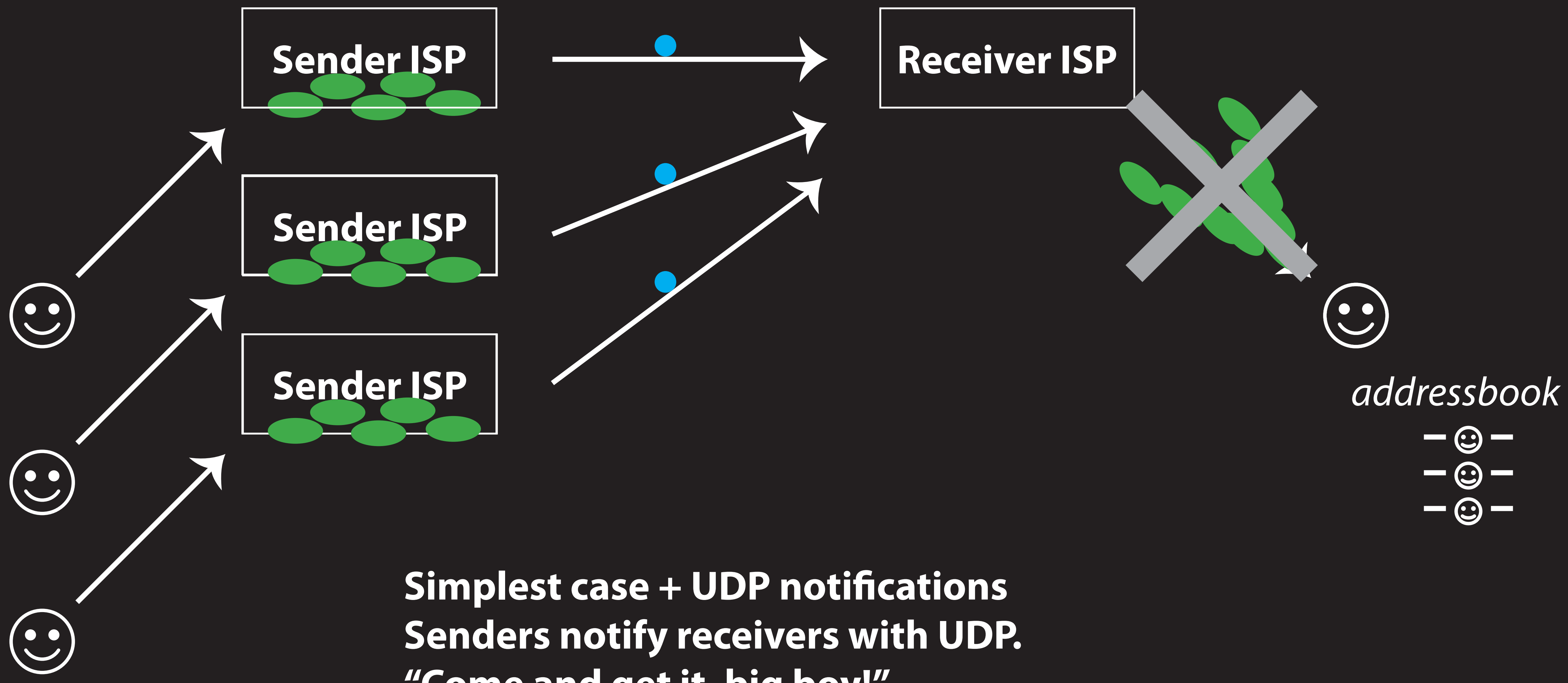


Simplest case + UDP notifications
Senders notify receivers with UDP.
“Come and get it, big boy!”



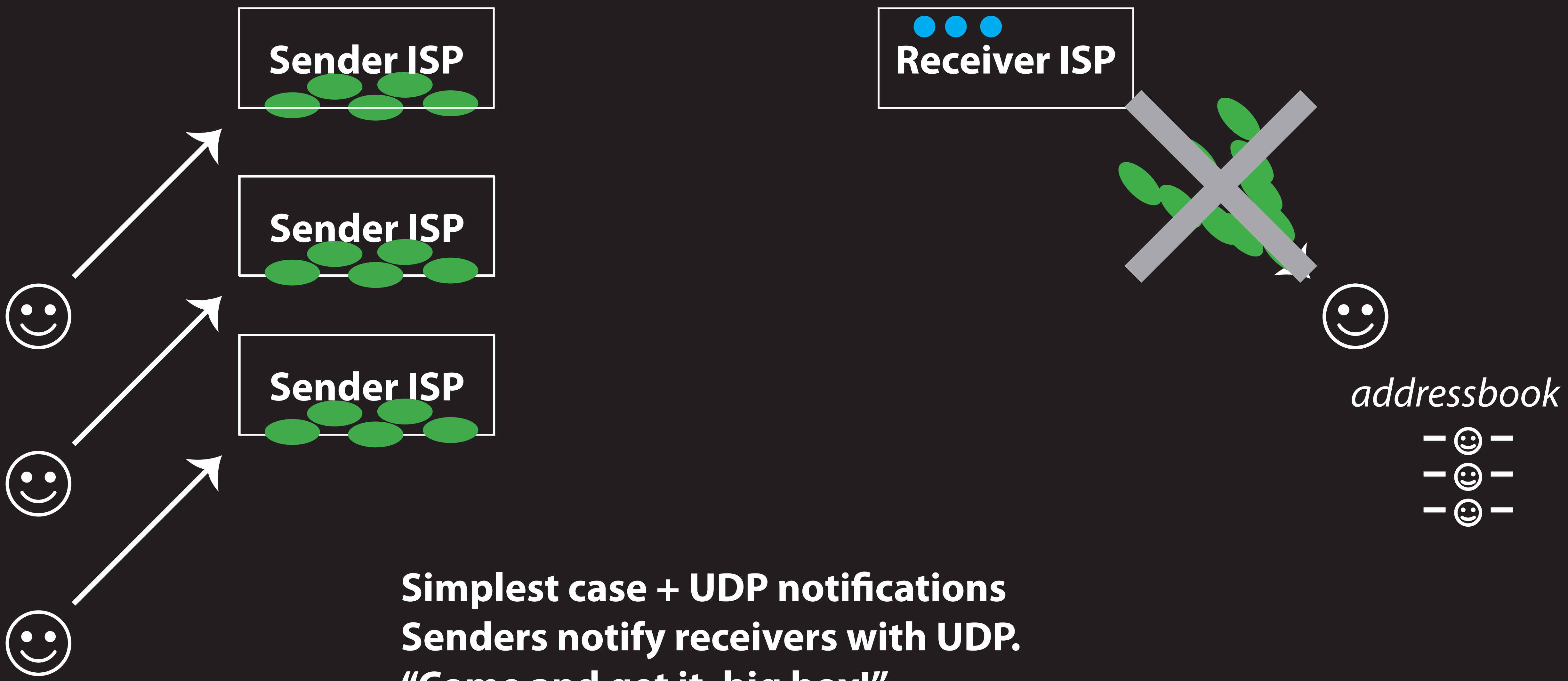
Simplest case + UDP notifications
Senders notify receivers with UDP.
"Come and get it, big boy!"

mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/



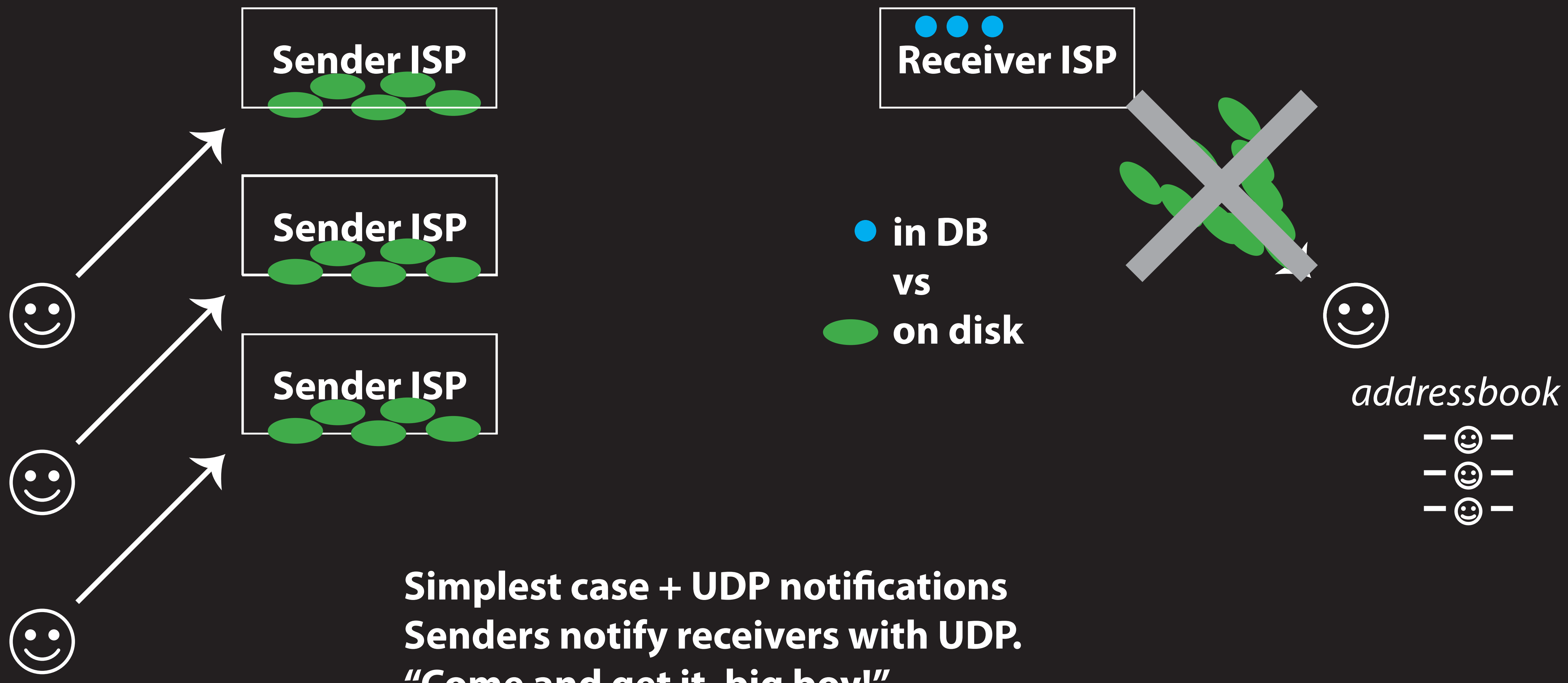
Simplest case + UDP notifications
Senders notify receivers with UDP.
“Come and get it, big boy!”

mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/



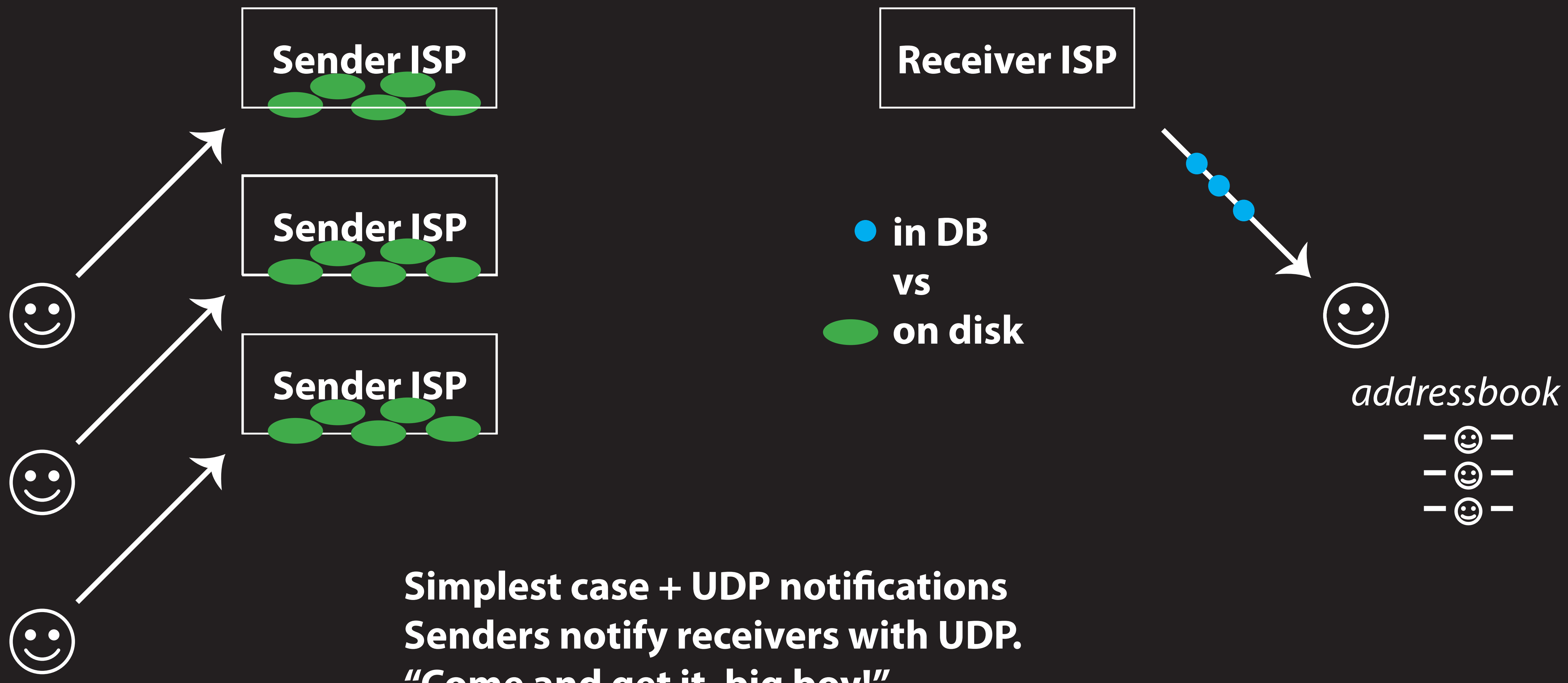
Simplest case + UDP notifications
Senders notify receivers with UDP.
"Come and get it, big boy!"

mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/



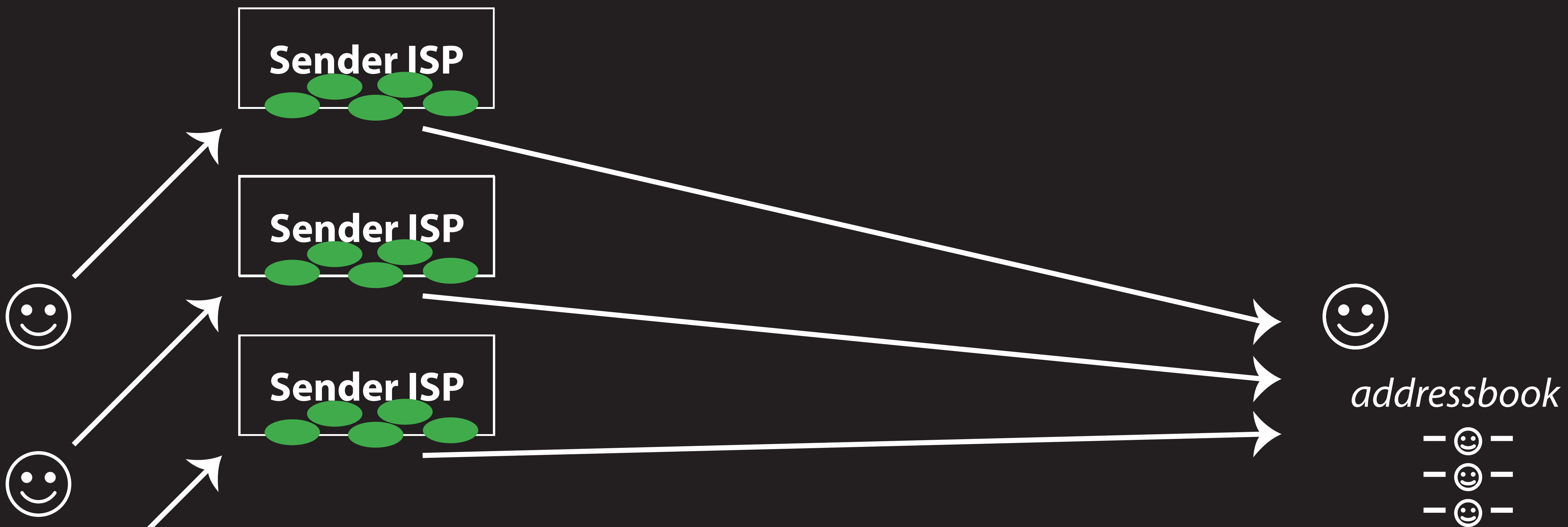
Simplest case + UDP notifications
Senders notify receivers with UDP.
“Come and get it, big boy!”

```
mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/
```



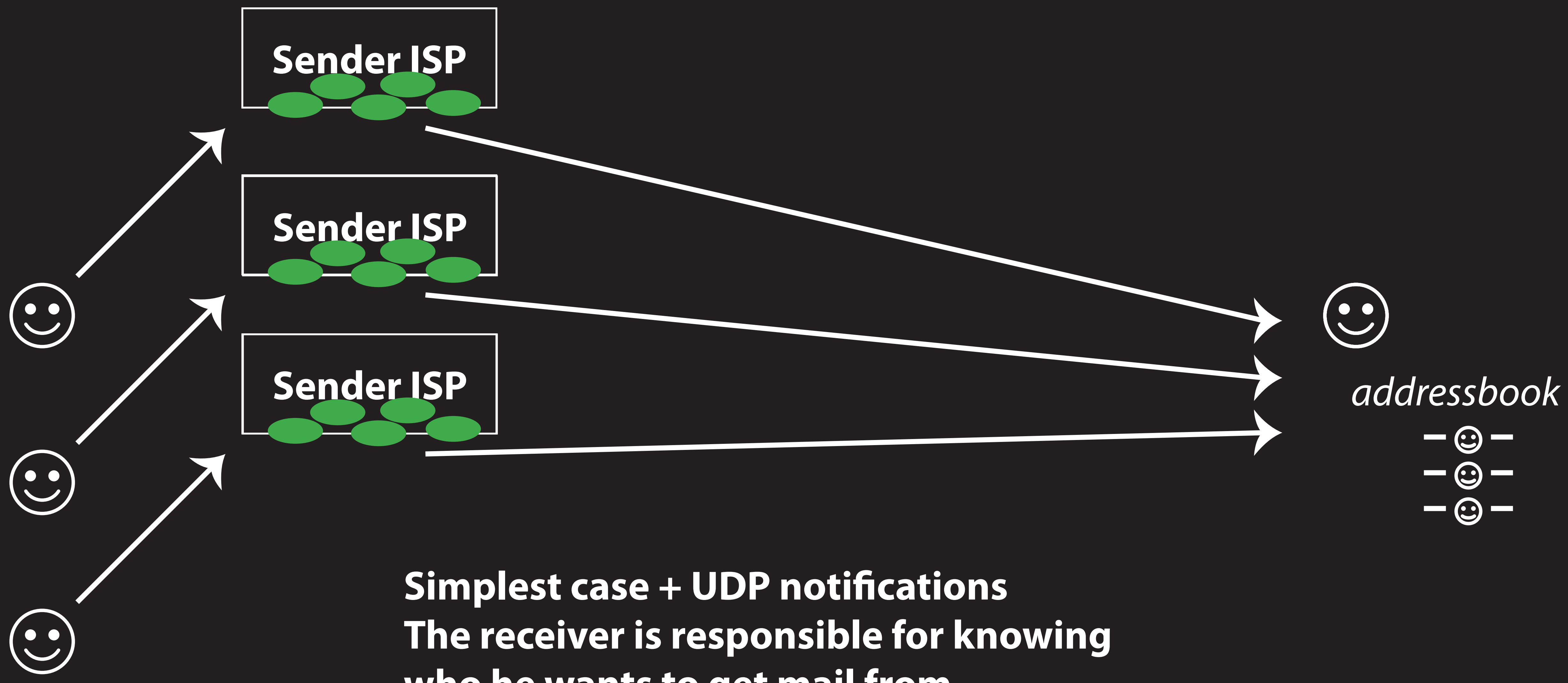
Simplest case + UDP notifications
Senders notify receivers with UDP.
“Come and get it, big boy!”

mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/

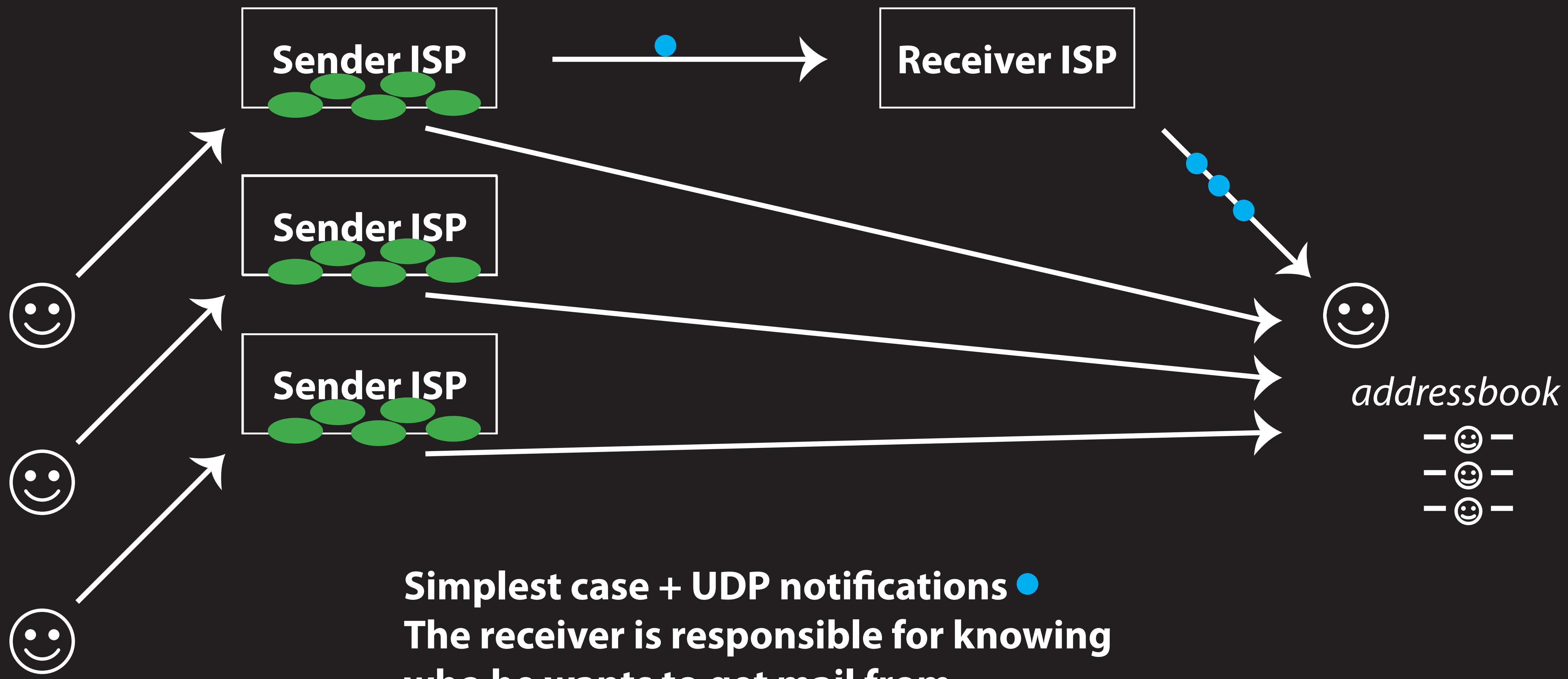


Simplest case + UDP notifications
Senders notify receivers with UDP.
“Come and get it, big boy!”

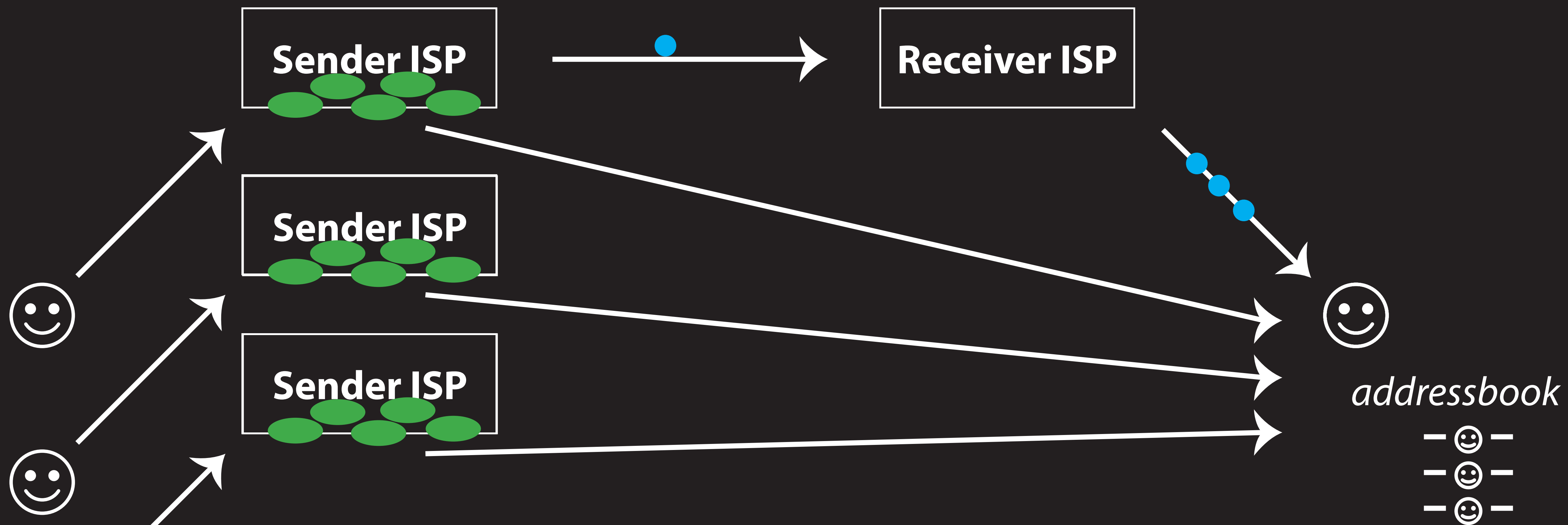
mail from: sender1@sender-isp.com
rcpt to: receiver1@receiver-isp.com
url: http://sender-isp.com/sender1/receiver1/



Simplest case + UDP notifications
The receiver is responsible for knowing who he wants to get mail from.
The receiver has to keep polling.
“But polling is inefficient! Lag!”

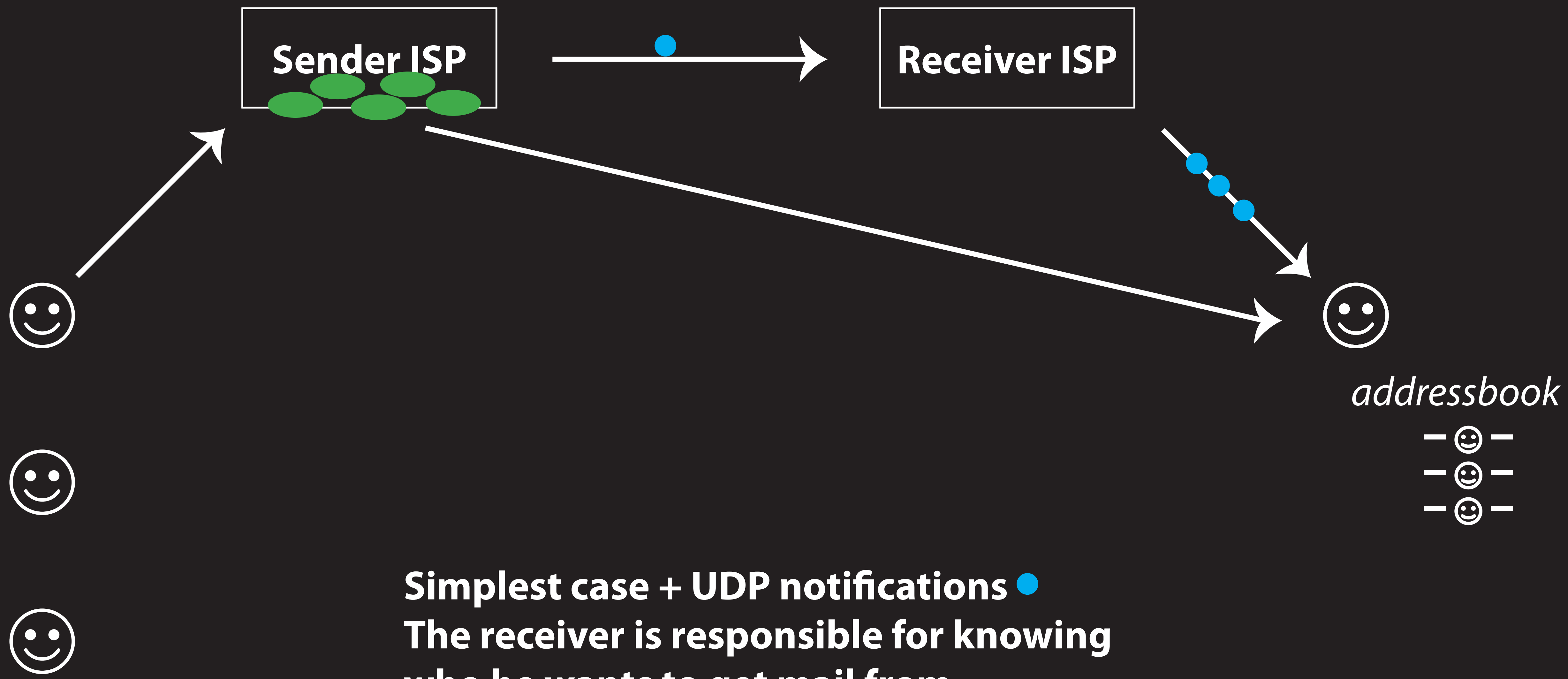


Simplest case + UDP notifications ●
The receiver is responsible for knowing who he wants to get mail from.
The receiver has to keep polling.
"But polling is inefficient! Lag!"

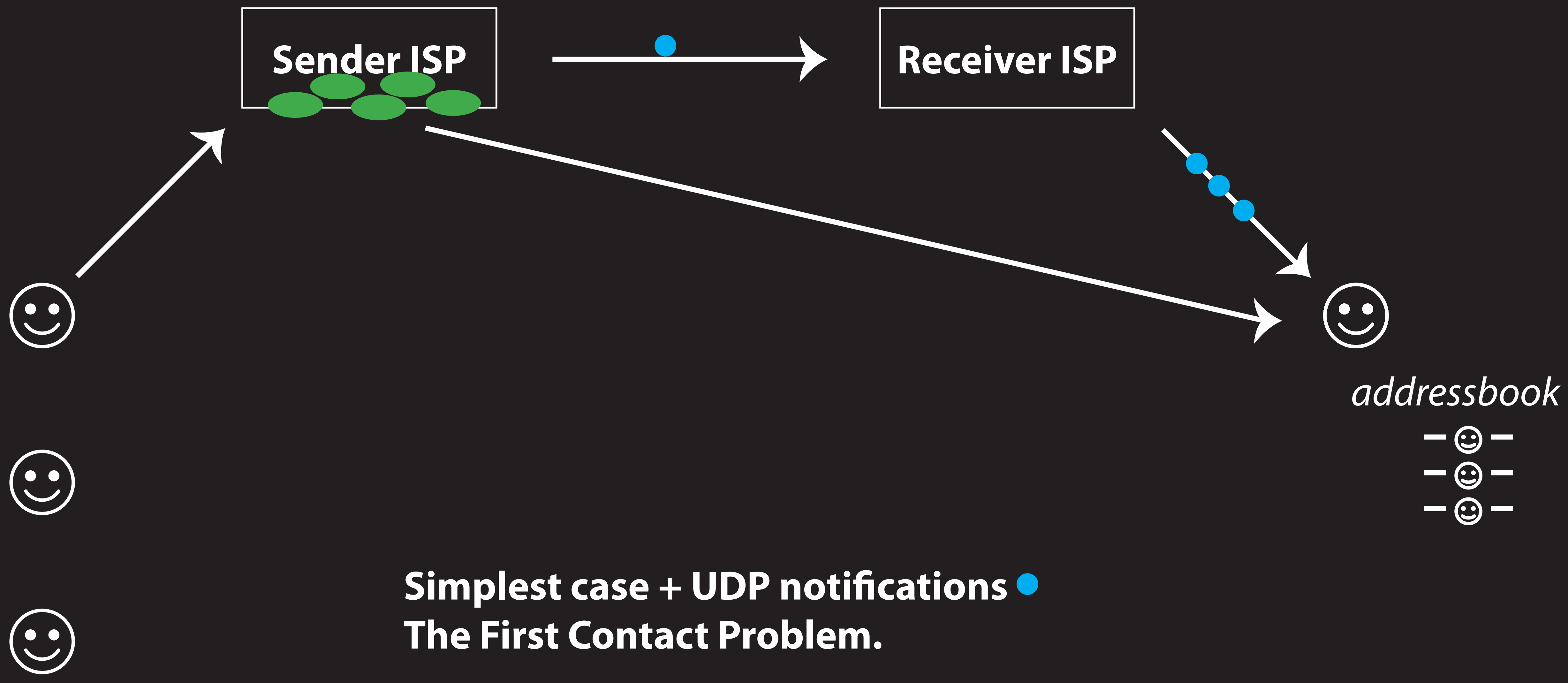


Simplest case + UDP notifications ●
The receiver is responsible for knowing who he wants to get mail from.

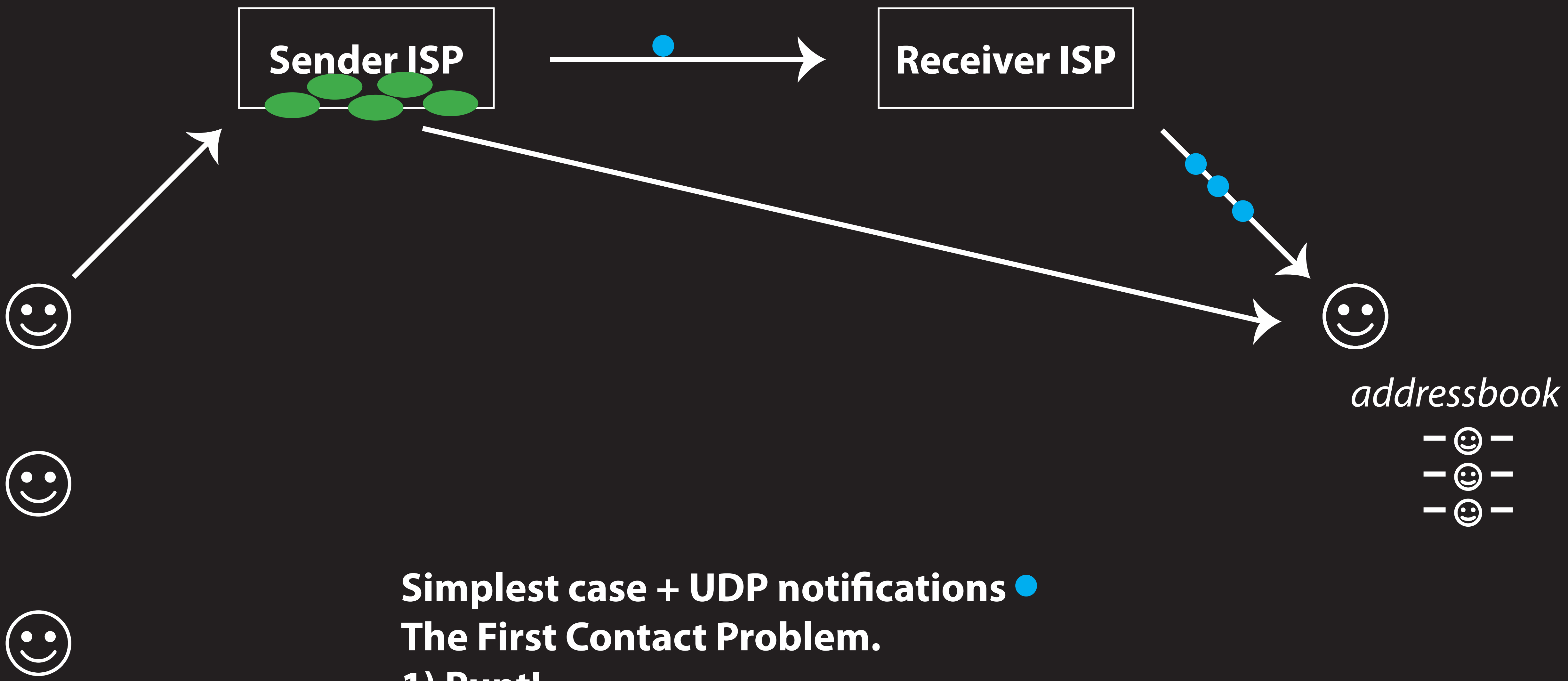
● **~~The receiver has to keep polling.~~**
~~"But polling is inefficient! Lag!"~~



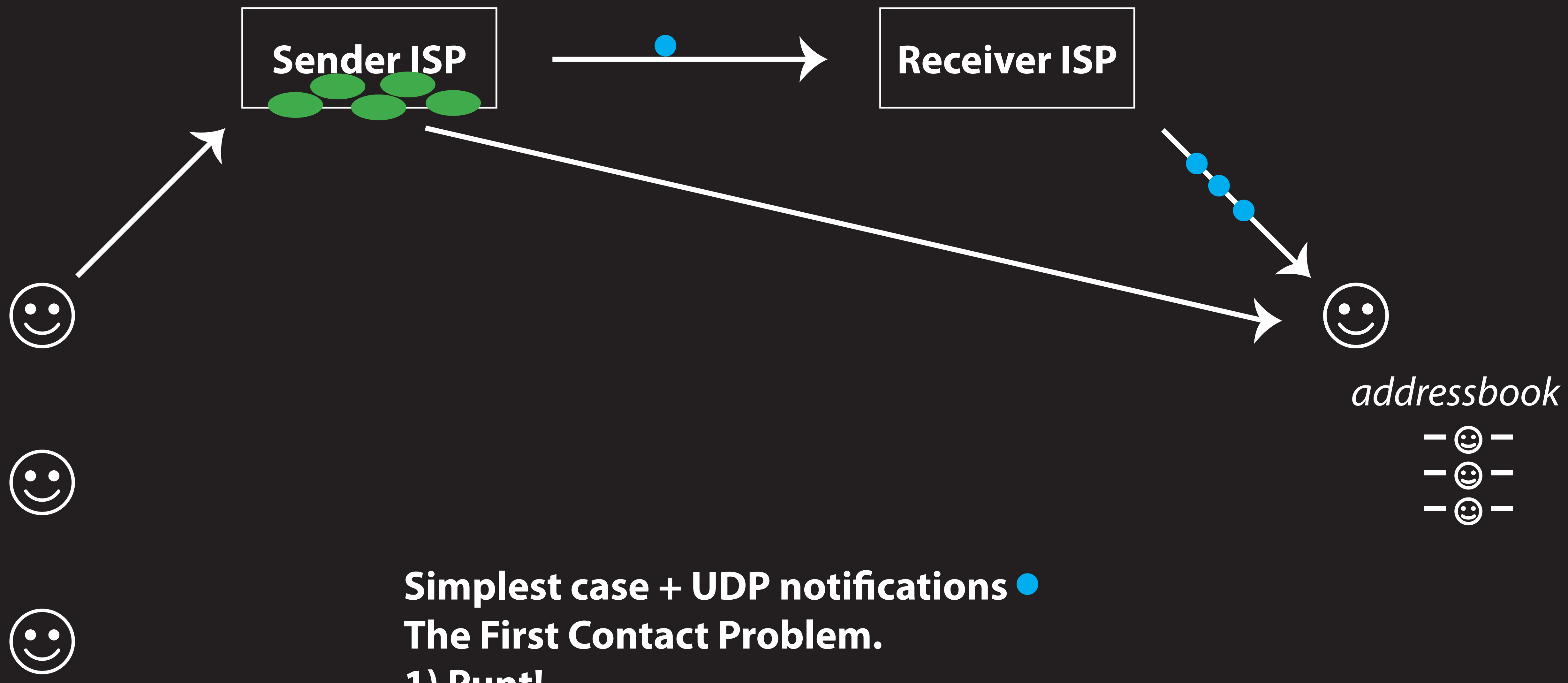
Simplest case + UDP notifications ●
The receiver is responsible for knowing who he wants to get mail from.



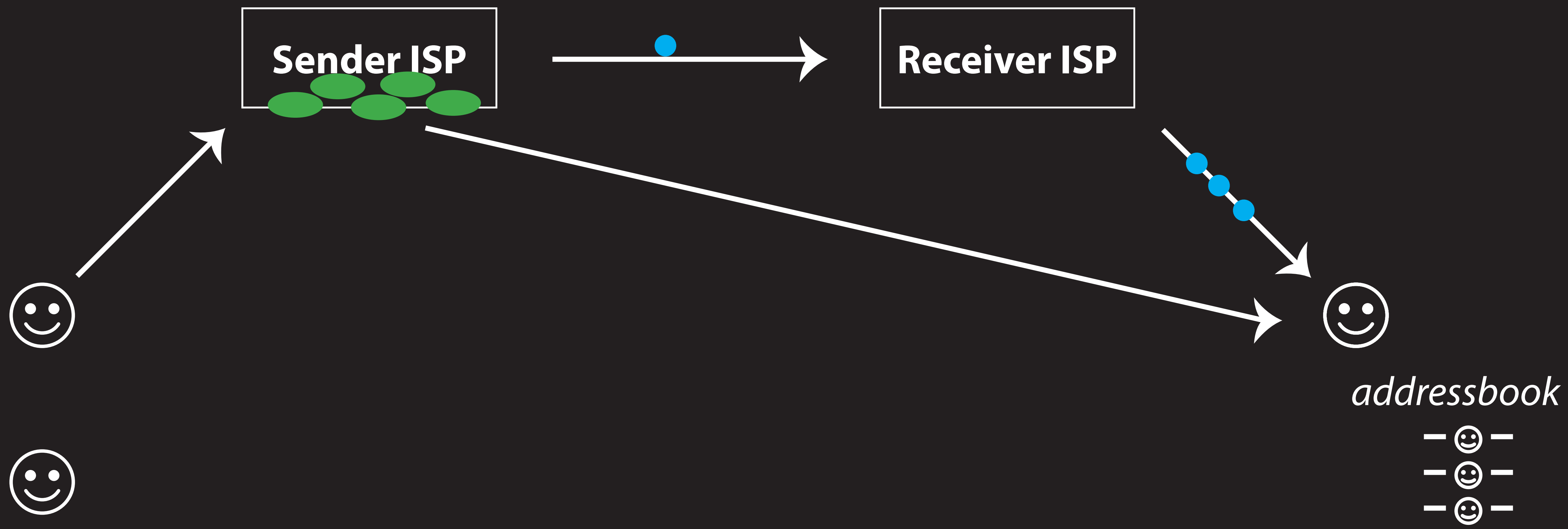
Simplest case + UDP notifications ●
The First Contact Problem.



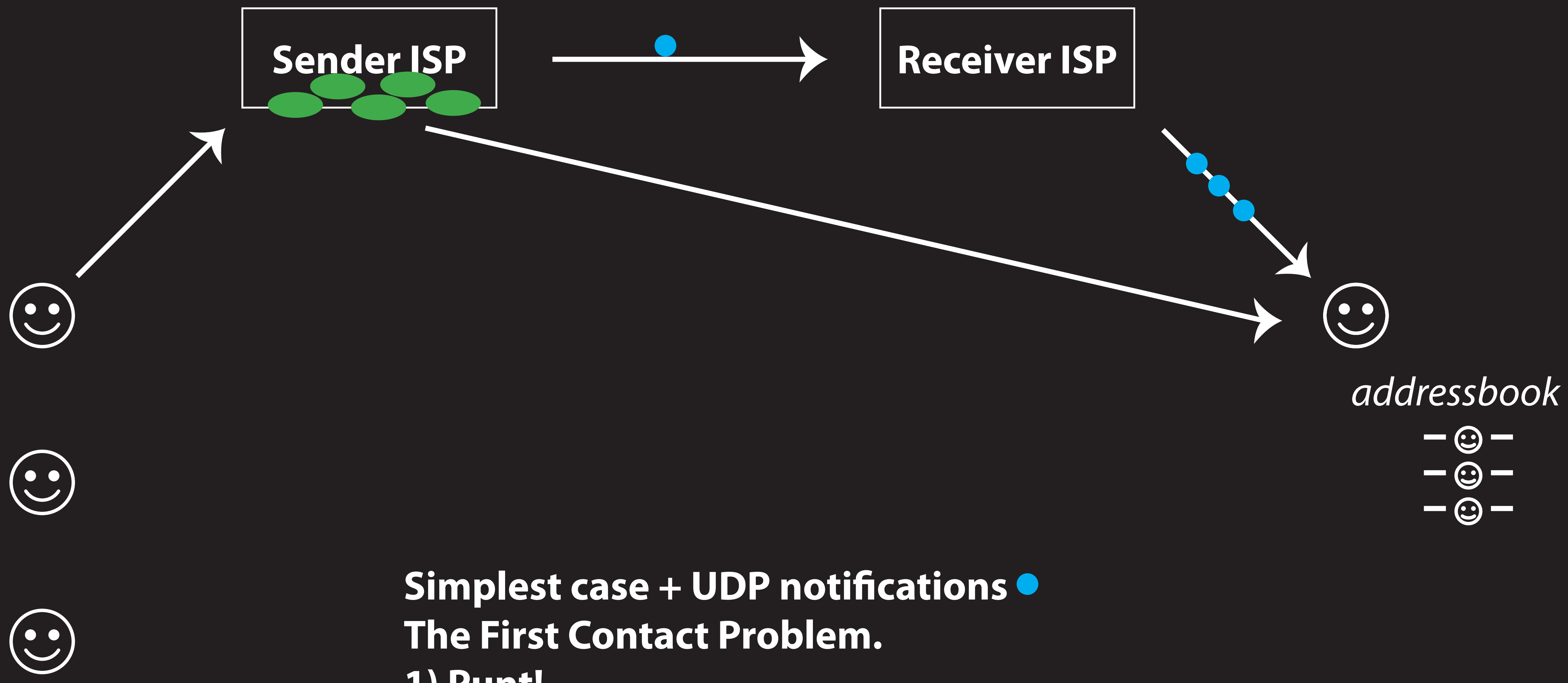
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!



Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
Out-of-band, Not Our Problem.



Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
Out-of-band, Not Our Problem.
Maybe: when I give you my business card it means "here's my email address, please fetch mail from me."

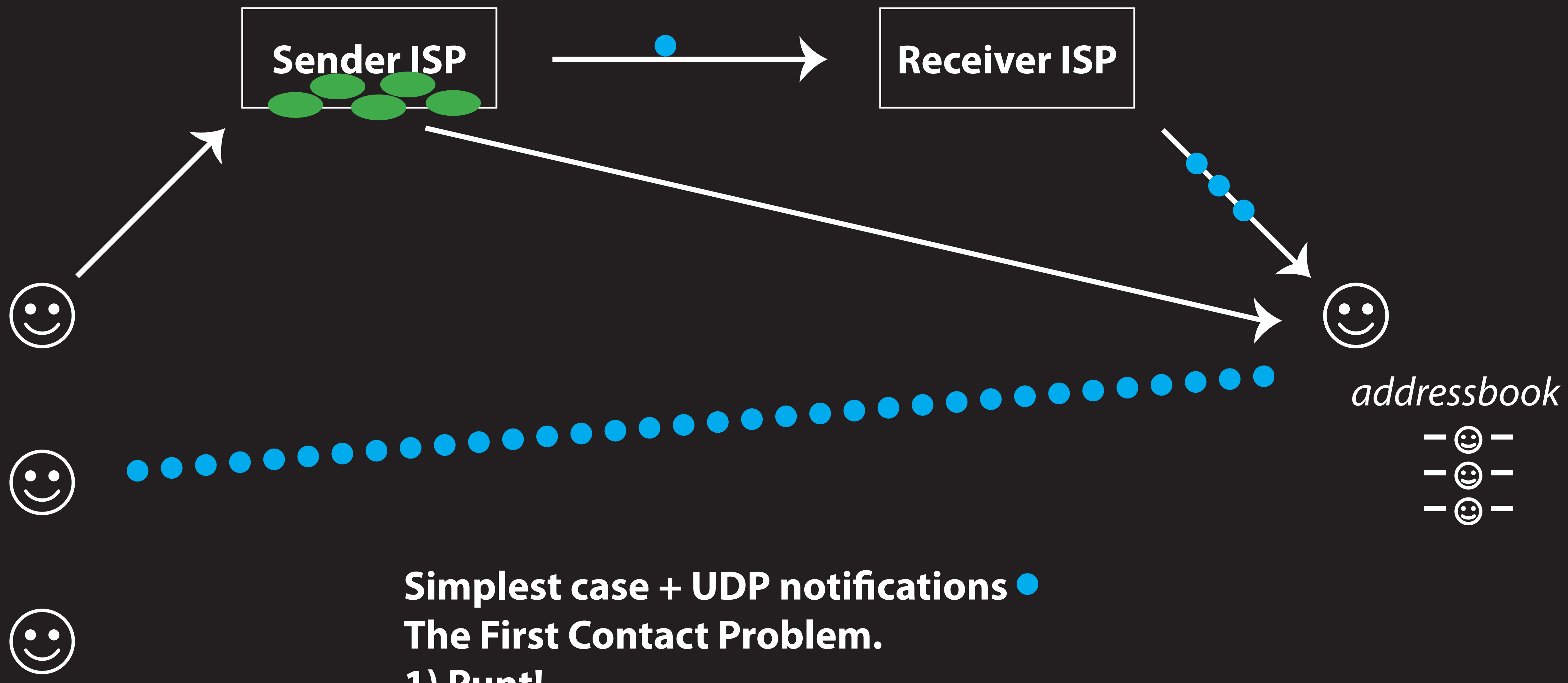


Simplest case + UDP notifications ●

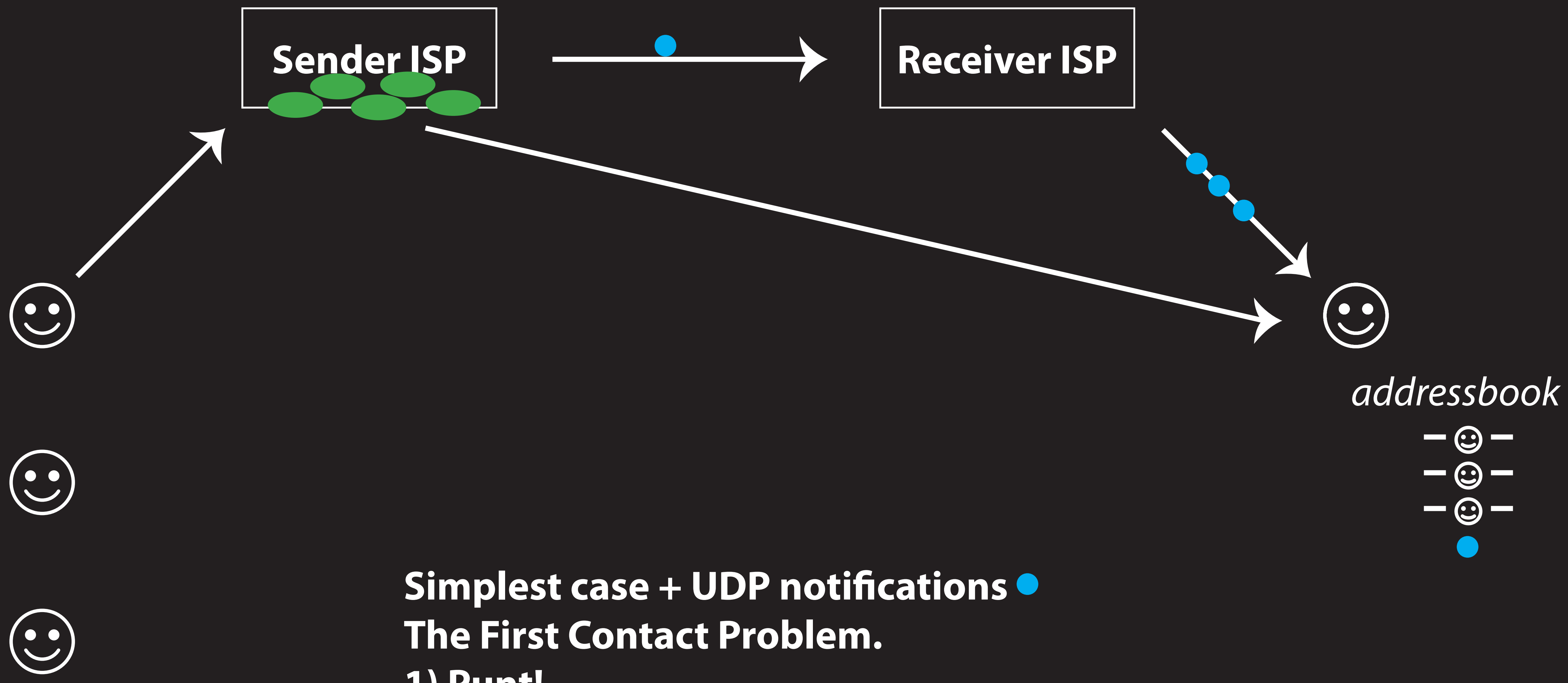
The First Contact Problem.

1) Punt!

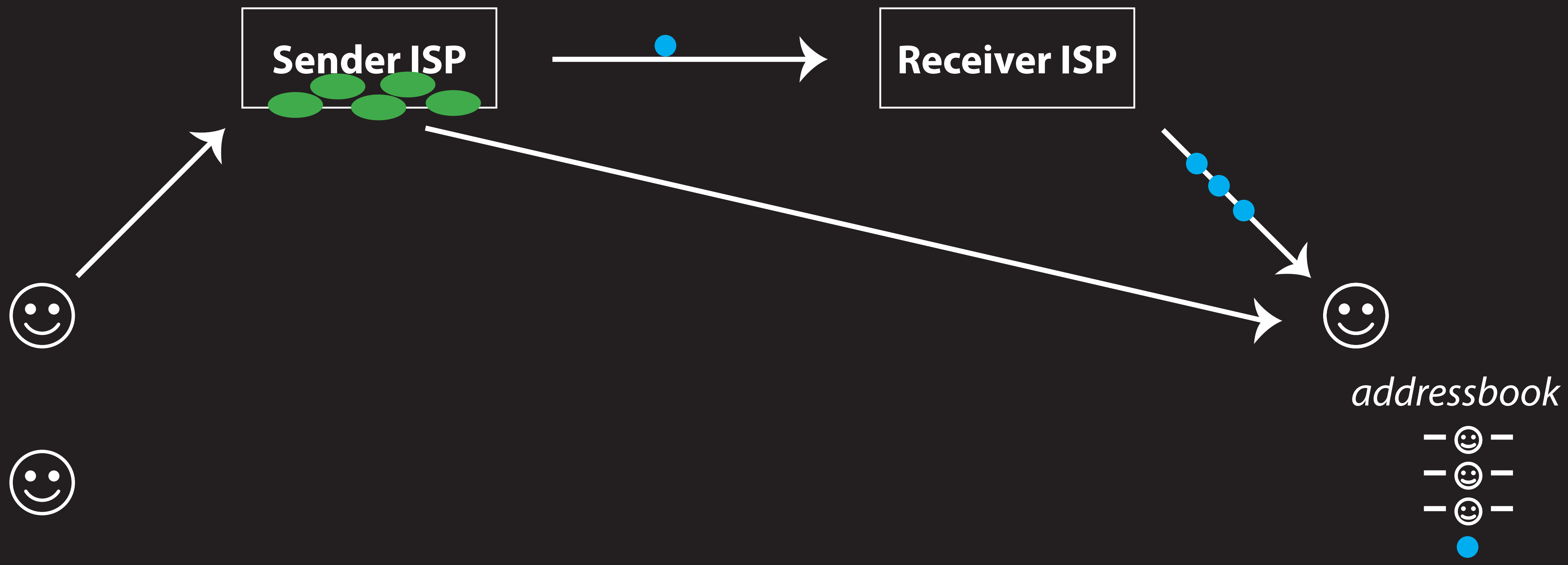
2) ●



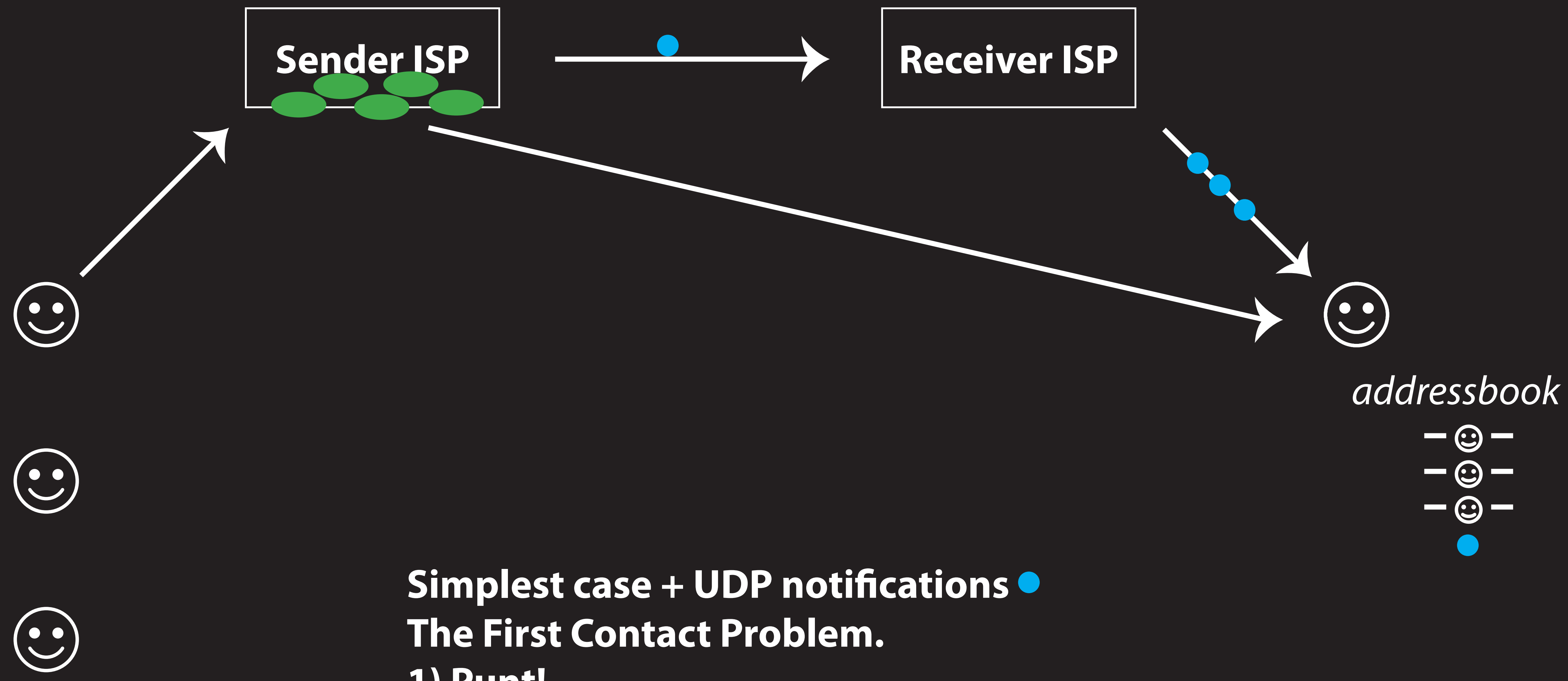
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
2) Senders keep notifying a receiver until the message is retrieved.



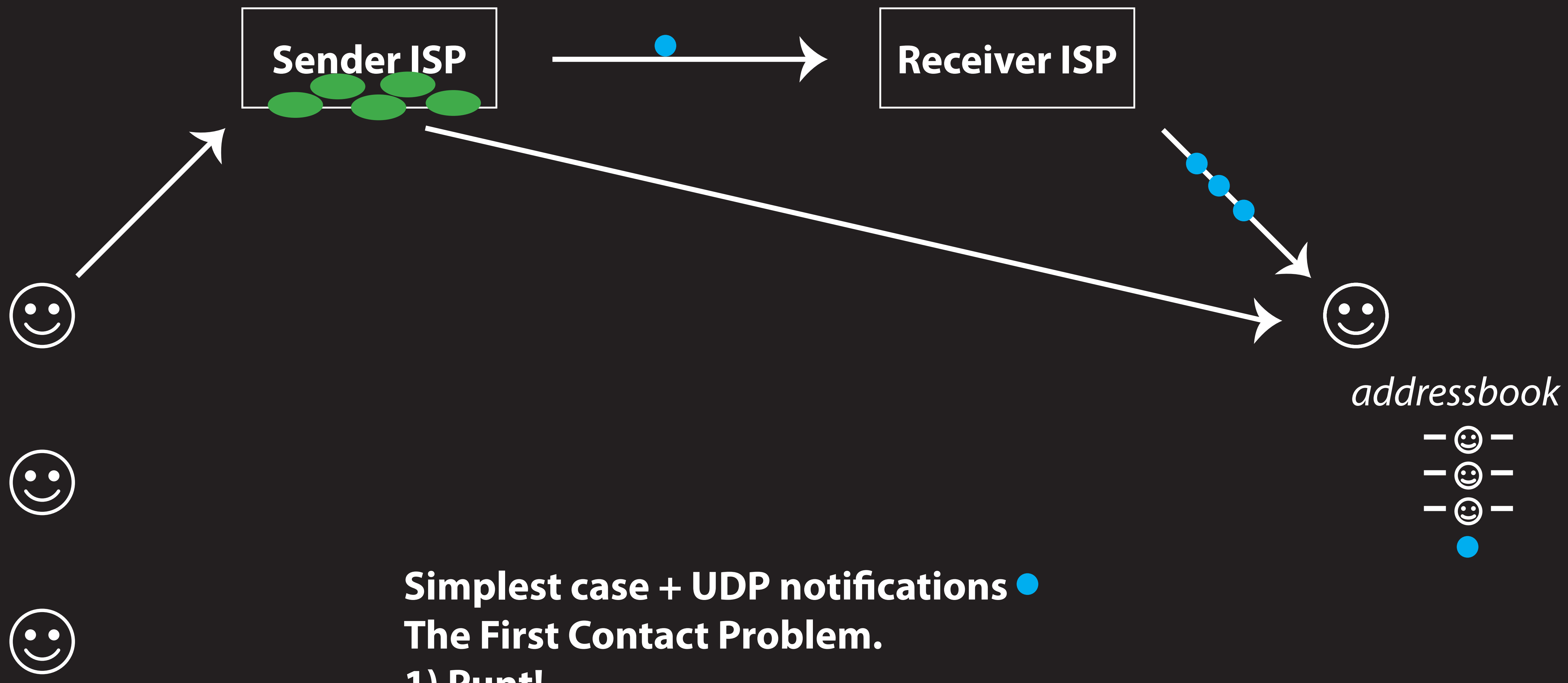
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
2) Senders keep notifying a receiver until the message is retrieved.
Receivers must respect notifications from out-of-addressbook senders.



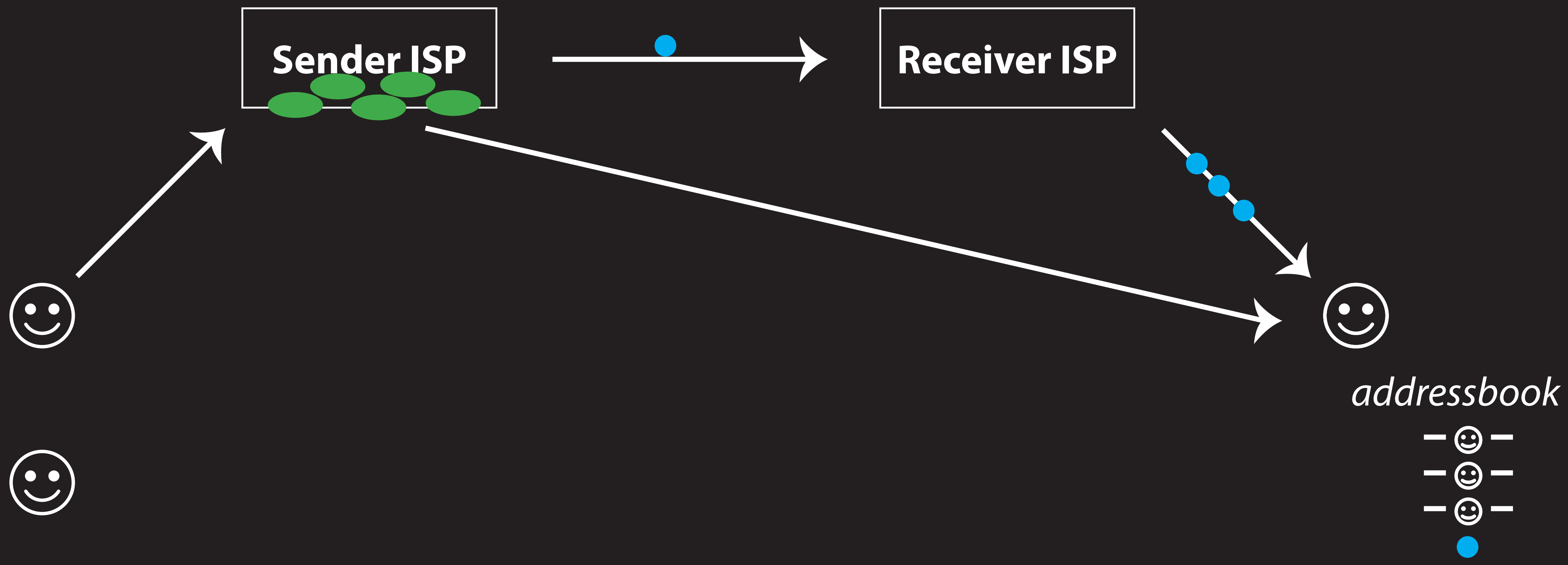
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
2) Senders keep notifying a receiver until the message is retrieved.
Receivers must respect notifications from out-of-addressbook senders.
This brings back the spam problem!



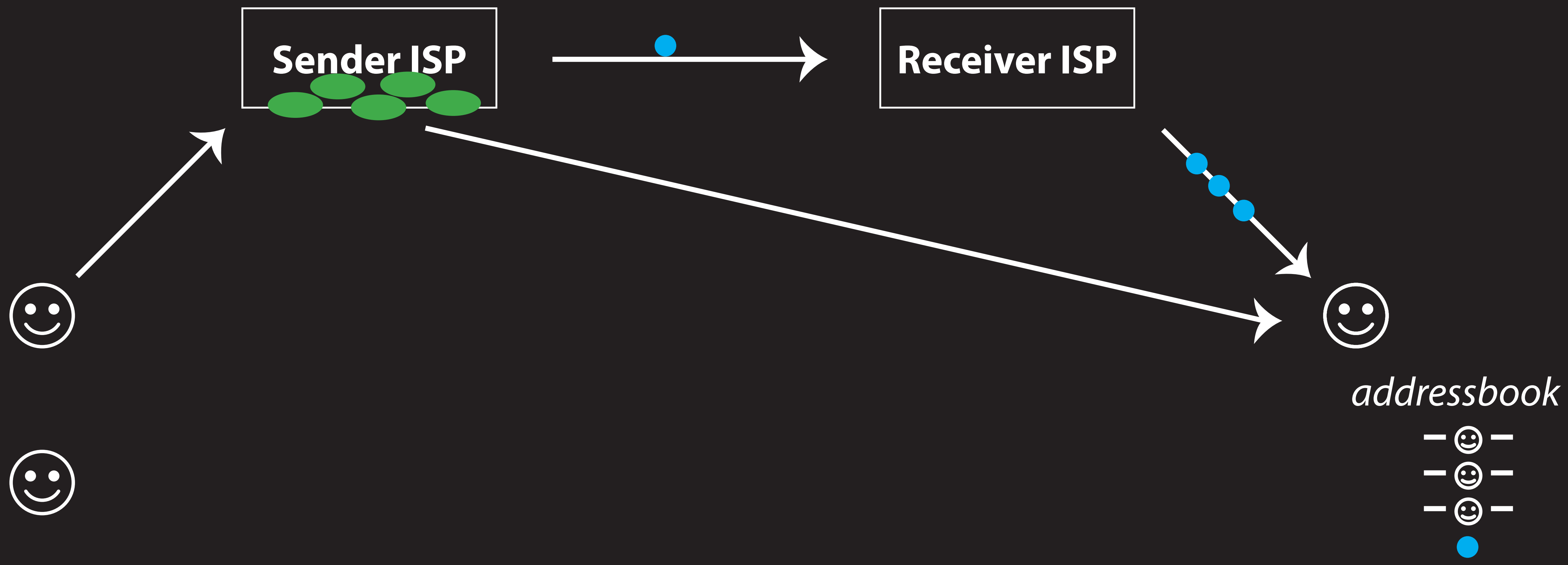
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
2) Senders keep notifying a receiver until the message is retrieved.
Receivers must respect notifications from out-of-addressbook senders.
This brings back the spam problem!
Easier to filter by reputation of sender.



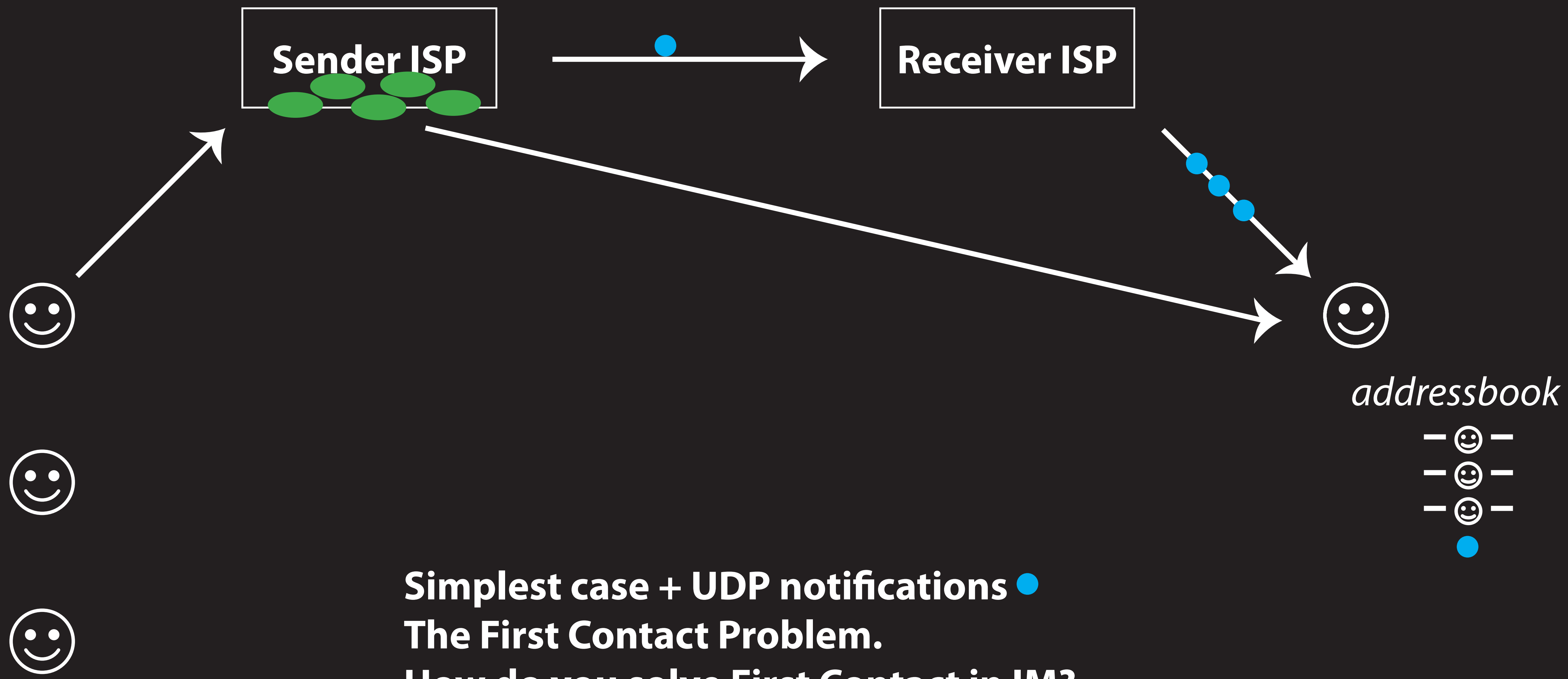
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
Out-of-band, Not Our Problem.



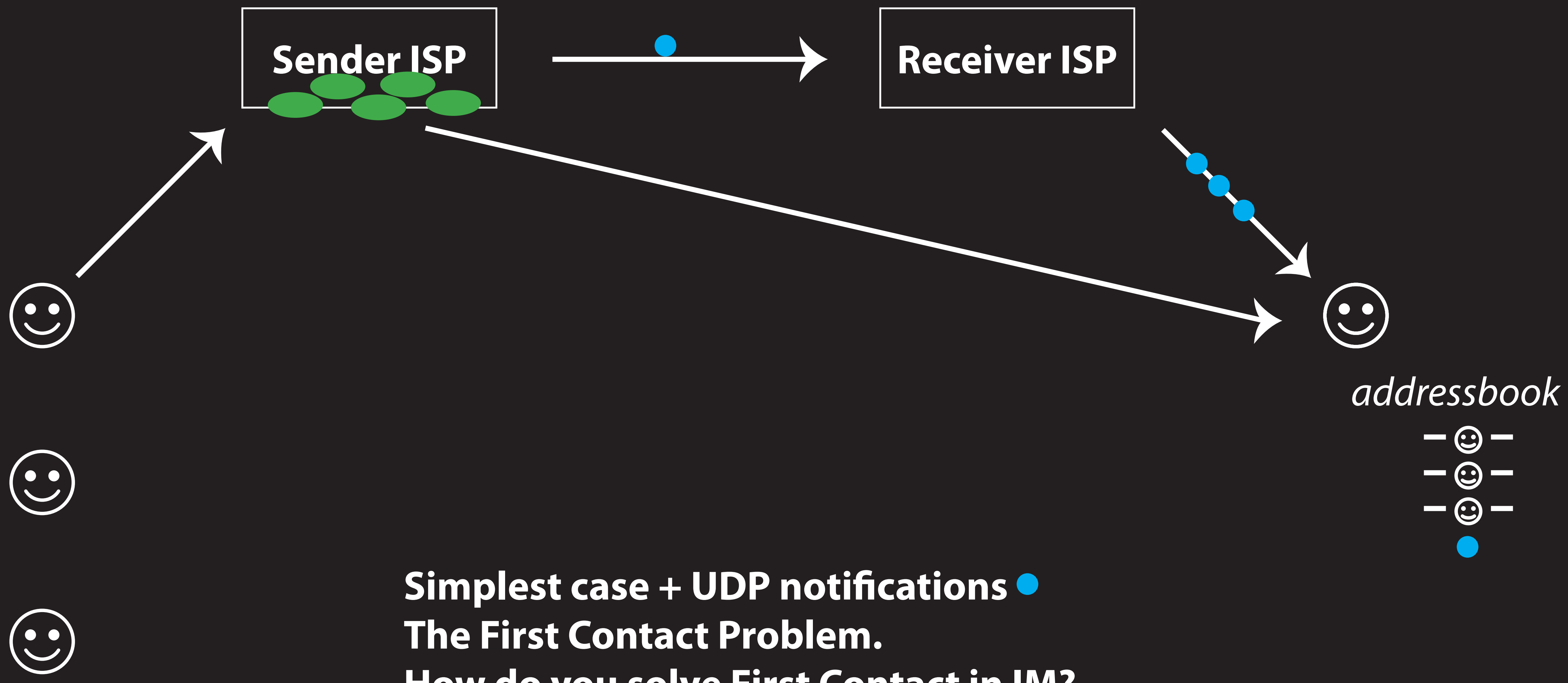
Simplest case + UDP notifications ●
The First Contact Problem.
1) Punt!
Out-of-band, Not Our Problem.
2) Traditional email could be an out-of-band mechanism.



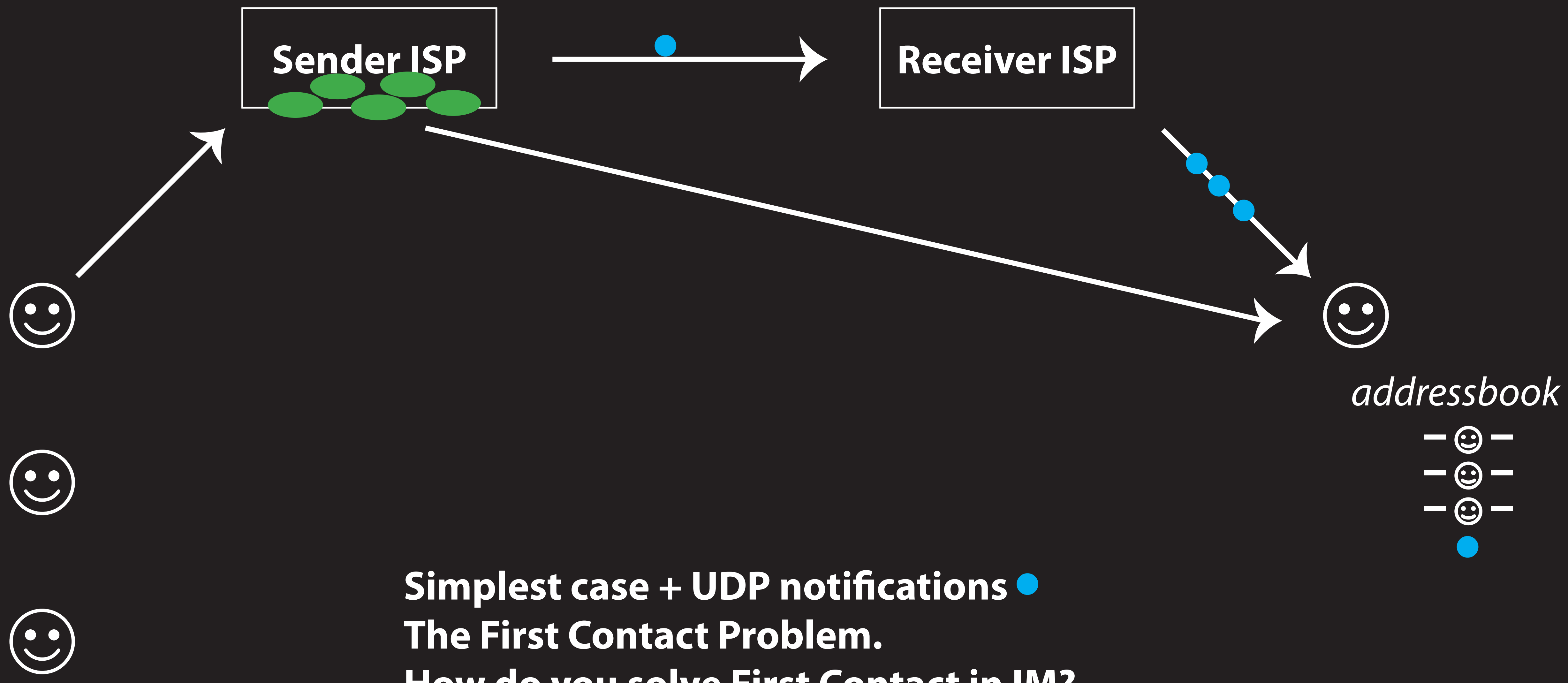
Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?



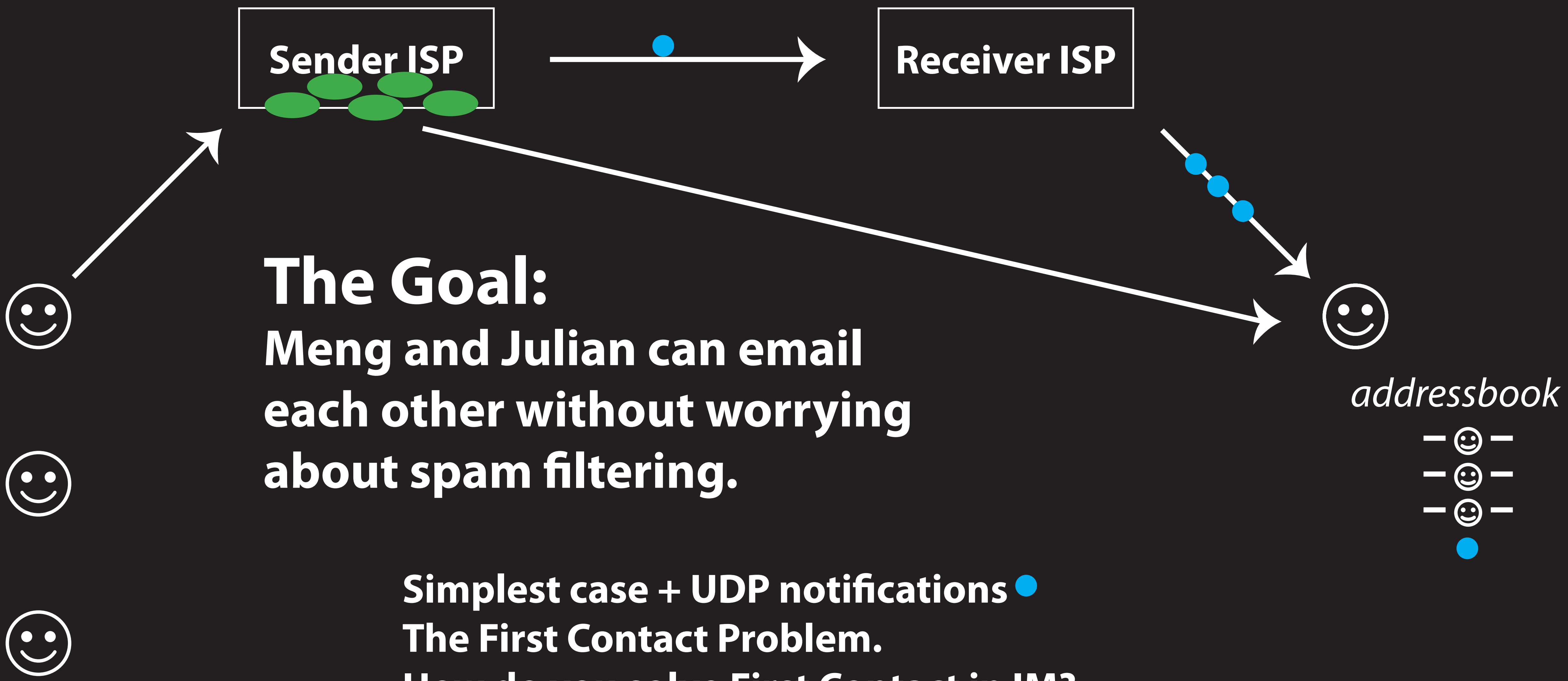
Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?
Different expectations for different media.



Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?
Different expectations for different media.
Email is the medium of least reserve.



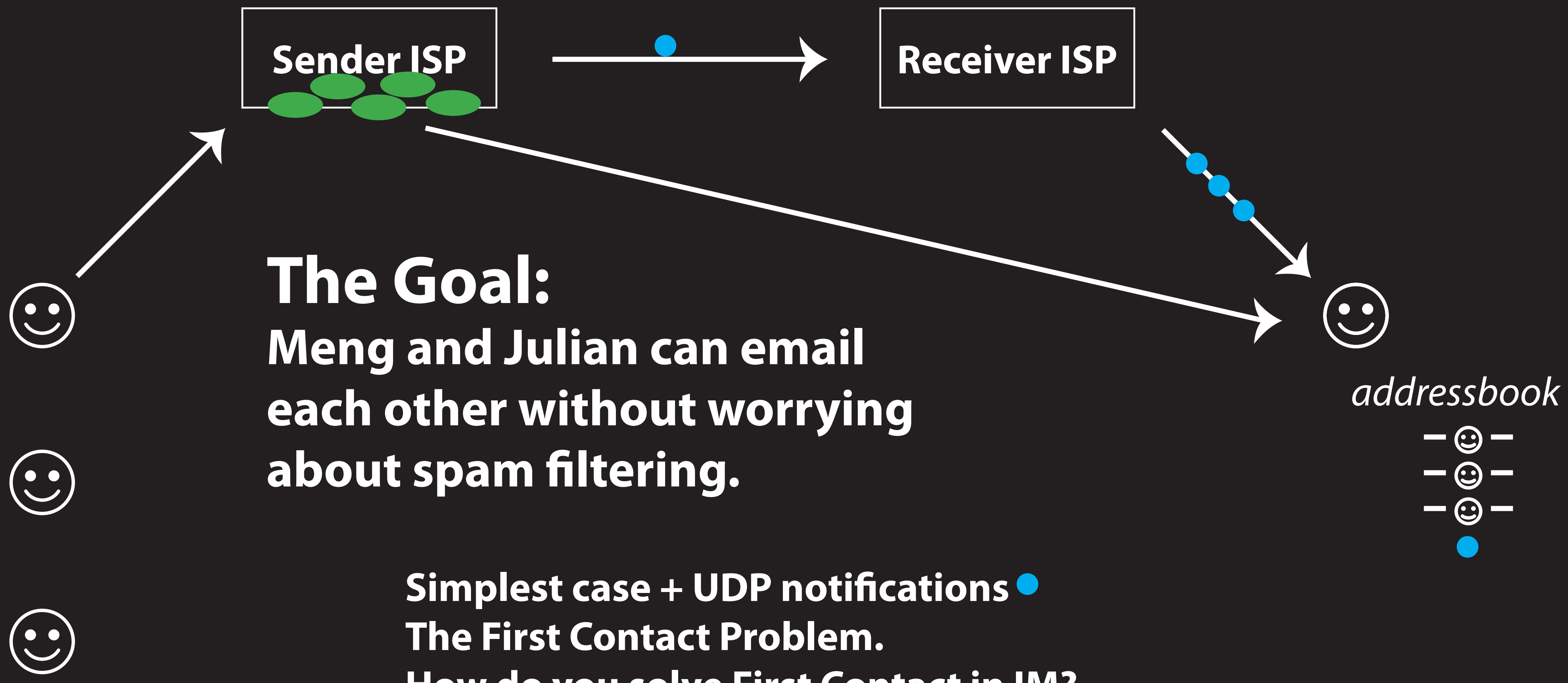
Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?
Different expectations for different media.
Email is the medium of least reserve.
Also now the medium of least confidence.



The Goal:

Meng and Julian can email each other without worrying about spam filtering.

Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?
Different expectations for different media.
Email is the medium of least reserve.
Also now the medium of least confidence.



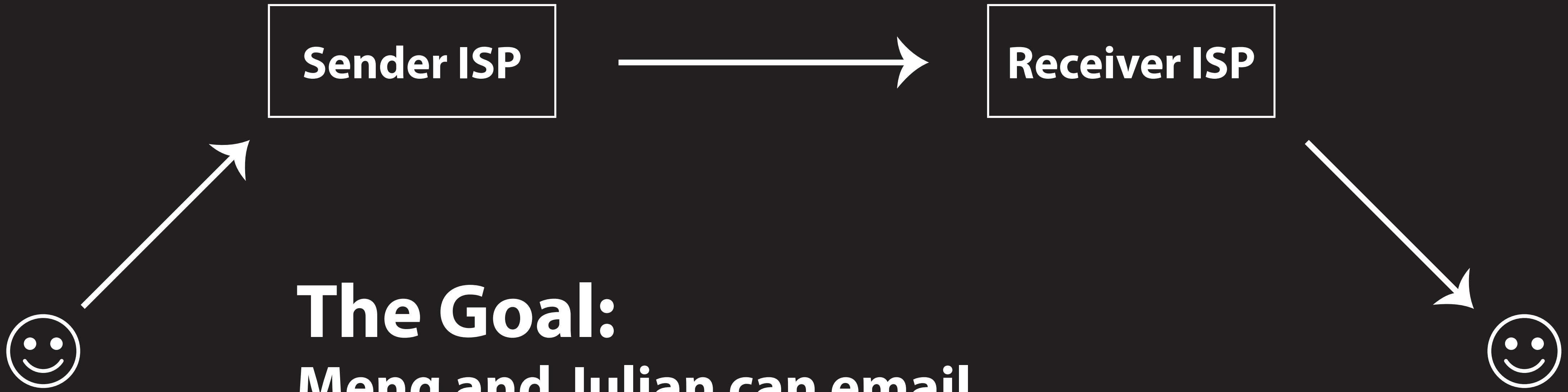
The Goal:

Meng and Julian can email each other without worrying about spam filtering.

Simplest case + UDP notifications ●
The First Contact Problem.
How do you solve First Contact in IM?
Different expectations for different media.
Email is the medium of least reserve.
Also now the medium of least confidence.
Shrug.

The Goal:

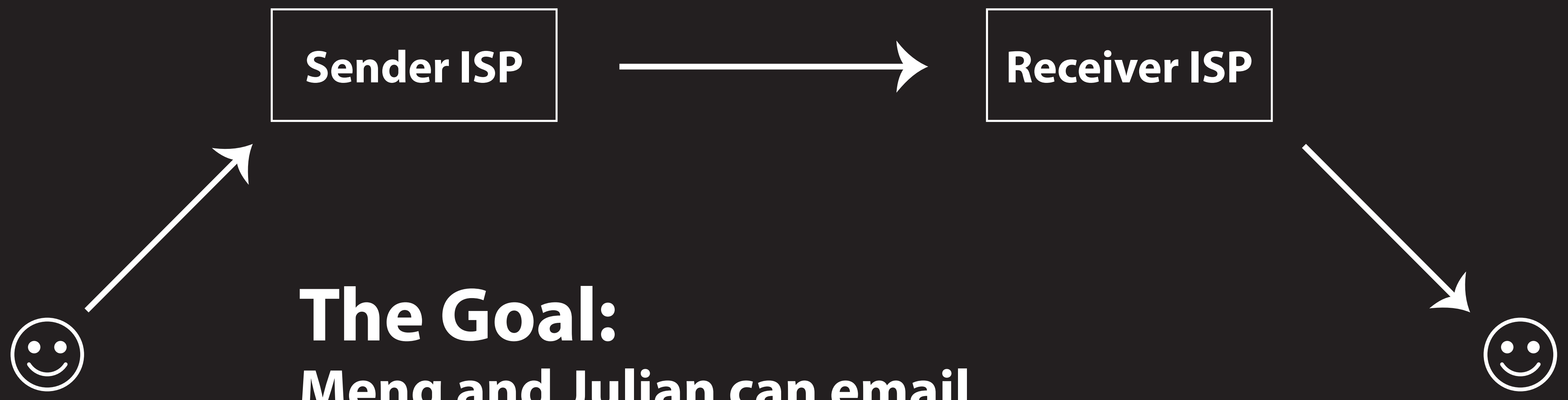
Meng and Julian can email each other without worrying about spam filtering.



The Goal:

Meng and Julian can email each other without worrying about spam filtering.

SHOW ME THE CODE!

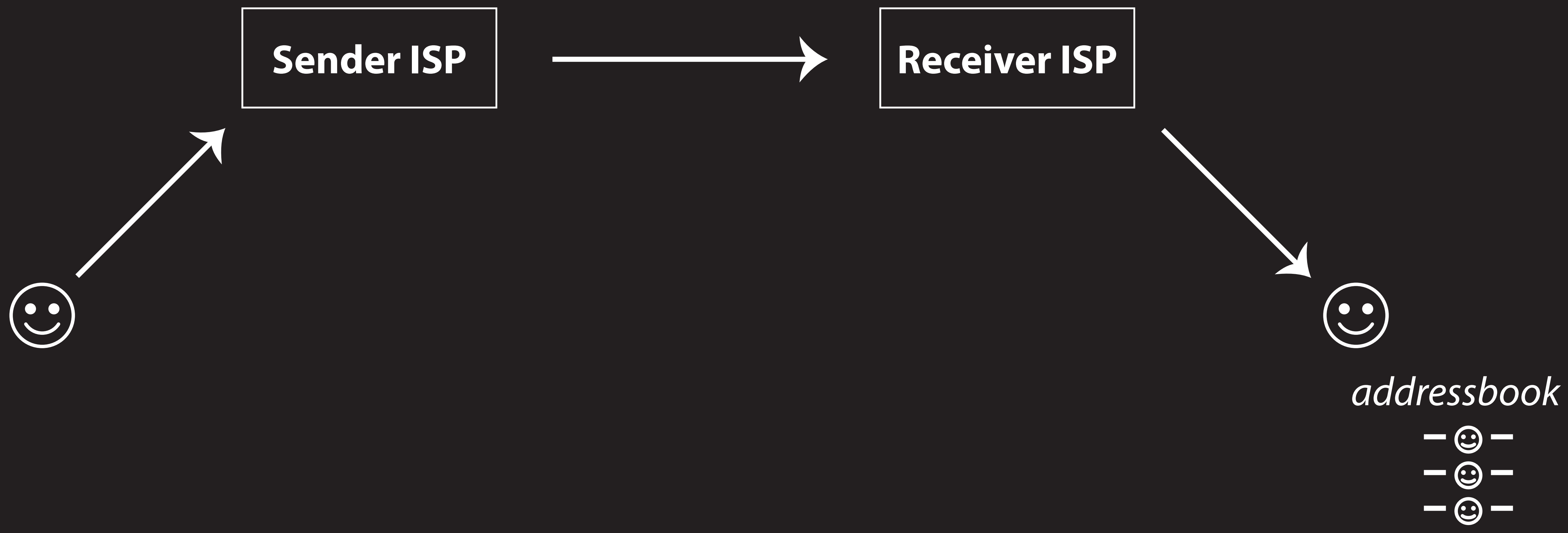


The Goal:

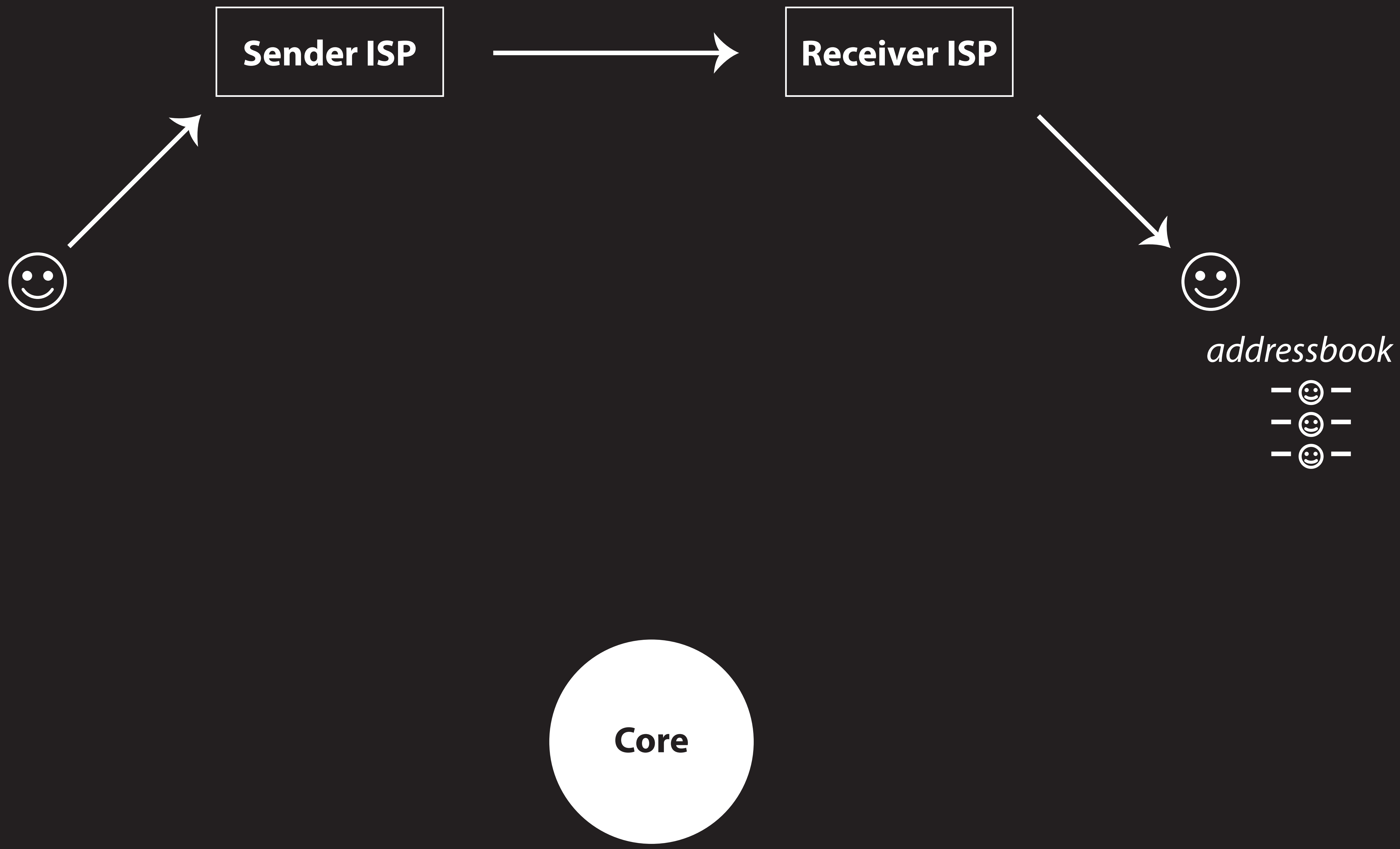
Meng and Julian can email each other without worrying about spam filtering.

SHOW ME THE CODE!
Let's do a demo.
Julian sends mail.
Meng retrieves it.

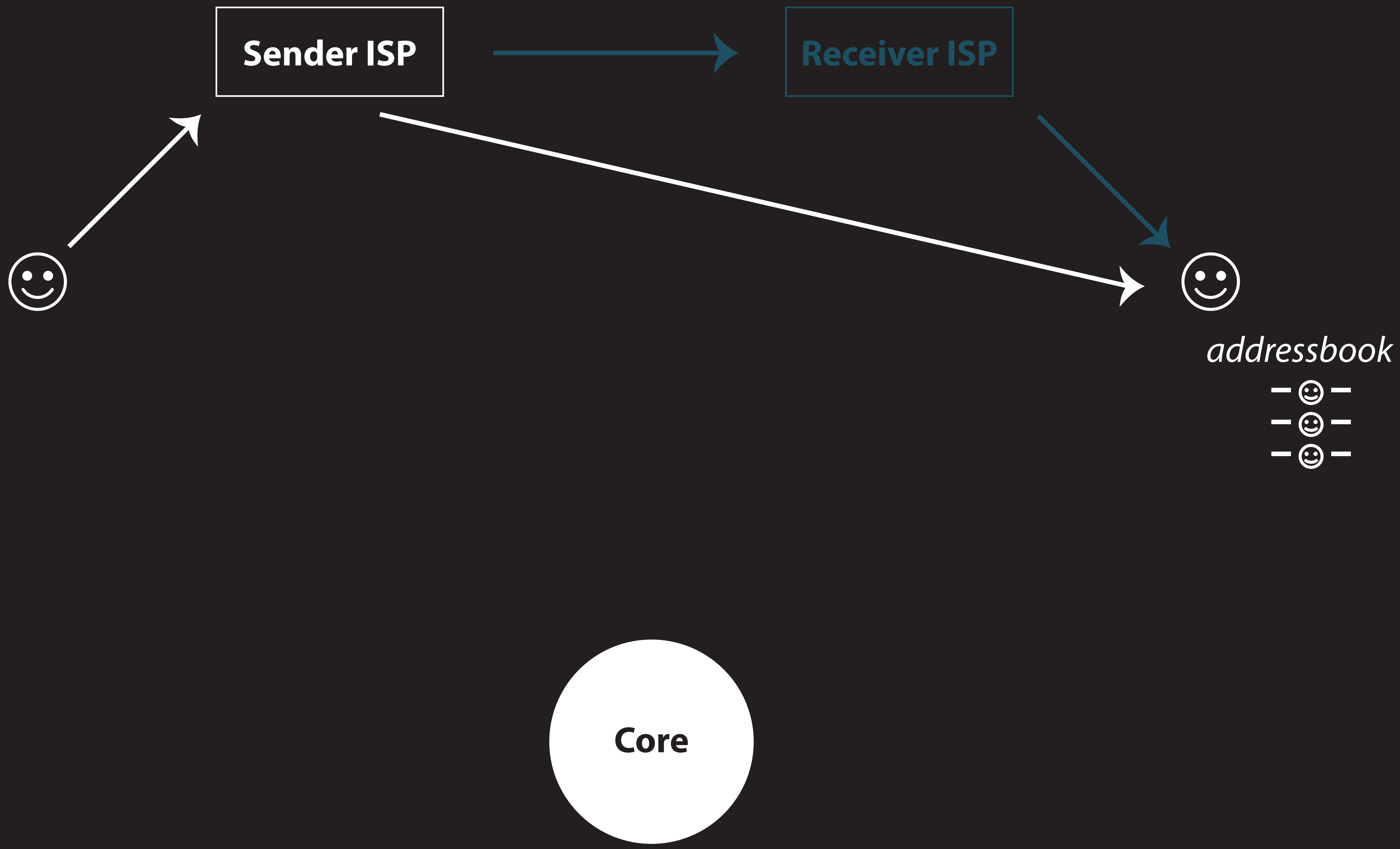
Simplest Possible Implementation



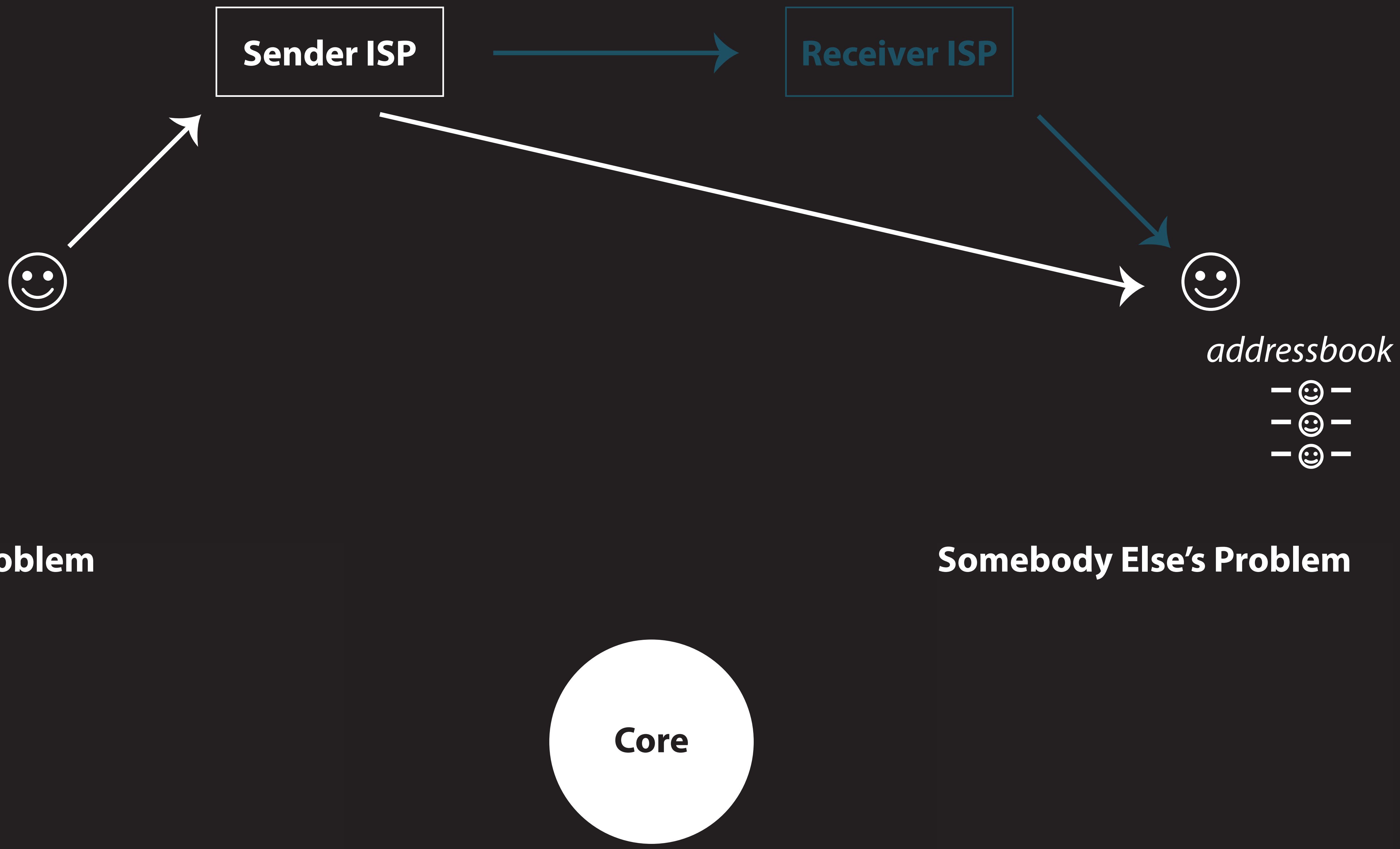
Simplest Possible Implementation



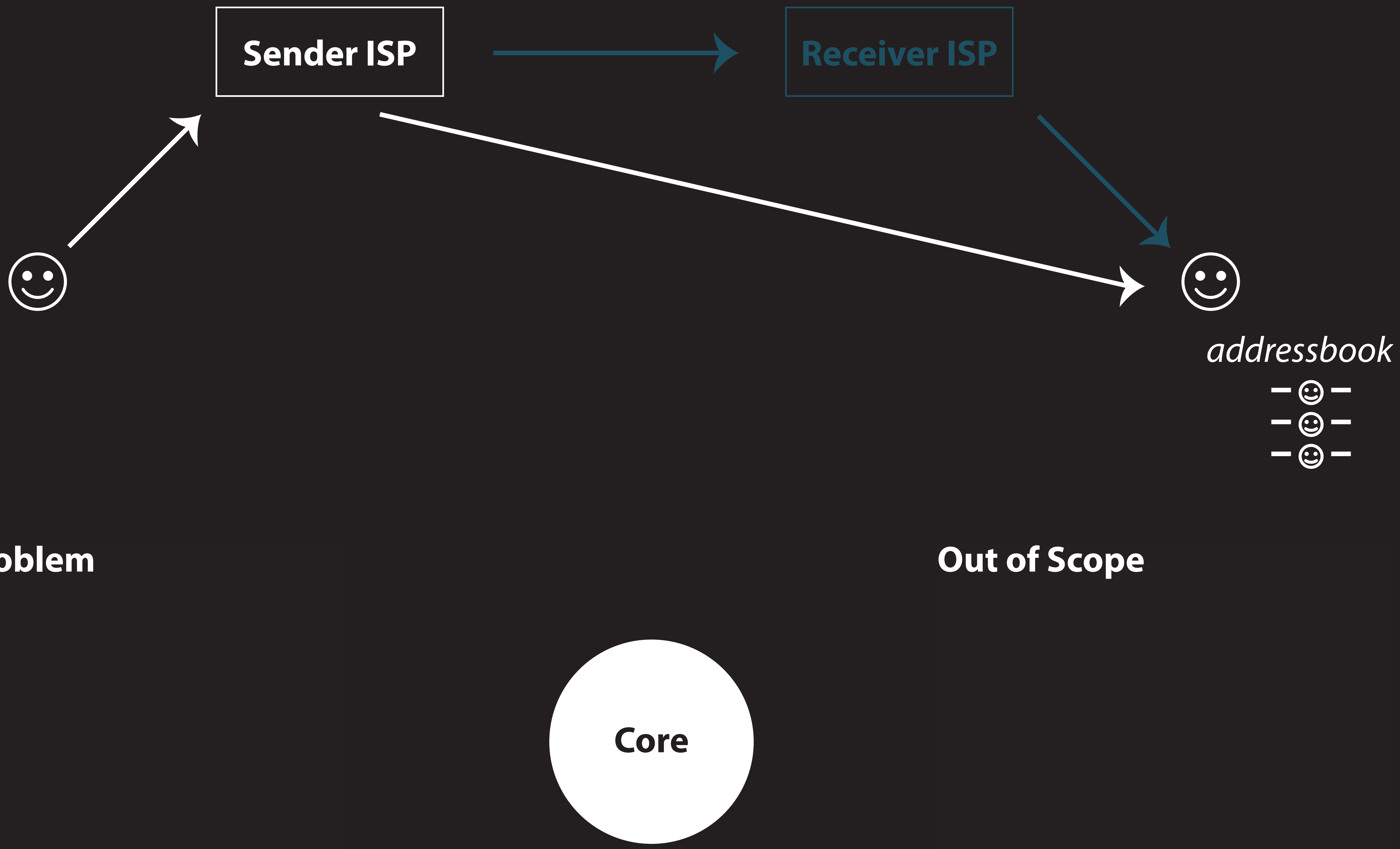
Simplest Possible Implementation



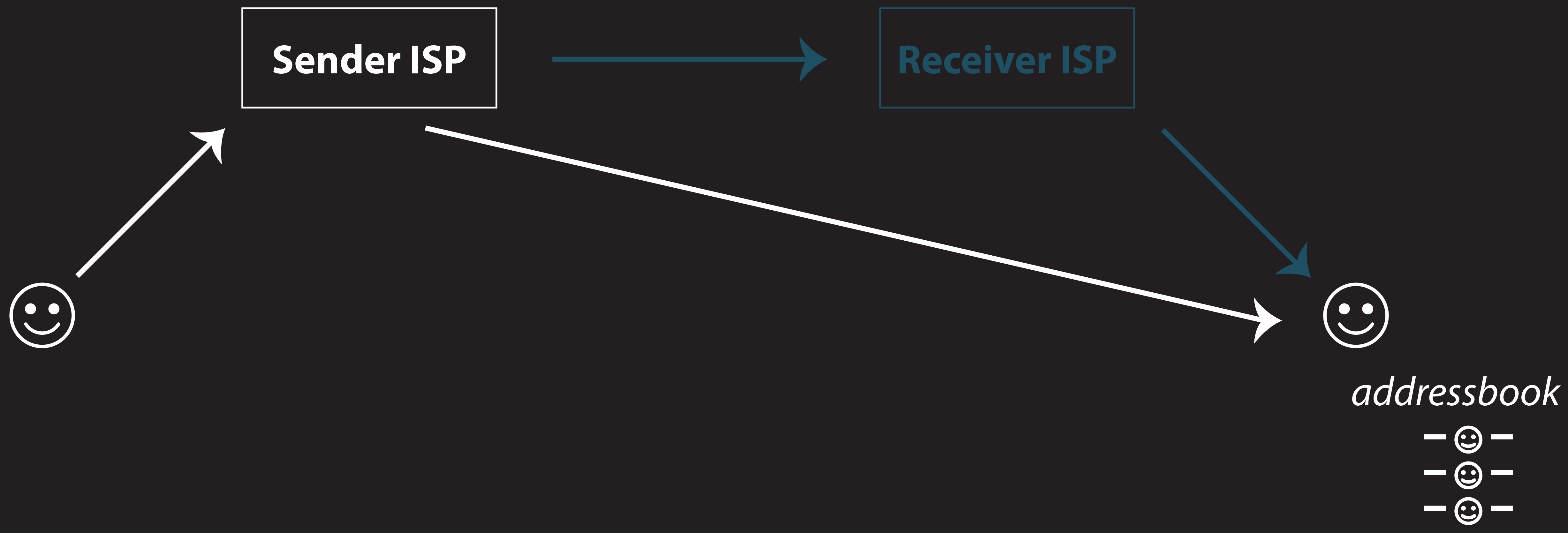
Simplest Possible Implementation



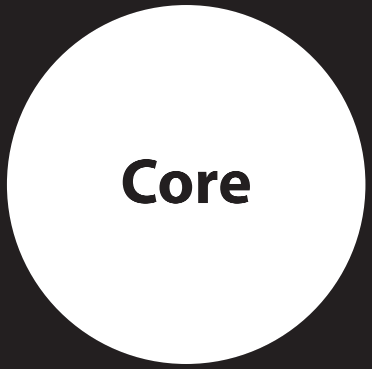
Simplest Possible Implementation



Simplest Possible Implementation

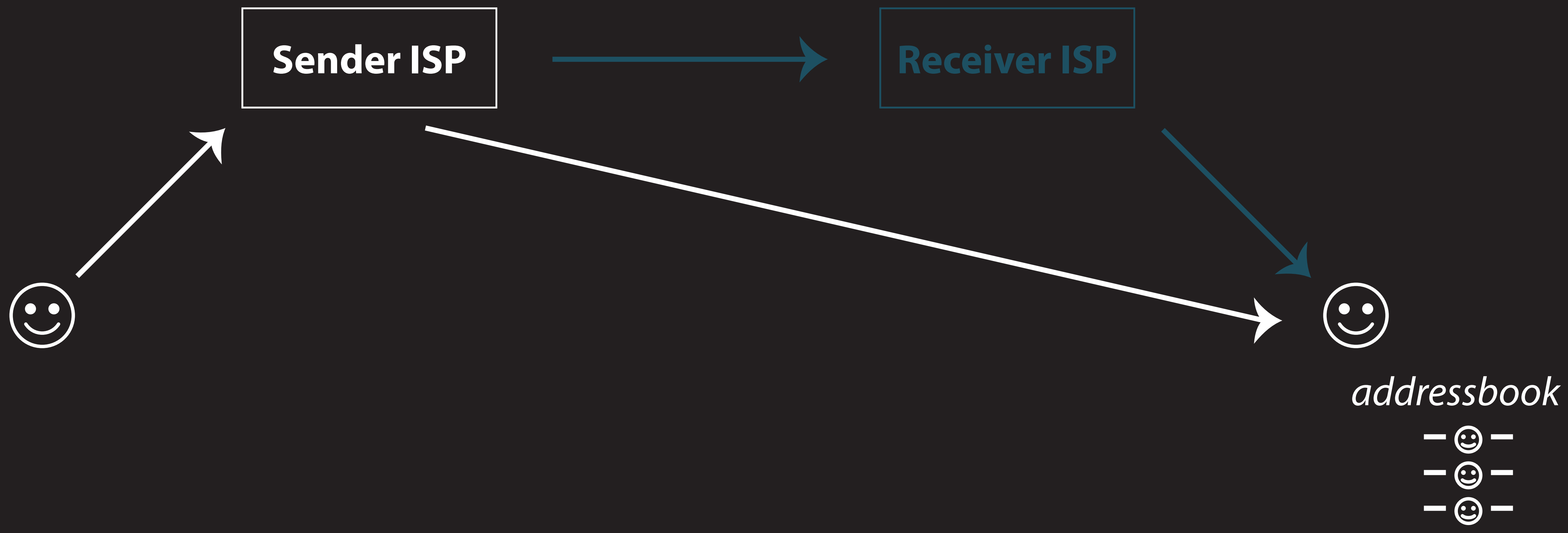


Our Problem
- **given an outbox URL, retrieve messages**



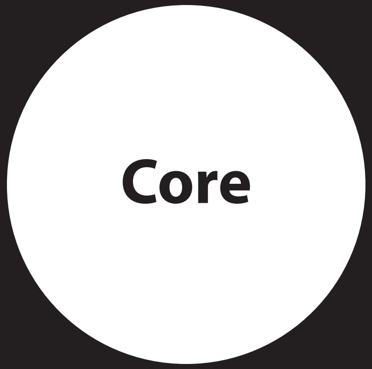
Out of Scope
- **where is the outbox URL kept?**
- **where did the outbox URL come from?**

Simplest Possible Implementation



Our Problem

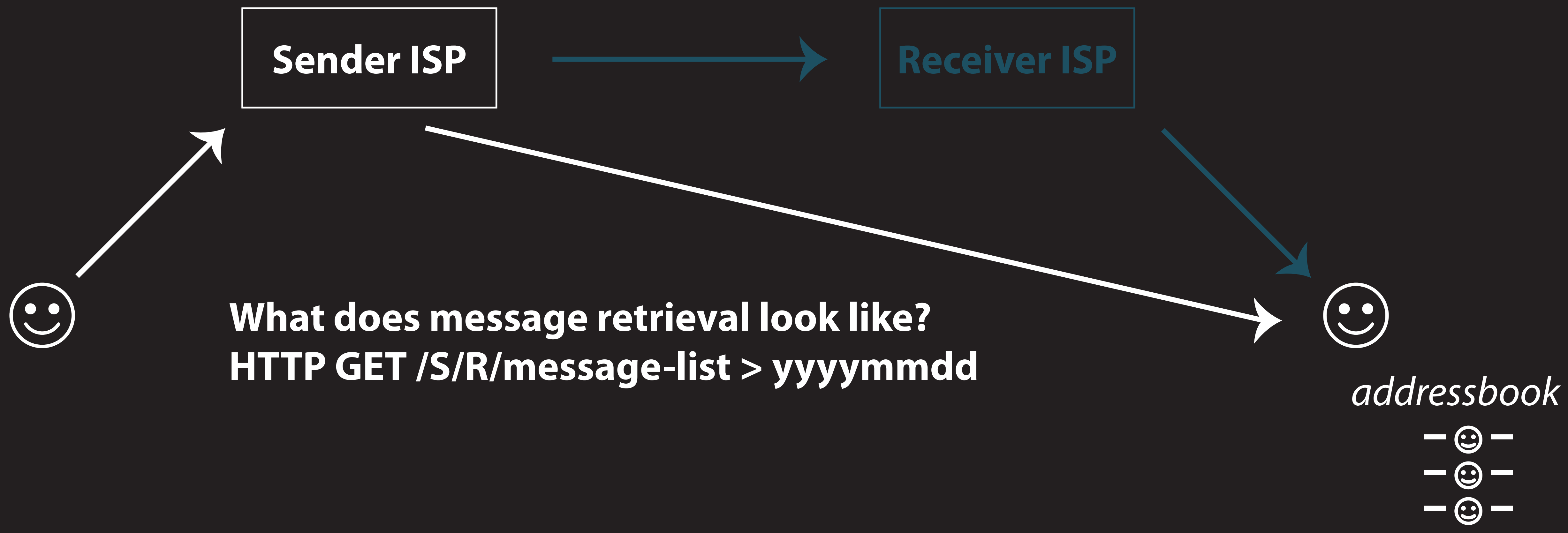
- given an outbox URL, retrieve messages



Out of Scope

- where is the outbox URL kept?
- where did the outbox URL come from?
- how did the message get into the outbox?

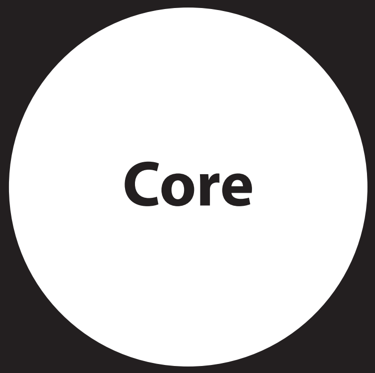
Simplest Possible Implementation



What does message retrieval look like?
HTTP GET /S/R/message-list > yyyymmdd

Our Problem

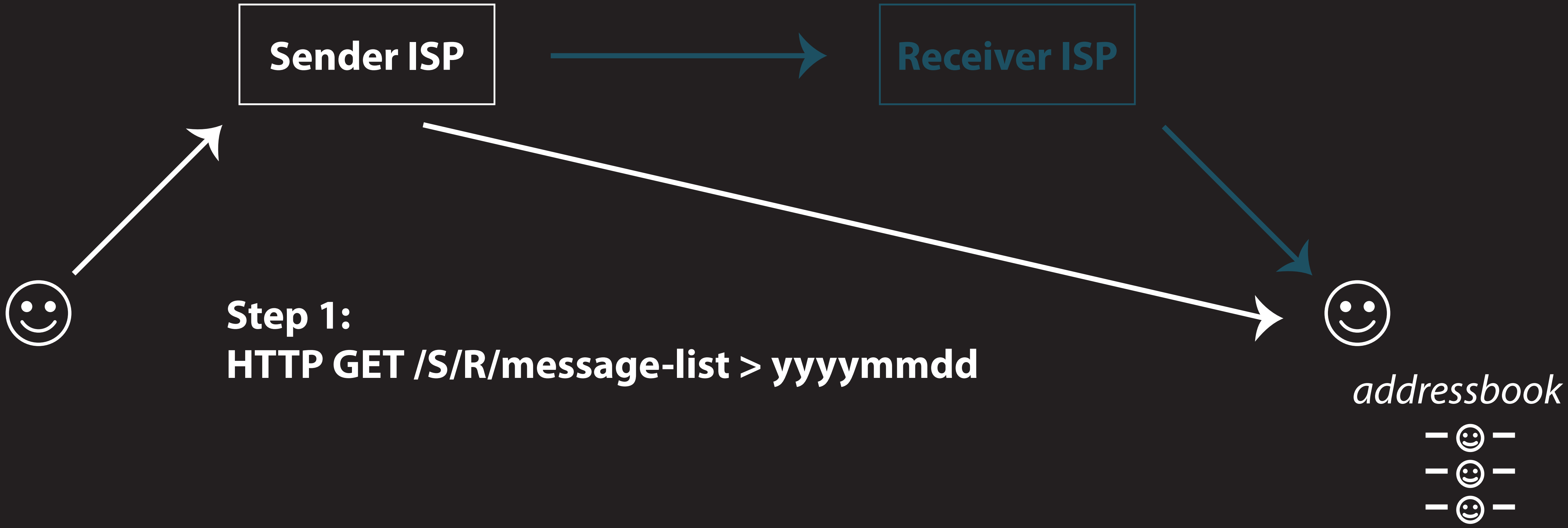
- **given an outbox URL, retrieve messages**



Out of Scope

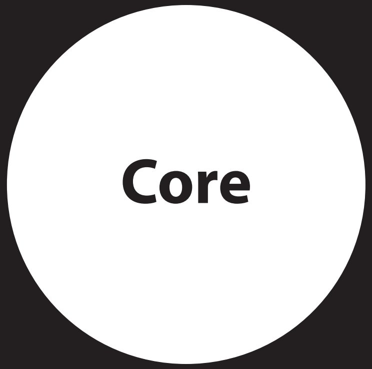
- **where is the outbox URL kept?**
- **where did the outbox URL come from?**
- **how did the message get into the outbox?**

Simplest Possible Implementation



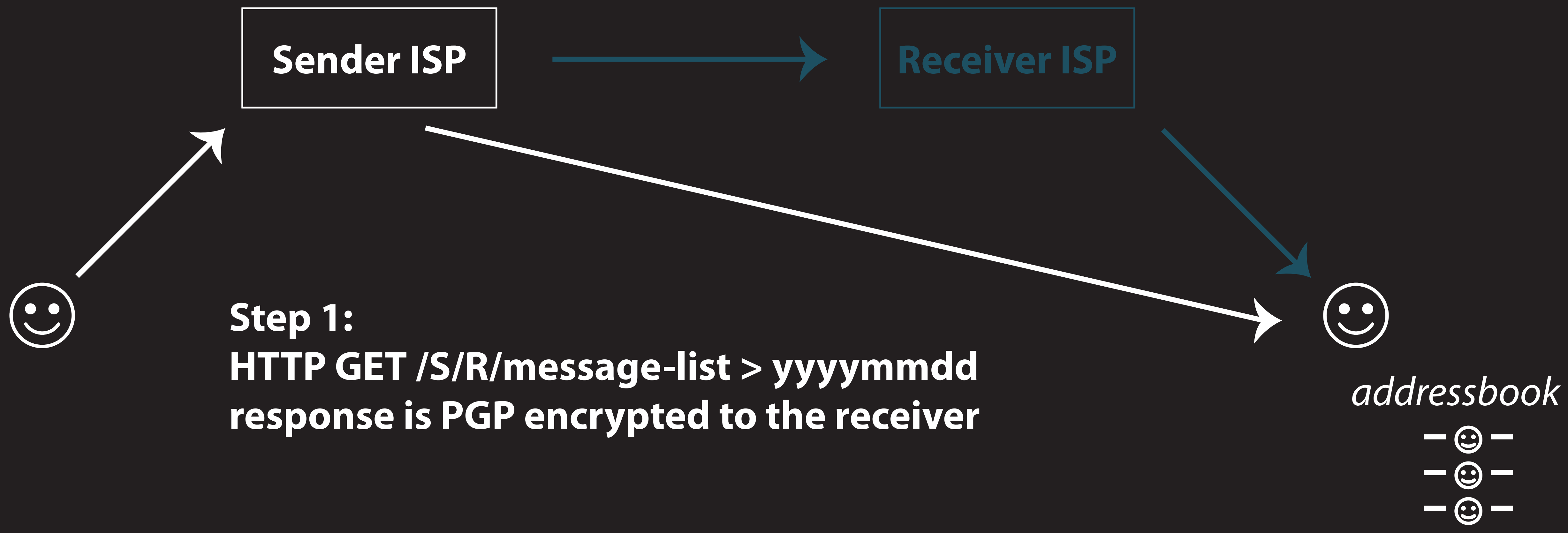
Step 1:
HTTP GET /S/R/message-list > yyyymmdd

Our Problem
- **given an outbox URL, retrieve messages**



Out of Scope
- **where is the outbox URL kept?**
- **where did the outbox URL come from?**
- **how did the message get into the outbox?**

Simplest Possible Implementation



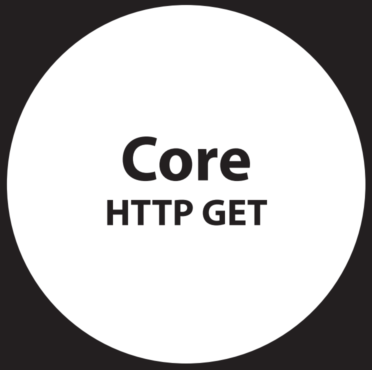
Step 1:
HTTP GET /S/R/message-list > yyyymmdd
response is PGP encrypted to the receiver

Our Problem

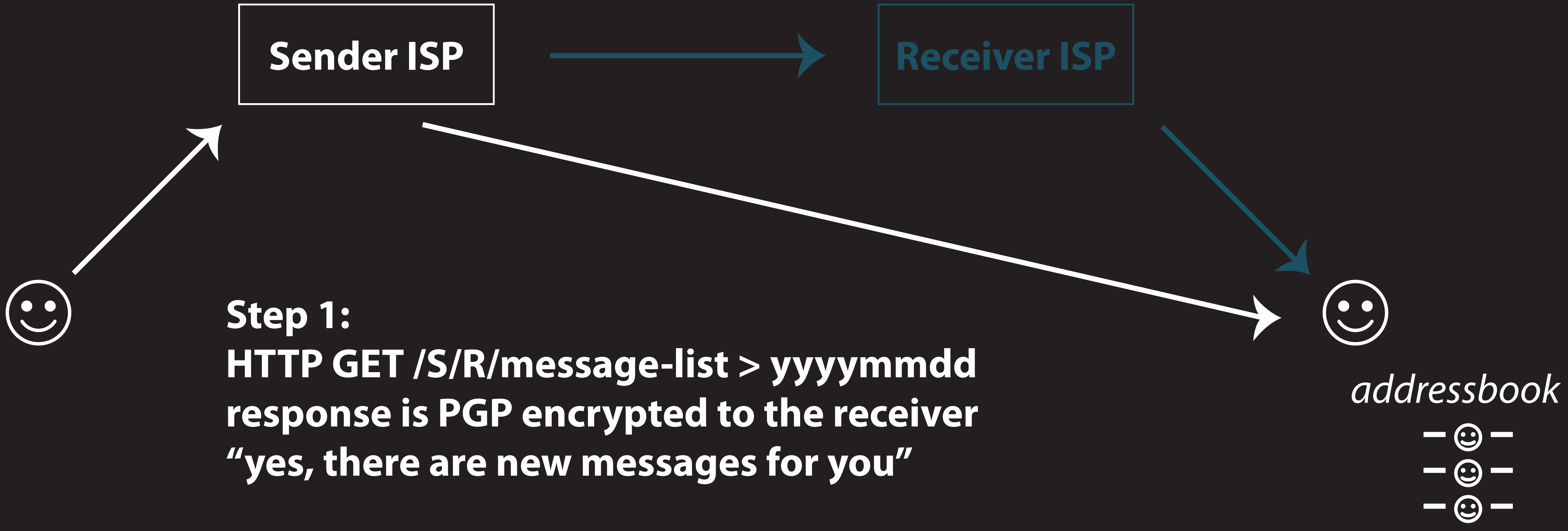
- given an outbox URL, retrieve messages

Out of Scope

- where is the outbox URL kept?
- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?



Simplest Possible Implementation

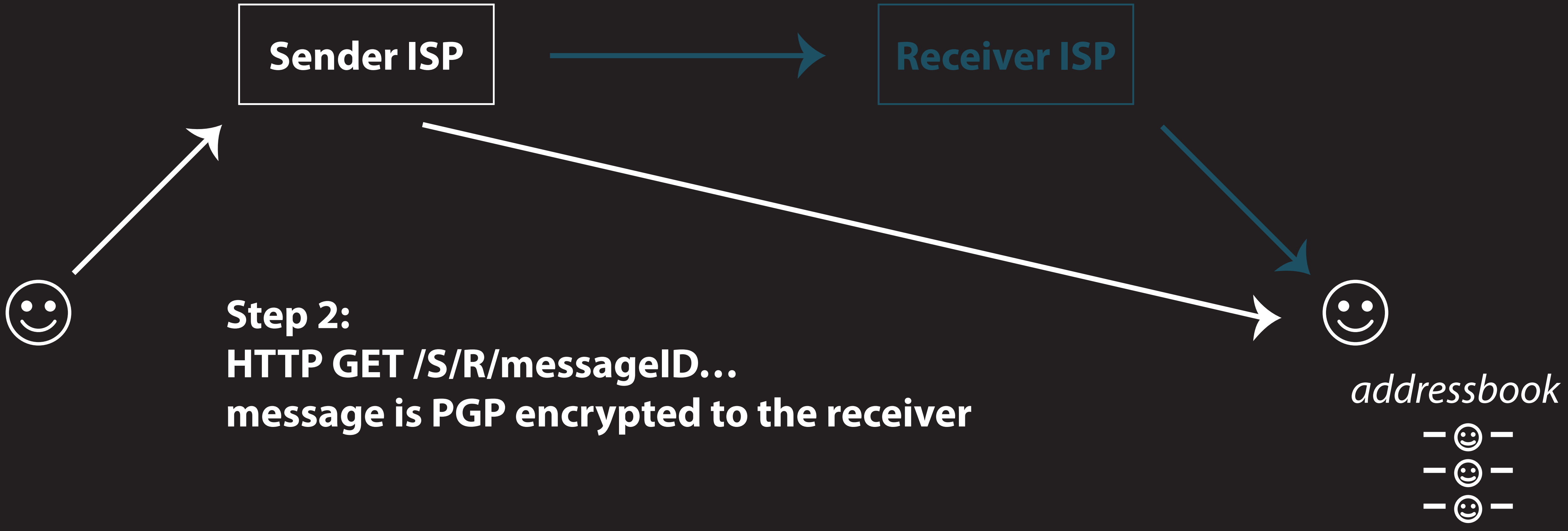


Step 1:
HTTP GET /S/R/message-list > yyyymmdd
response is PGP encrypted to the receiver
“yes, there are new messages for you”

Our Problem
- **given an outbox URL, retrieve messages**

Out of Scope
- **where is the outbox URL kept?**
- **where did the outbox URL come from?**
- **how did the message get into the outbox?**
- **how does the sender server know the receiver’s PGP key?**

Simplest Possible Implementation



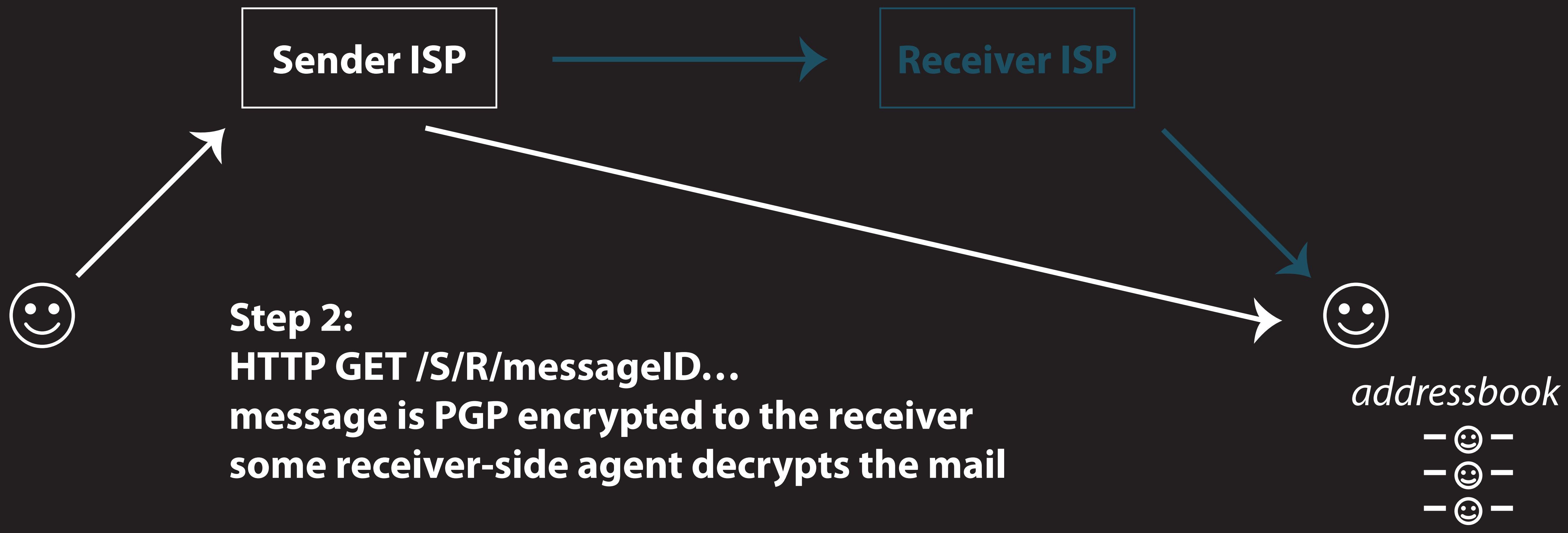
Step 2:
HTTP GET /S/R/messageID…
message is PGP encrypted to the receiver

Our Problem
- given an outbox URL, retrieve messages

Core
HTTP GET
Full Crypto

Out of Scope
- where is the outbox URL kept?
- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?

Simplest Possible Implementation



Step 2:
HTTP GET /S/R/messageID…
message is PGP encrypted to the receiver
some receiver-side agent decrypts the mail

Our Problem

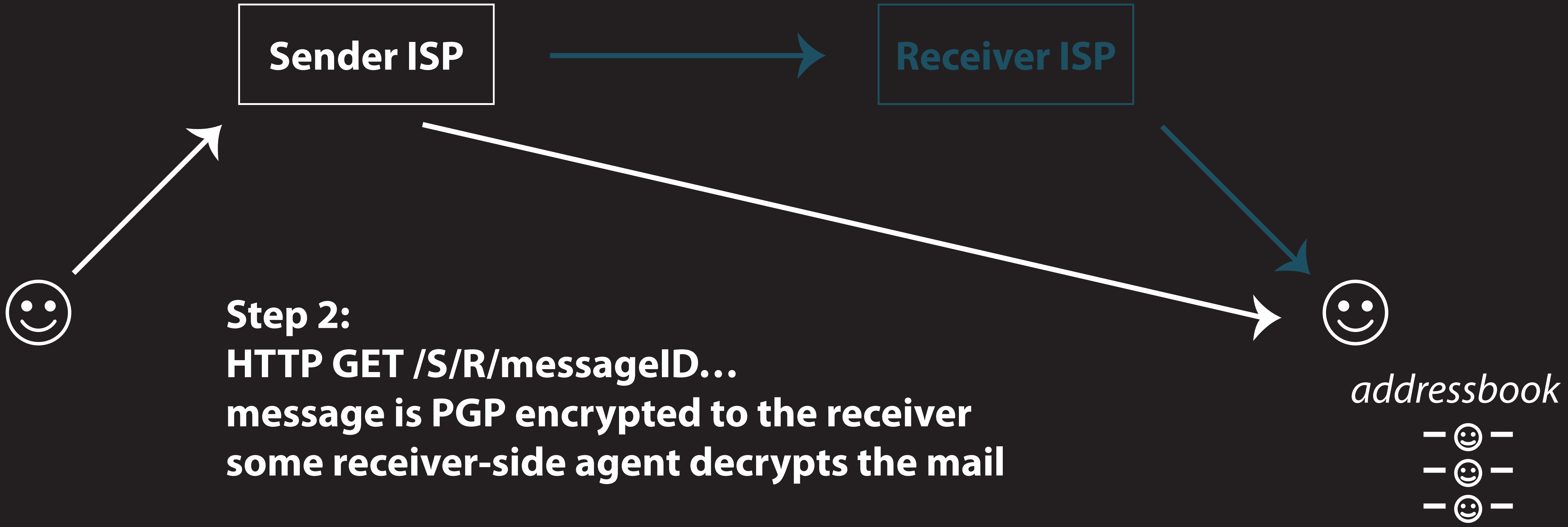
- given an outbox URL, retrieve messages

Core
HTTP GET
Full Crypto

Out of Scope

- where is the outbox URL kept?
- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



Step 2:
HTTP GET /S/R/messageID…
message is PGP encrypted to the receiver
some receiver-side agent decrypts the mail

Our Problem

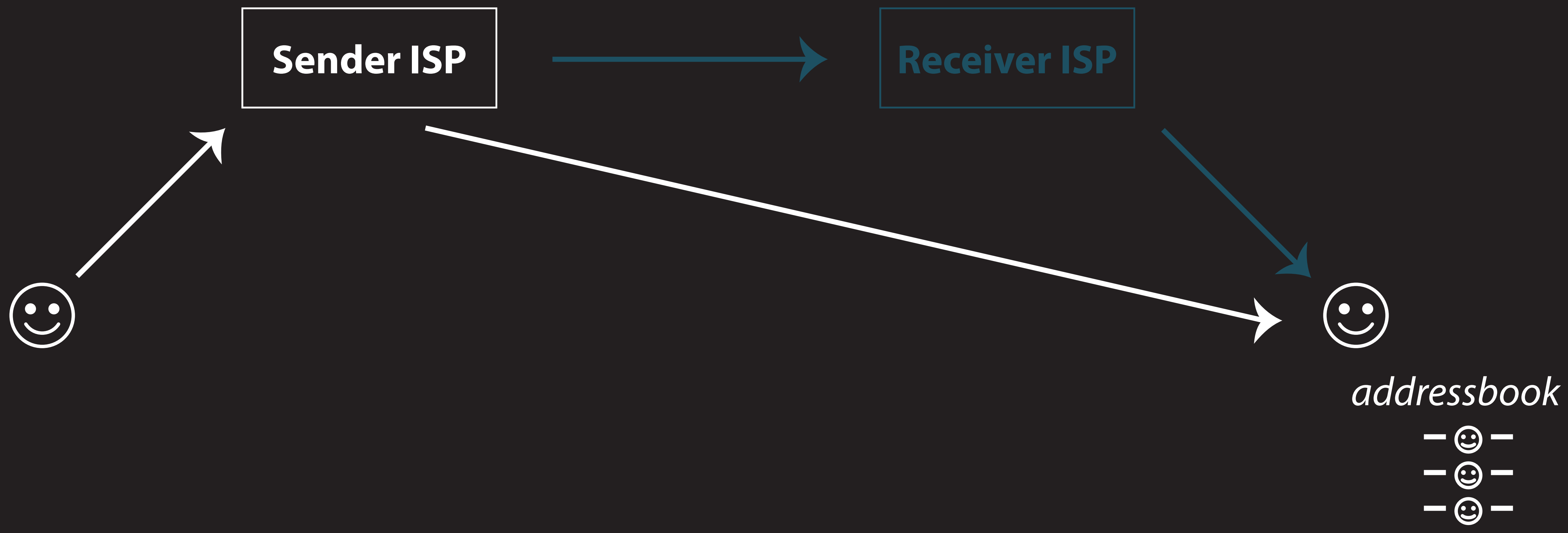
- **given an outbox URL, retrieve messages**

Core
HTTP GET
Full Crypto

Out of Scope

- **where did the outbox URL come from?**
- **how did the message get into the outbox?**
- **how does the sender server know the receiver's PGP key?**
- **aren't you reinventing SSL?**
- **who decrypts the message?**

Simplest Possible Implementation



Our Problem

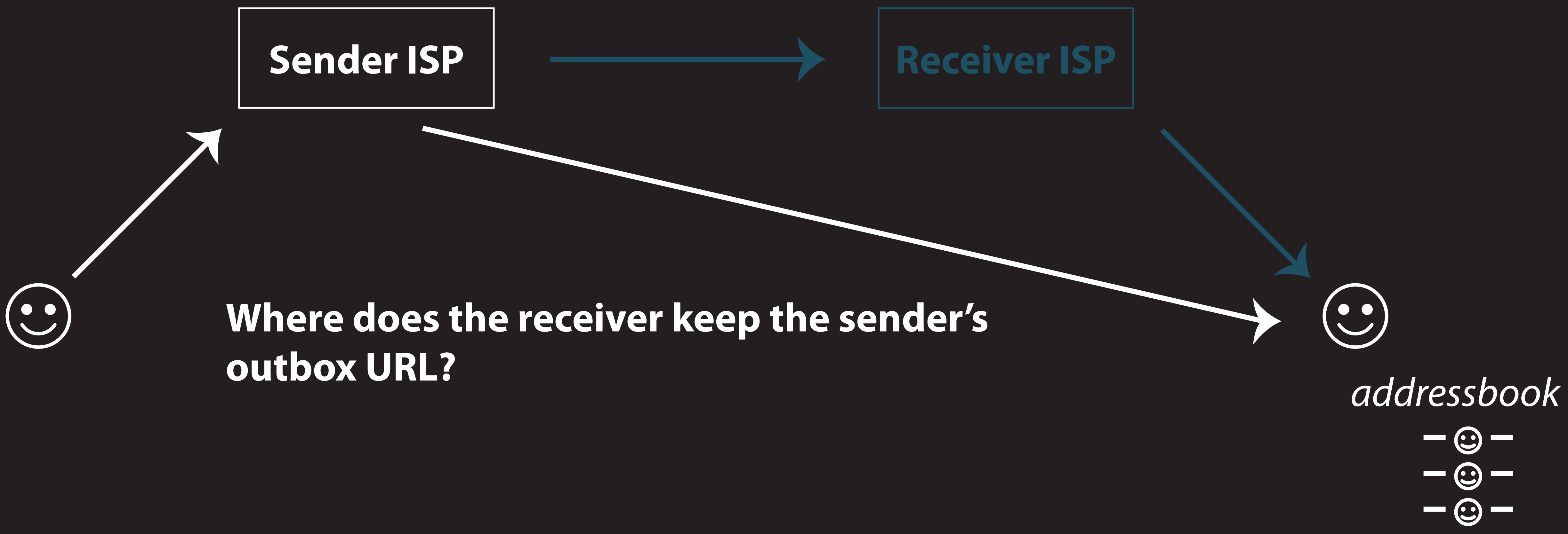
- given an outbox URL, retrieve messages

Core
HTTP GET
Full Crypto

Out of Scope

- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



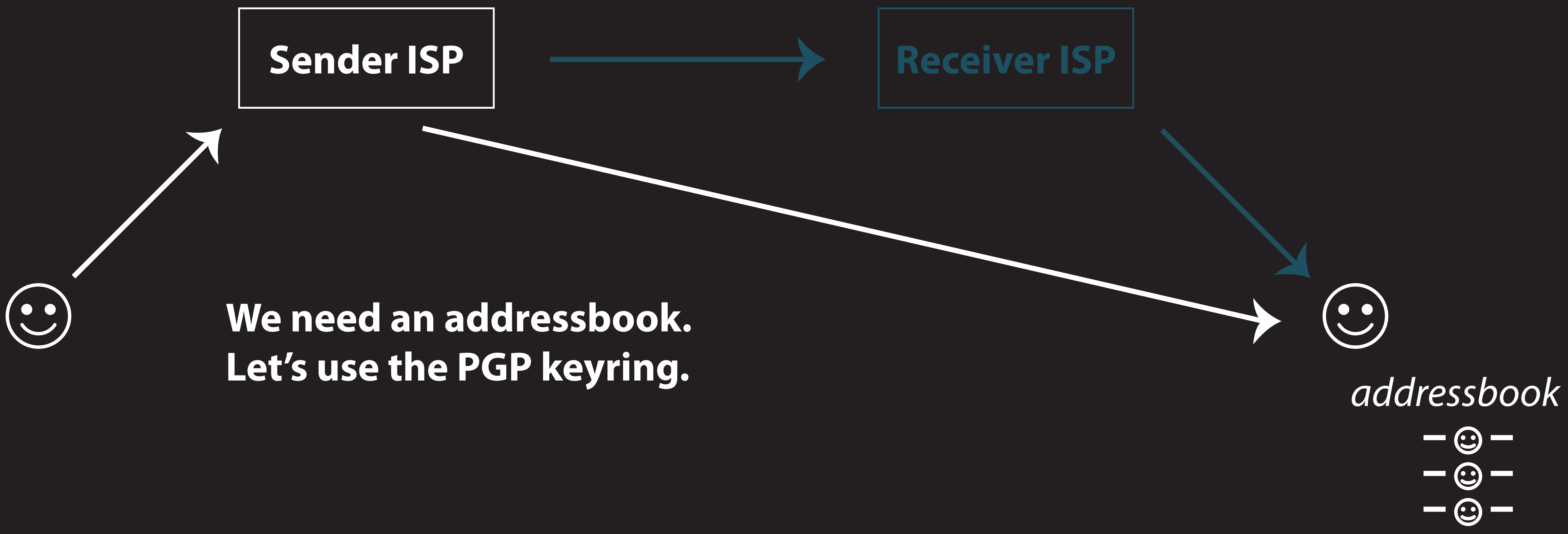
Where does the receiver keep the sender's outbox URL?

Our Problem
- **given an outbox URL, retrieve messages**
- **where is the outbox URL kept?**

Core
HTTP GET
Full Crypto

Out of Scope
- **where did the outbox URL come from?**
- **how did the message get into the outbox?**
- **how does the sender server know the receiver's PGP key?**
- **aren't you reinventing SSL?**
- **who decrypts the message?**

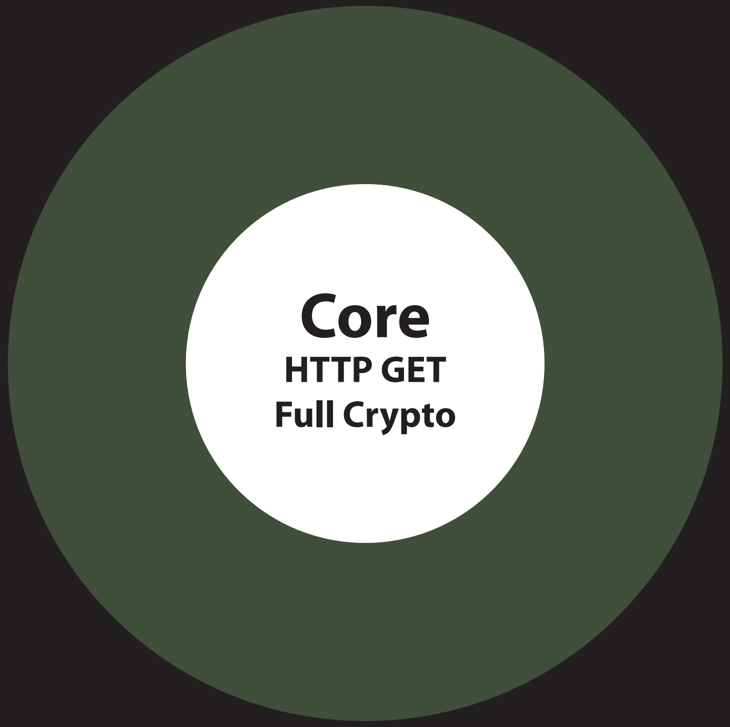
Simplest Possible Implementation



We need an addressbook.
Let's use the PGP keyring.

Our Problem

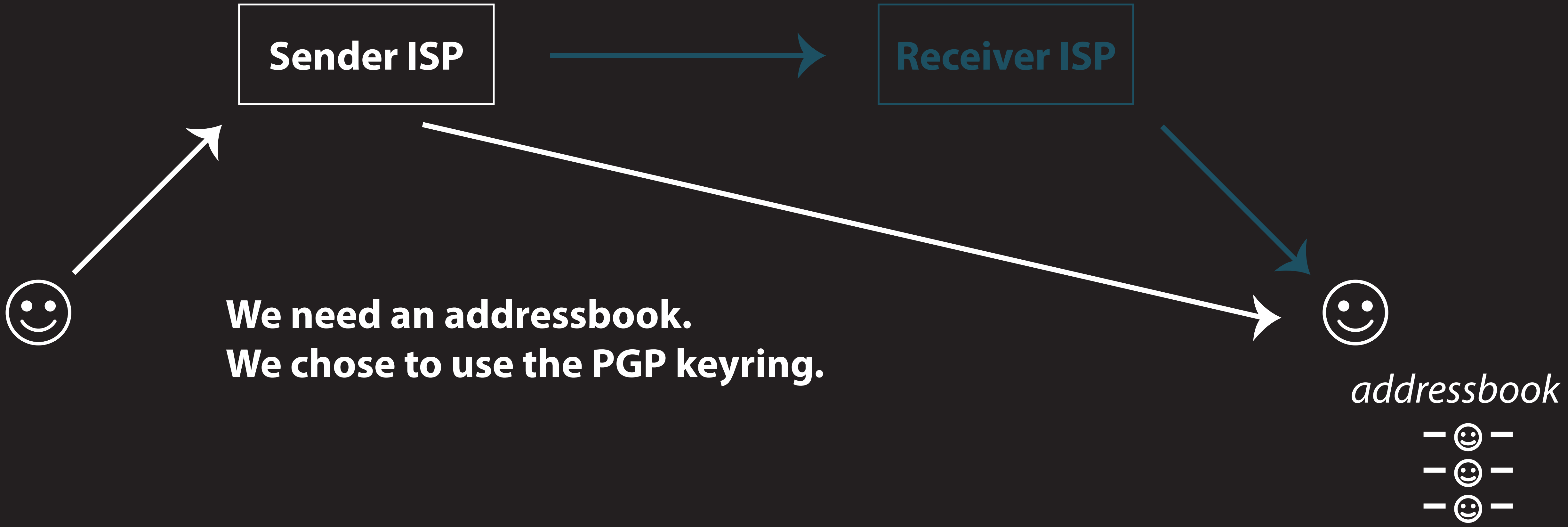
- **given an outbox URL, retrieve messages**
- **where is the outbox URL kept?**



Out of Scope

- **where did the outbox URL come from?**
- **how did the message get into the outbox?**
- **how does the sender server know the receiver's PGP key?**
- **aren't you reinventing SSL?**
- **who decrypts the message?**

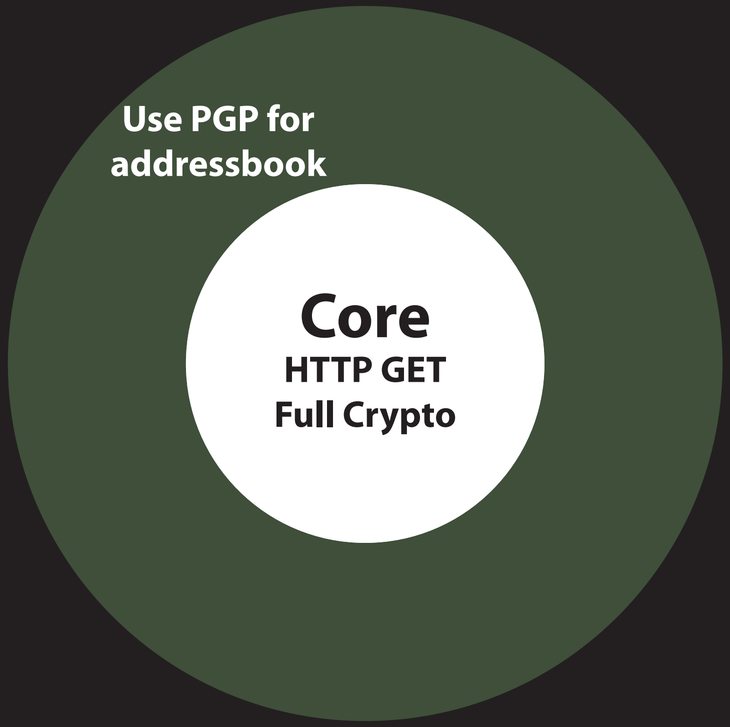
Simplest Possible Implementation



We need an addressbook.
We chose to use the PGP keyring.

Our Problem

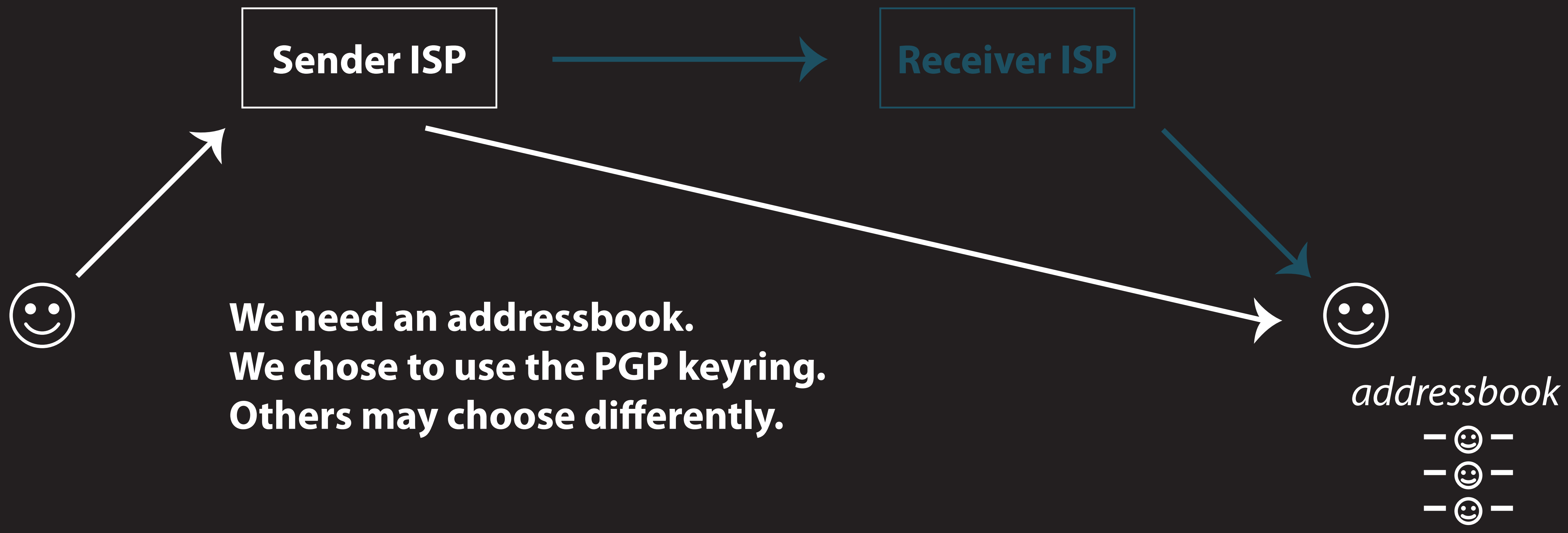
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs



Out of Scope

- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

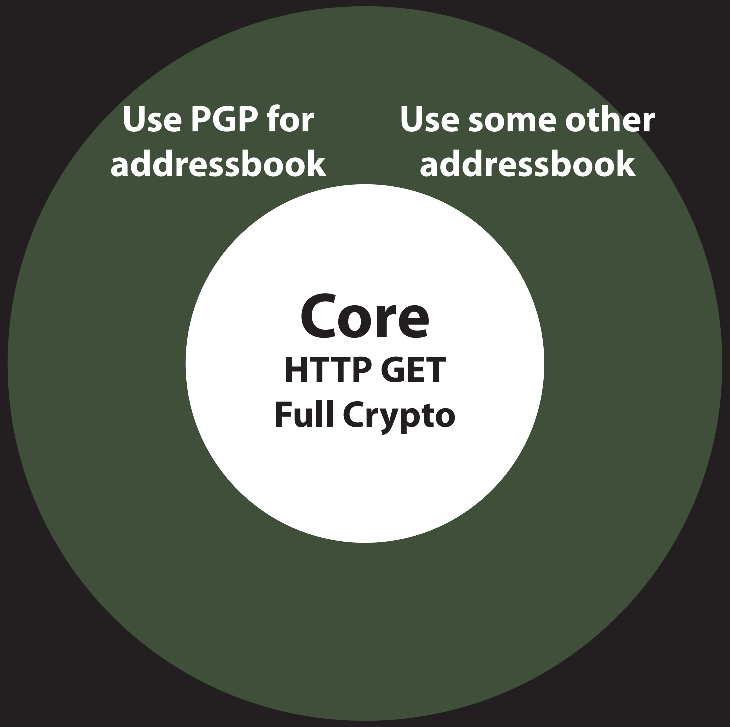
Simplest Possible Implementation



We need an addressbook.
We chose to use the PGP keyring.
Others may choose differently.

Our Problem

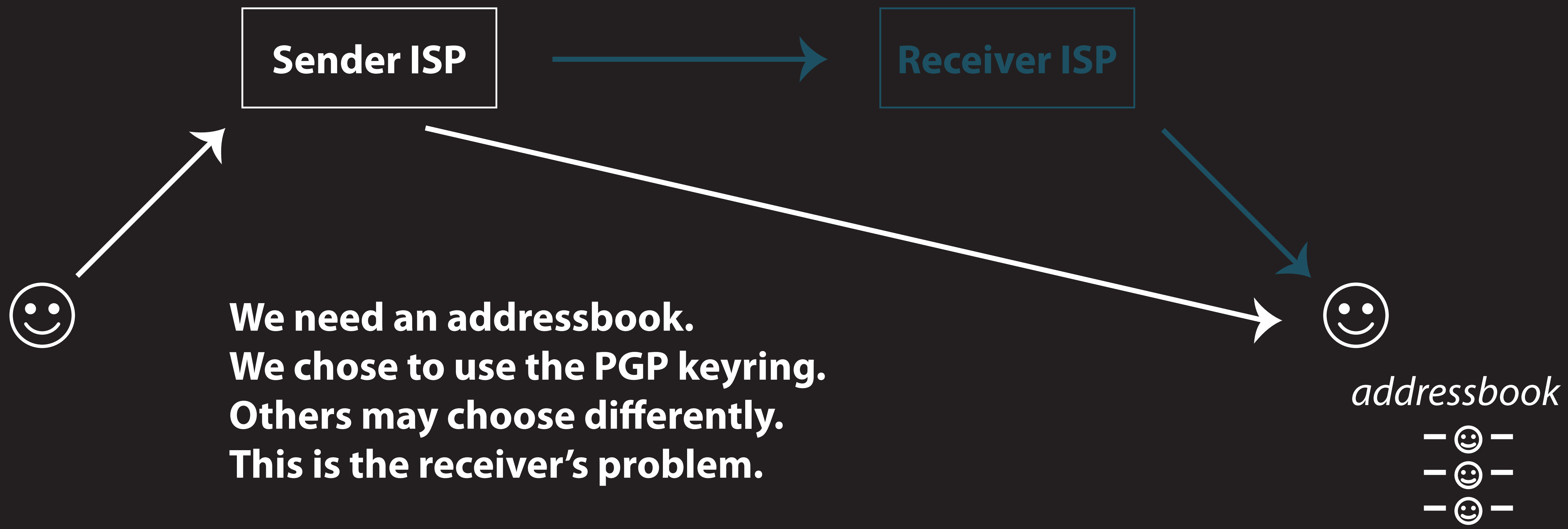
- **given an outbox URL, retrieve messages**
- **PGP keyring contains the URLs**



Out of Scope

- **where did the outbox URL come from?**
- **how did the message get into the outbox?**
- **how does the sender server know the receiver's PGP key?**
- **aren't you reinventing SSL?**
- **who decrypts the message?**

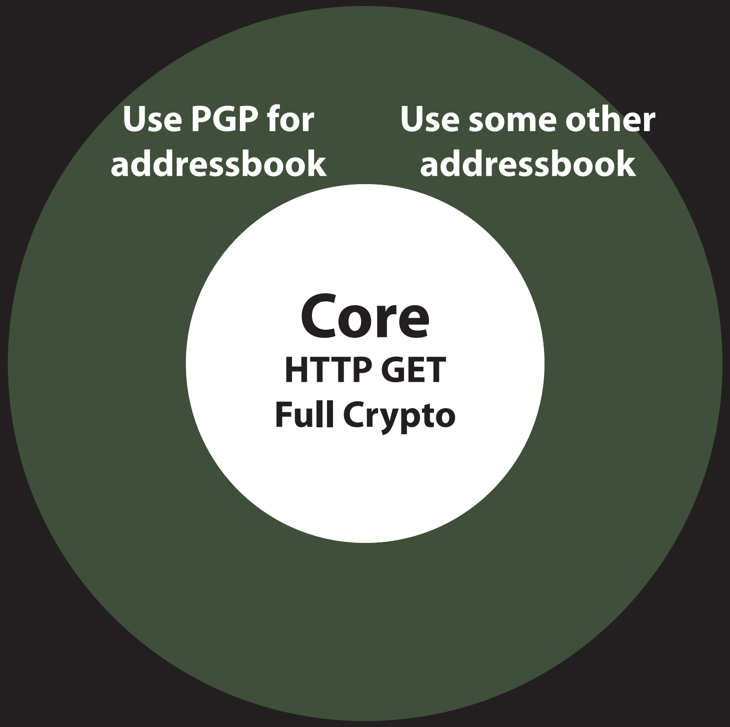
Simplest Possible Implementation



We need an addressbook.
We chose to use the PGP keyring.
Others may choose differently.
This is the receiver's problem.

Our Problem

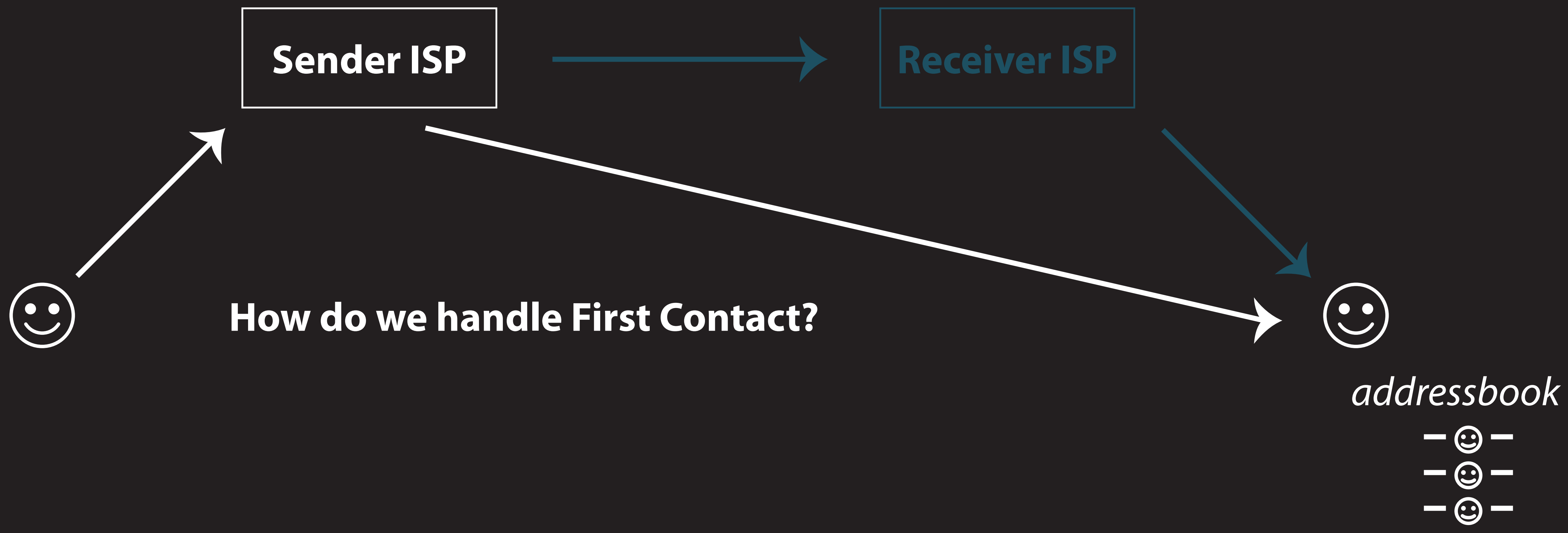
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs



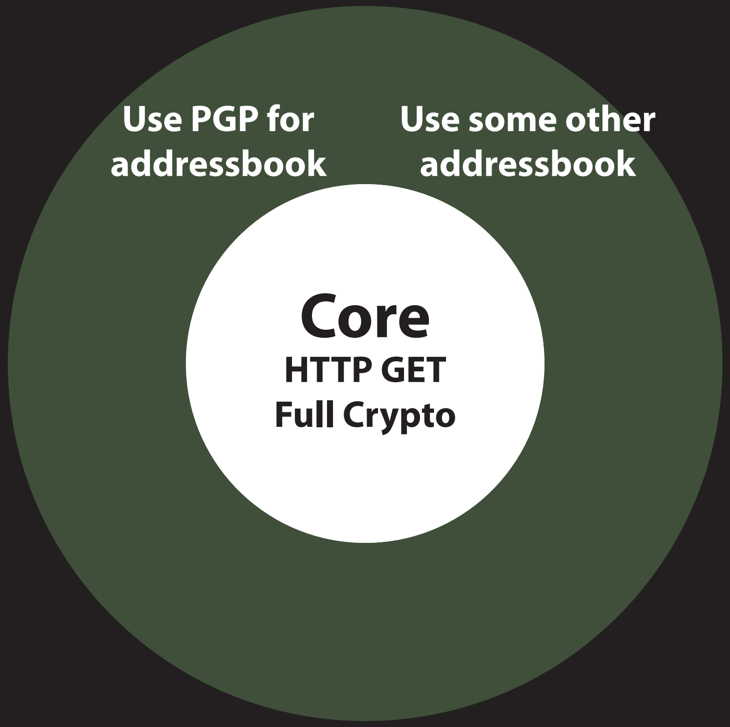
Out of Scope

- where did the outbox URL come from?
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation

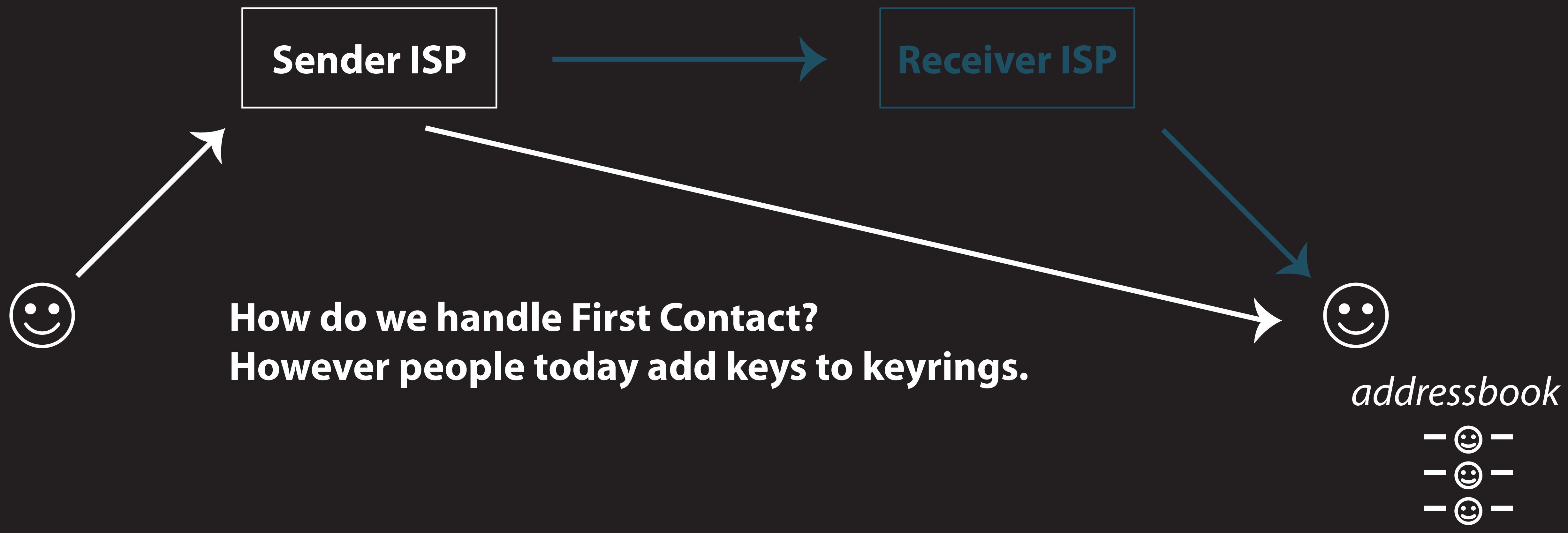


Our Problem
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- where did the outbox URL come from?



Out of Scope
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

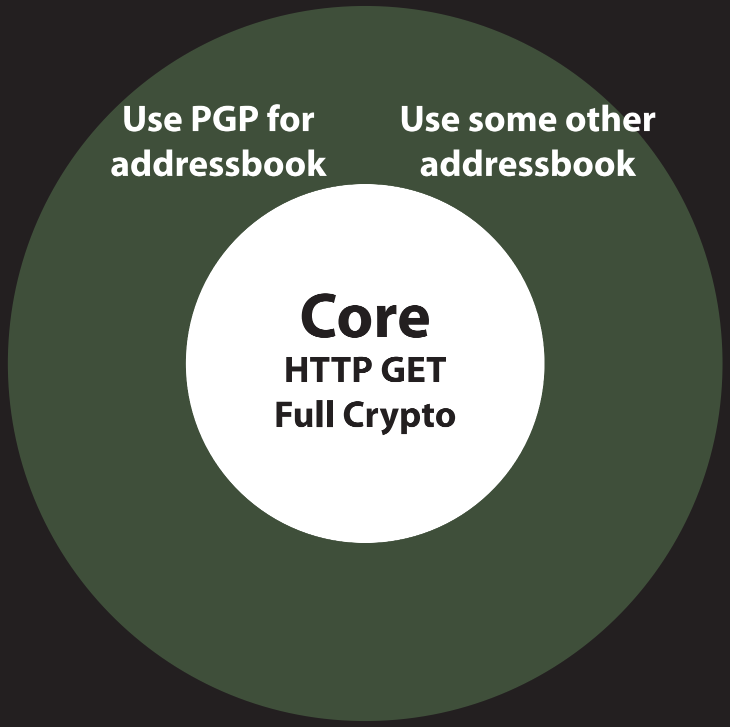
Simplest Possible Implementation



How do we handle First Contact?
However people today add keys to keyrings.

Our Problem

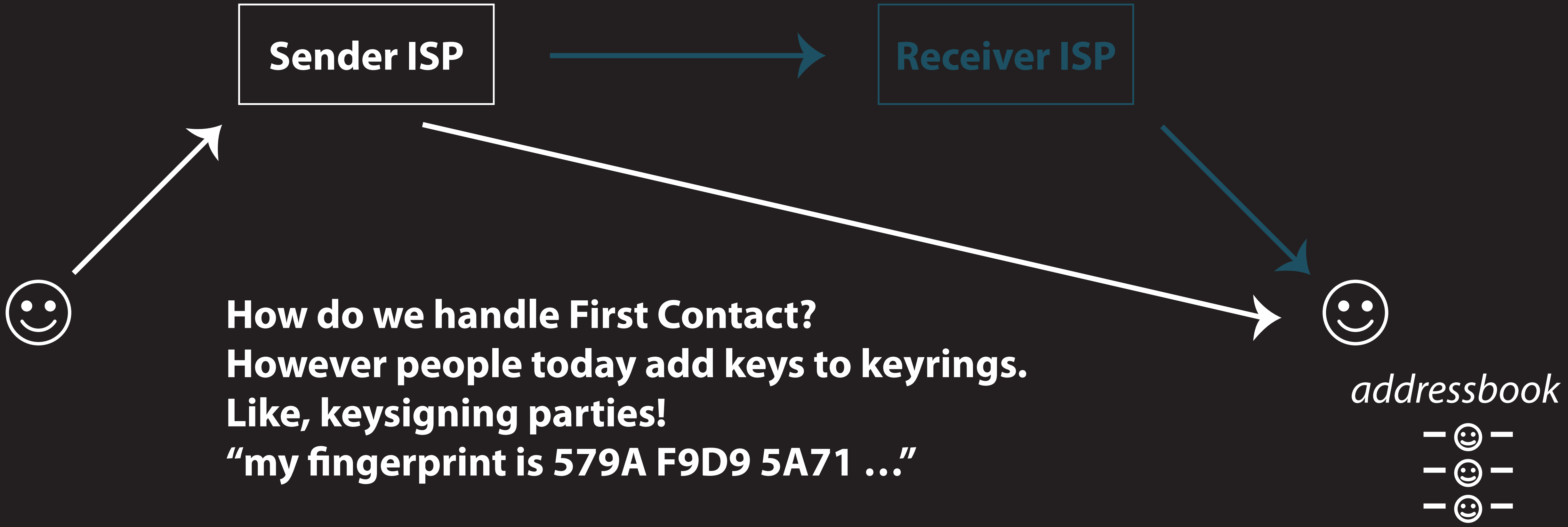
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- where did the outbox URL come from?



Out of Scope

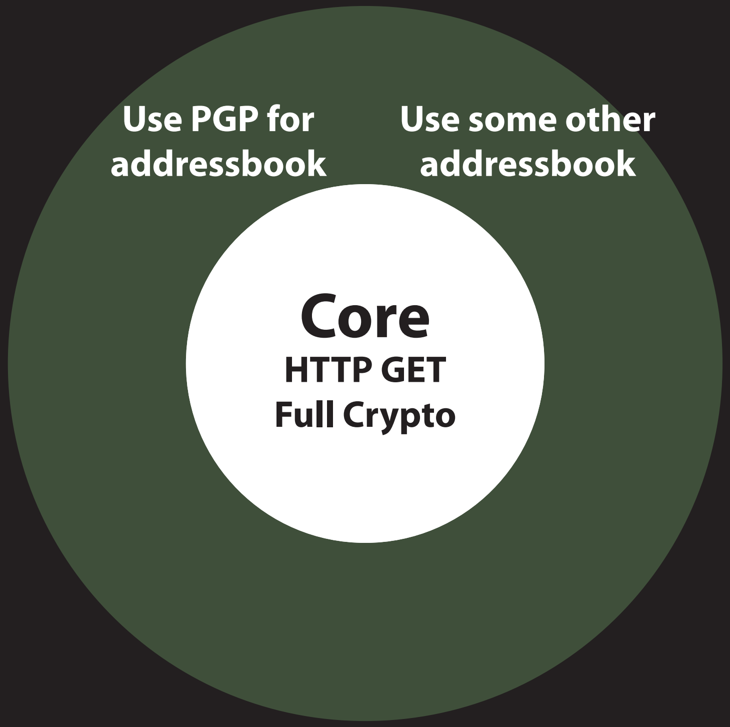
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



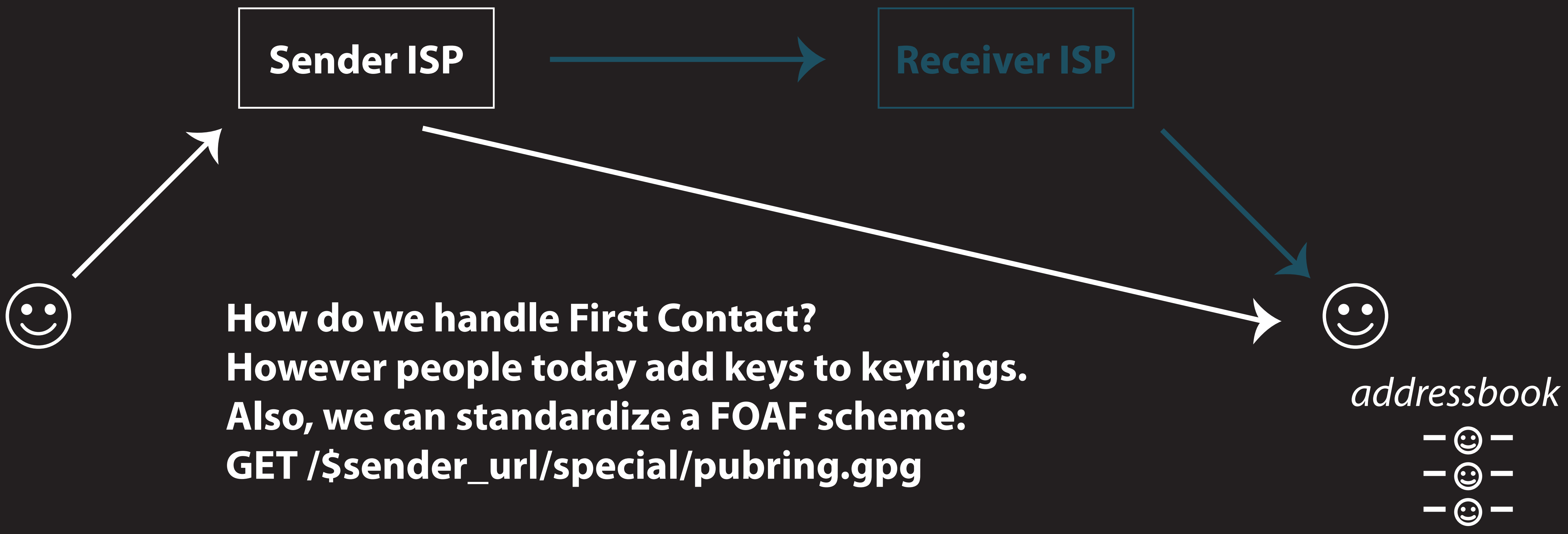
How do we handle First Contact?
However people today add keys to keyrings.
Like, keysigning parties!
“my fingerprint is 579A F9D9 5A71 …”

Our Problem
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons



Out of Scope
- how did the message get into the outbox?
- how does the sender server know the receiver’s PGP key?
- aren’t you reinventing SSL?
- who decrypts the message?

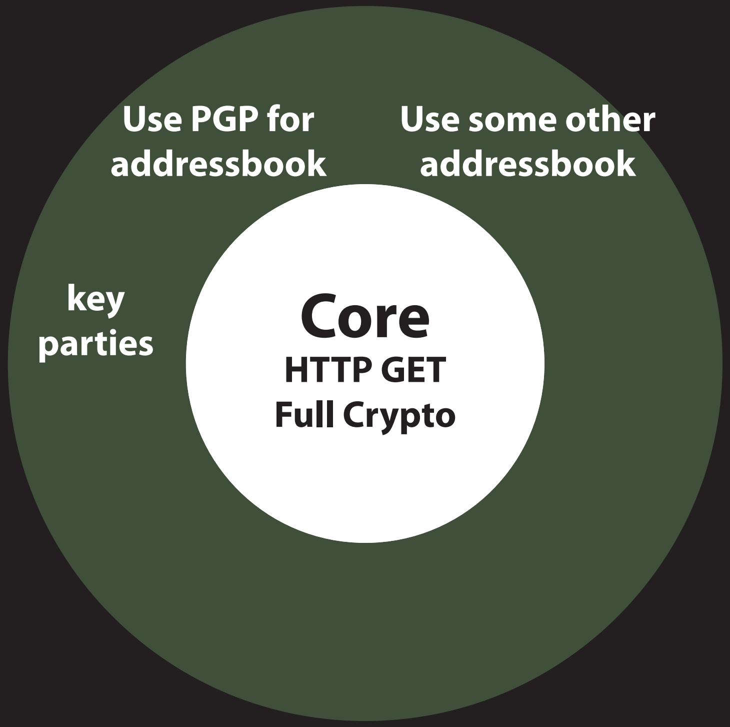
Simplest Possible Implementation



How do we handle First Contact?
However people today add keys to keyrings.
Also, we can standardize a FOAF scheme:
GET /$sender_url/special/pubring.gpg

Our Problem

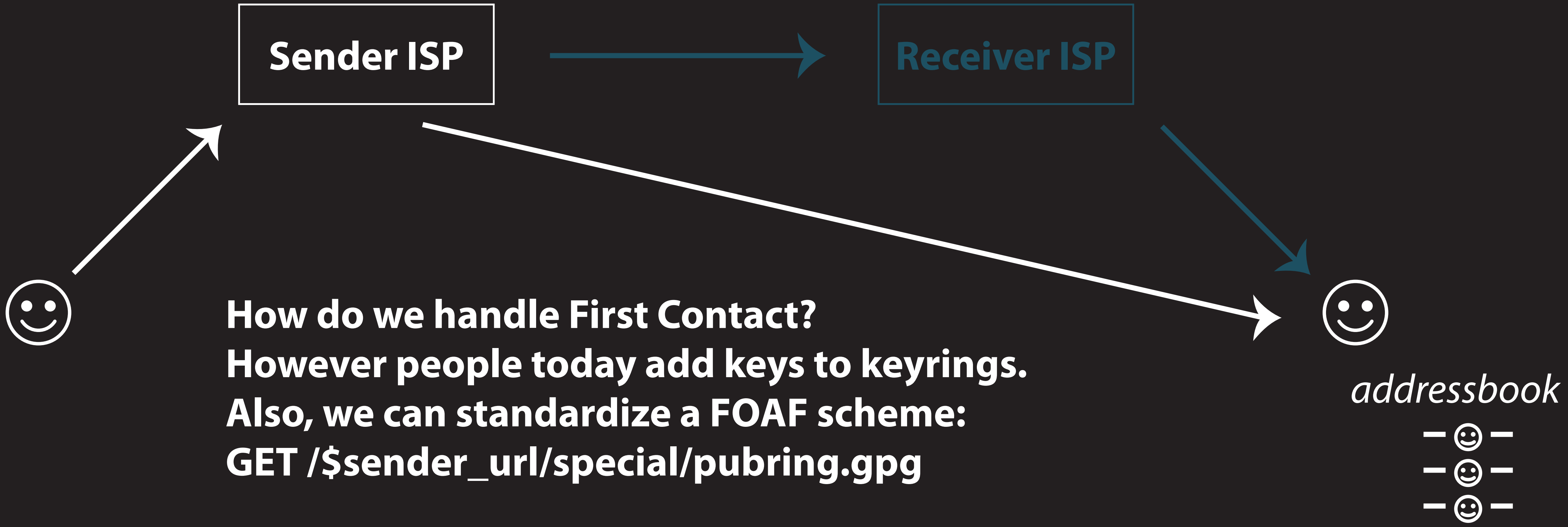
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background



Out of Scope

- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

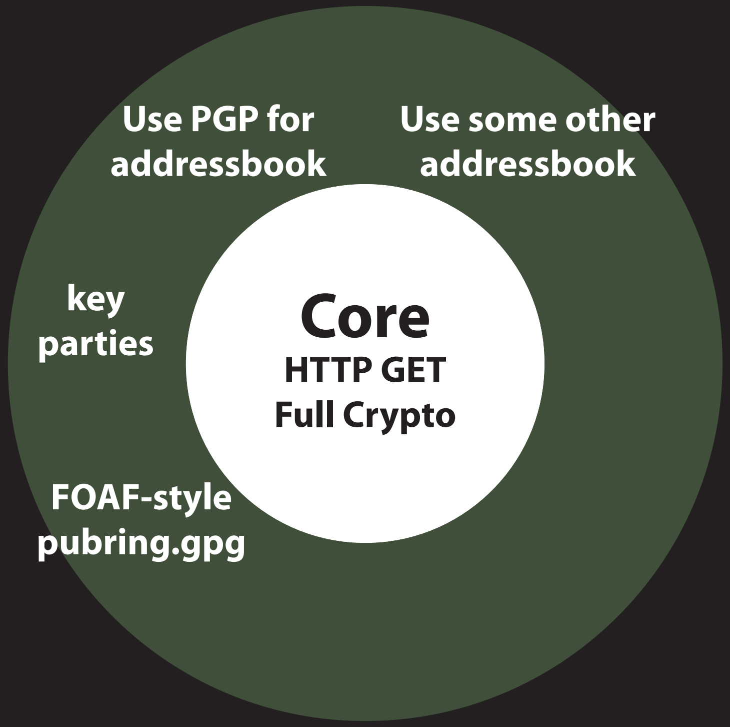
Simplest Possible Implementation



How do we handle First Contact?
However people today add keys to keyrings.
Also, we can standardize a FOAF scheme:
GET /$sender_url/special/pubring.gpg

Our Problem

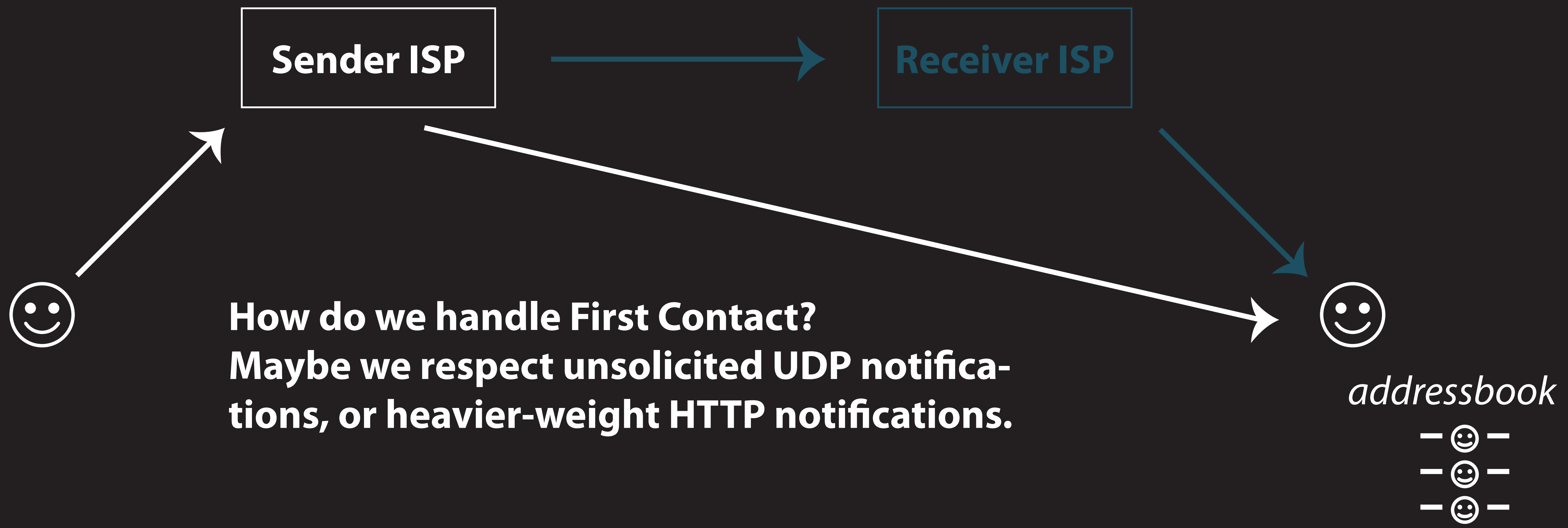
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background



Out of Scope

- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

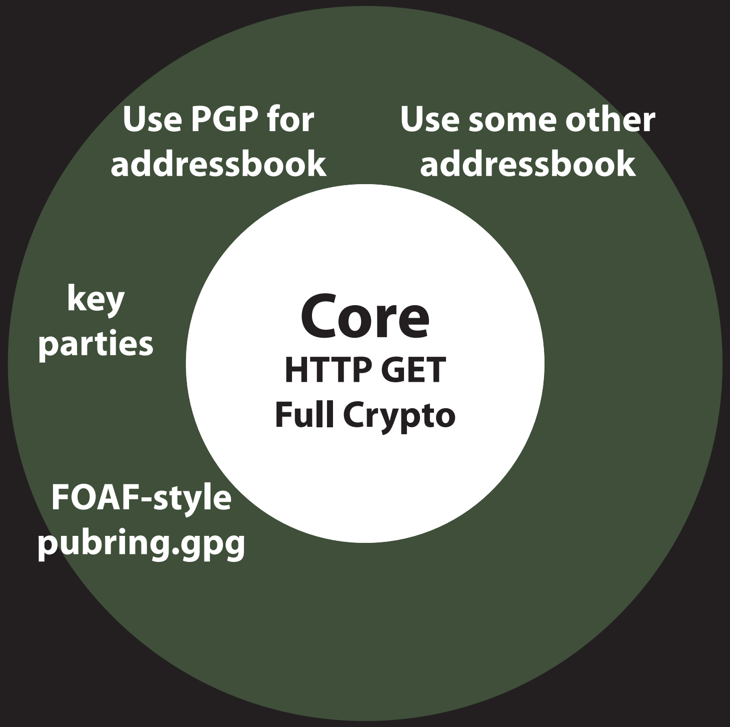
Simplest Possible Implementation



How do we handle First Contact?
Maybe we respect unsolicited UDP notifications, or heavier-weight HTTP notifications.

Our Problem

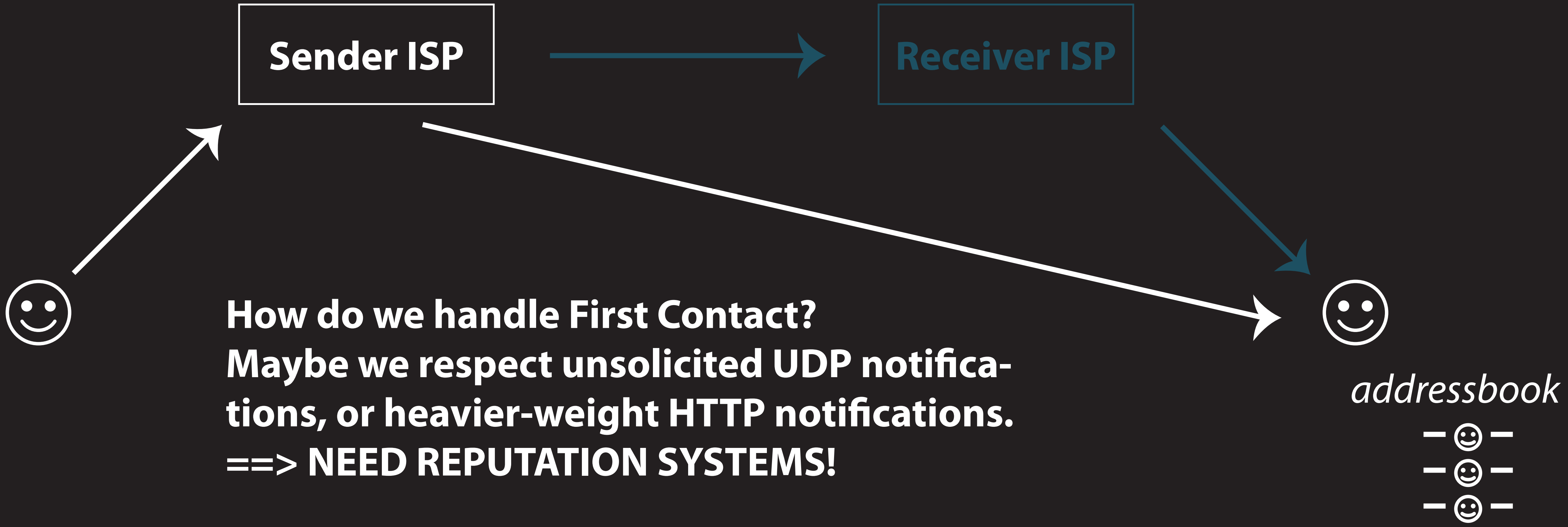
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background



Out of Scope

- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

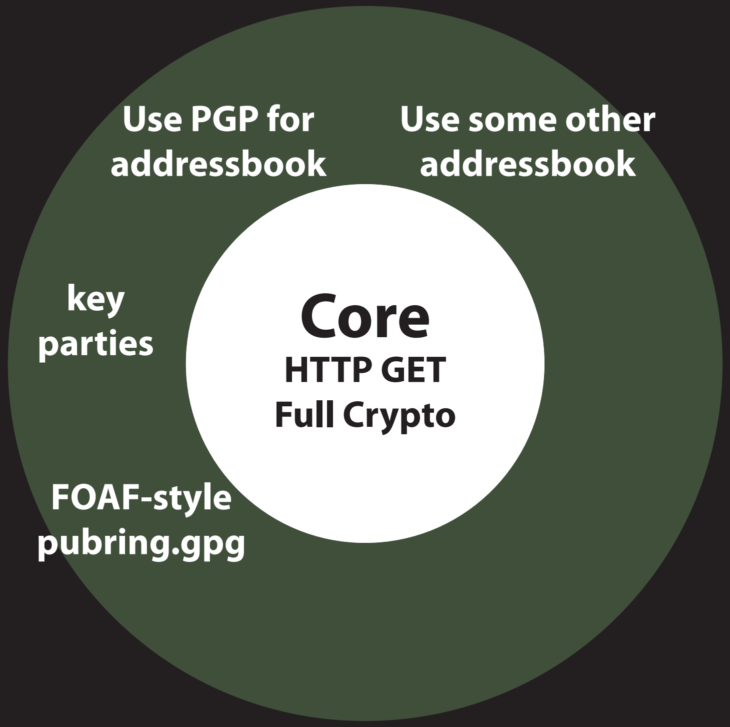
Simplest Possible Implementation



How do we handle First Contact?
Maybe we respect unsolicited UDP notifications, or heavier-weight HTTP notifications.
==> NEED REPUTATION SYSTEMS!

Our Problem

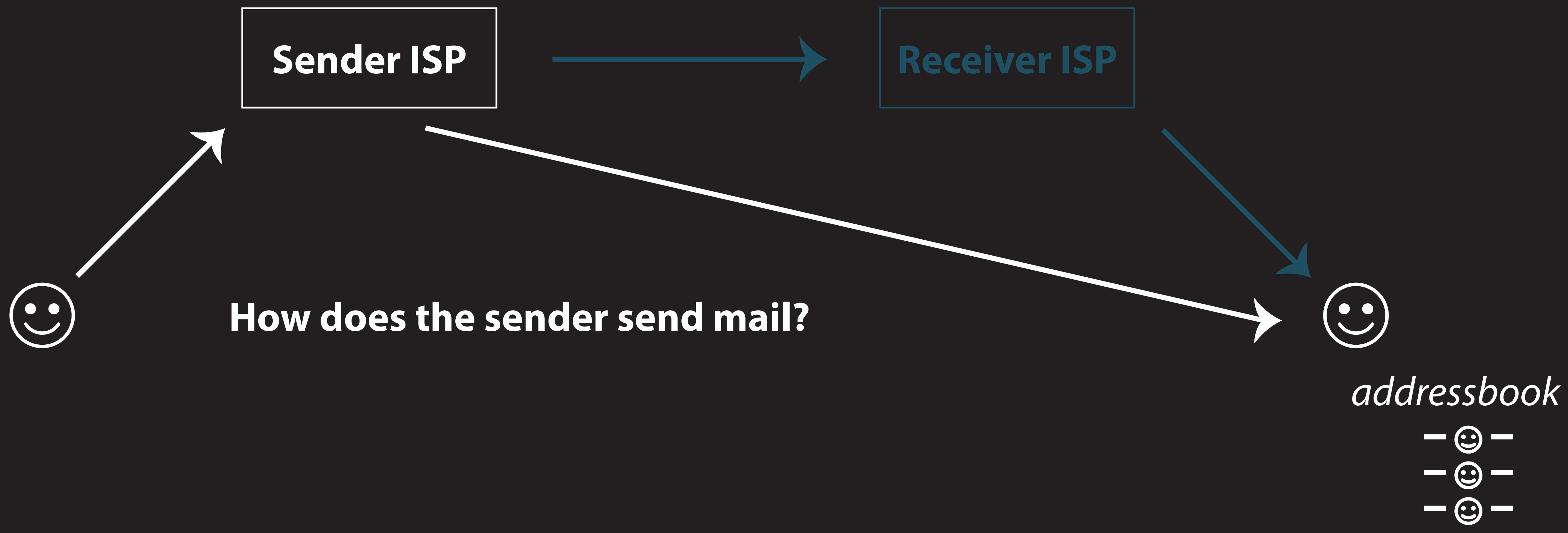
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background



Out of Scope

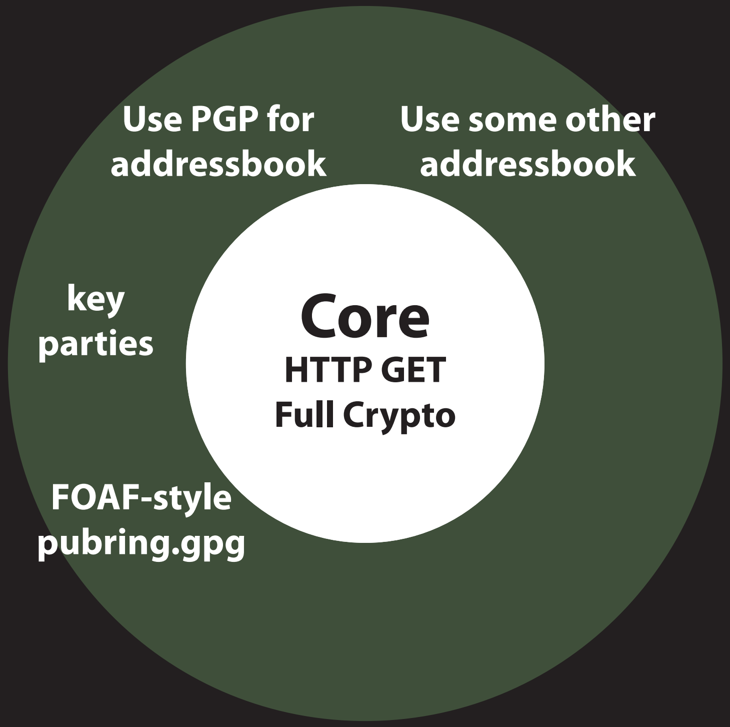
- how did the message get into the outbox?
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



Our Problem

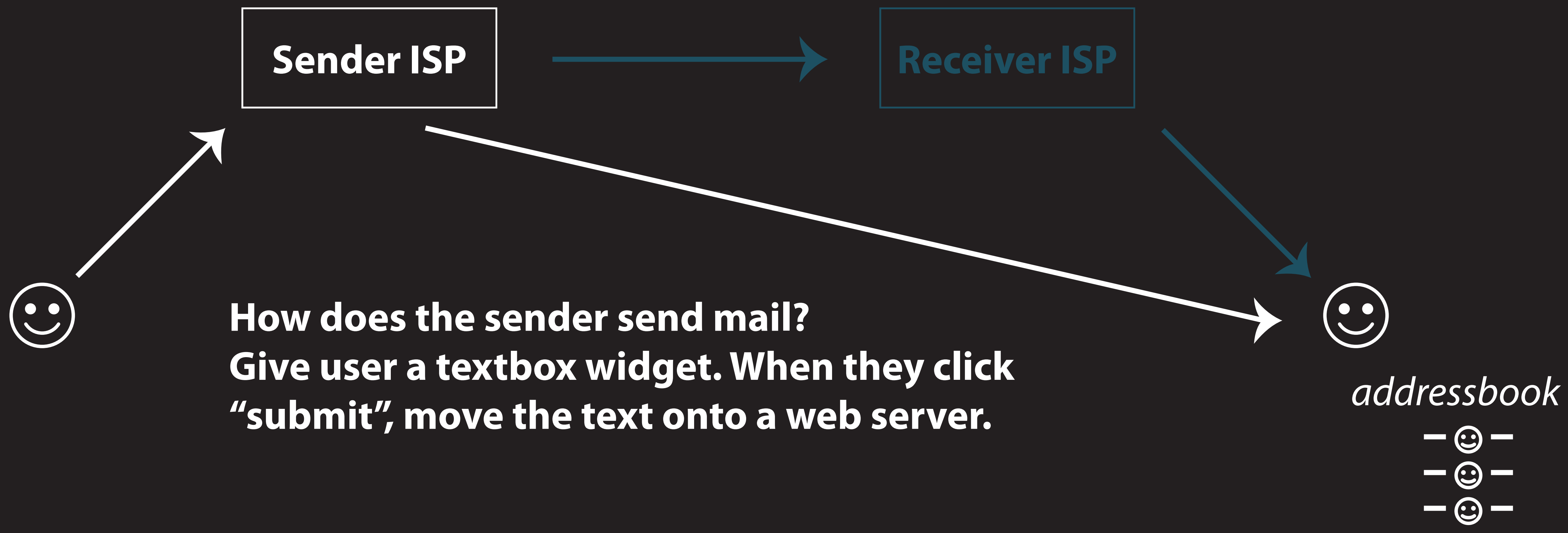
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- how did the message get into the outbox?



Out of Scope

- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

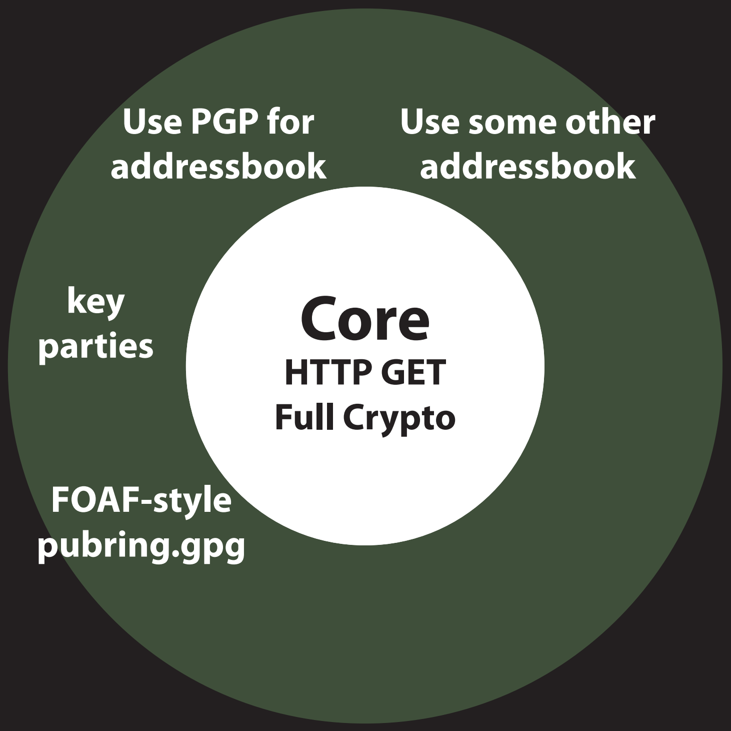
Simplest Possible Implementation



How does the sender send mail?
Give user a textbox widget. When they click "submit", move the text onto a web server.

Our Problem

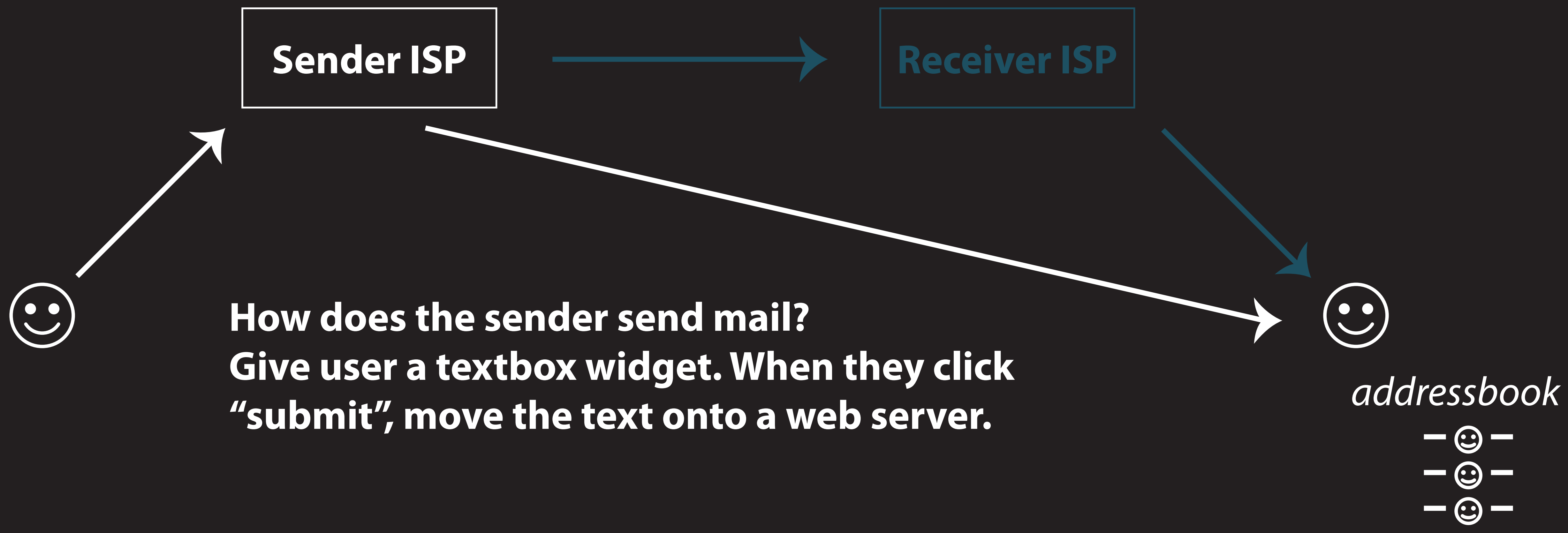
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- how did the message get into the outbox?



Out of Scope

- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

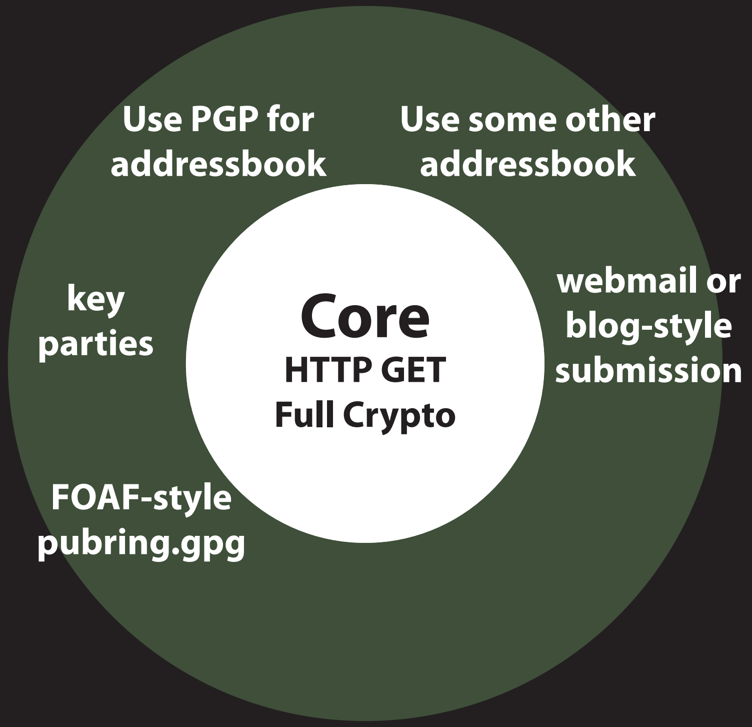
Simplest Possible Implementation



How does the sender send mail?
Give user a textbox widget. When they click "submit", move the text onto a web server.

Our Problem

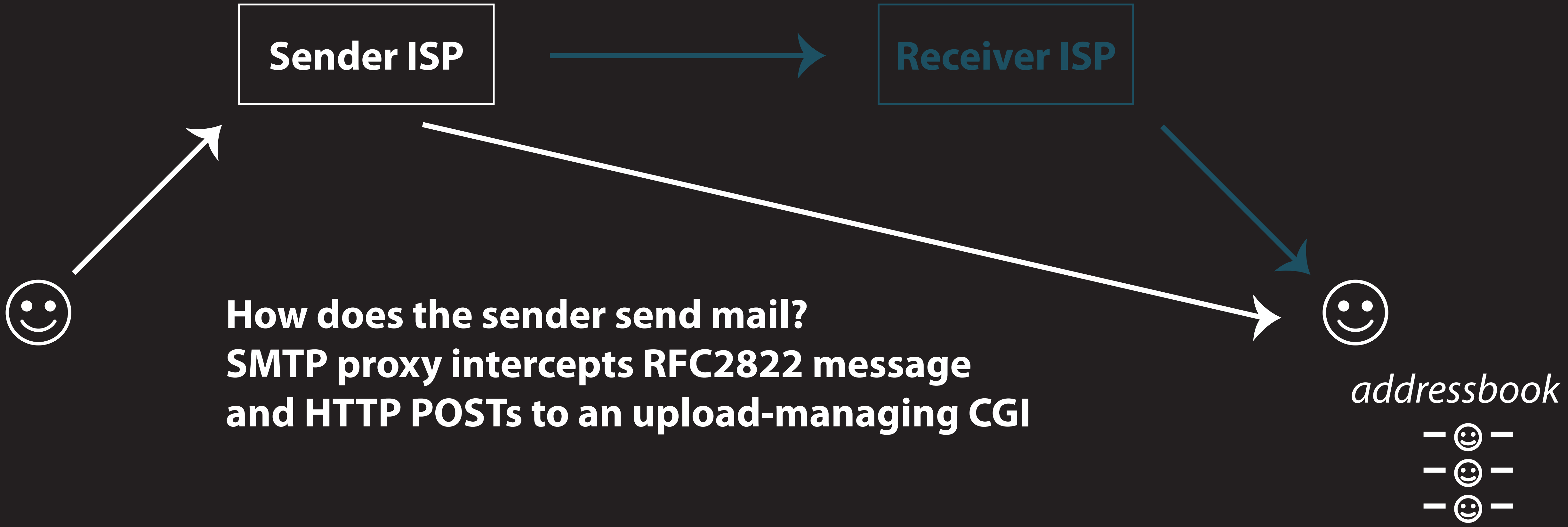
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- how did the message get into the outbox?



Out of Scope

- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

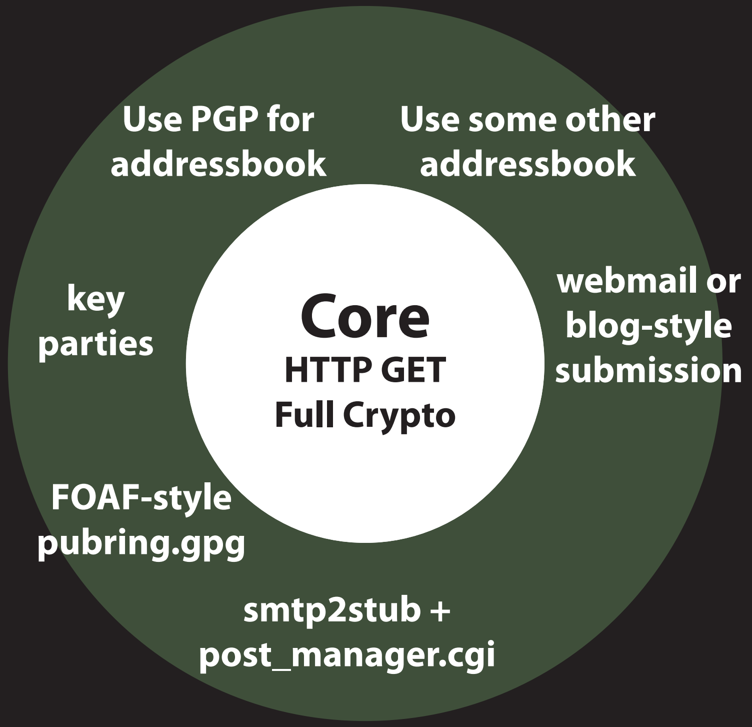
Simplest Possible Implementation



How does the sender send mail?
SMTP proxy intercepts RFC2822 message and HTTP POSTs to an upload-managing CGI

Our Problem

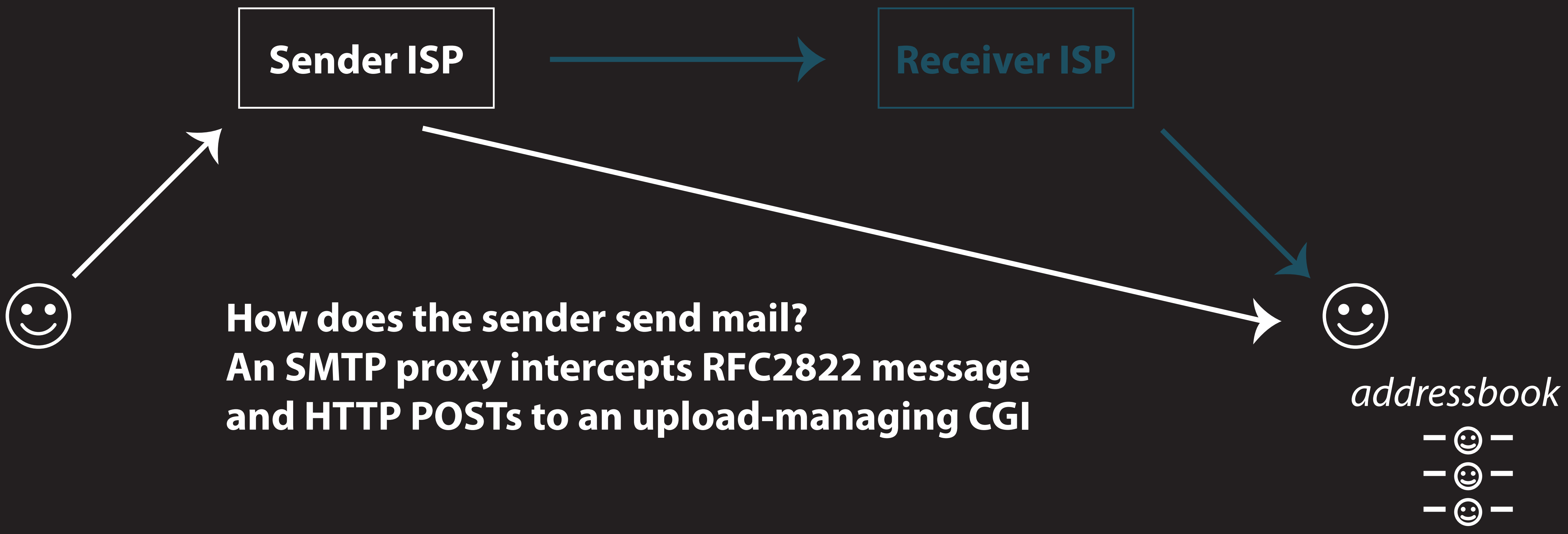
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
key parties
webmail or blog-style submission
FOAF-style pubring.gpg
smtp2stub + post_manager.cgi

Out of Scope

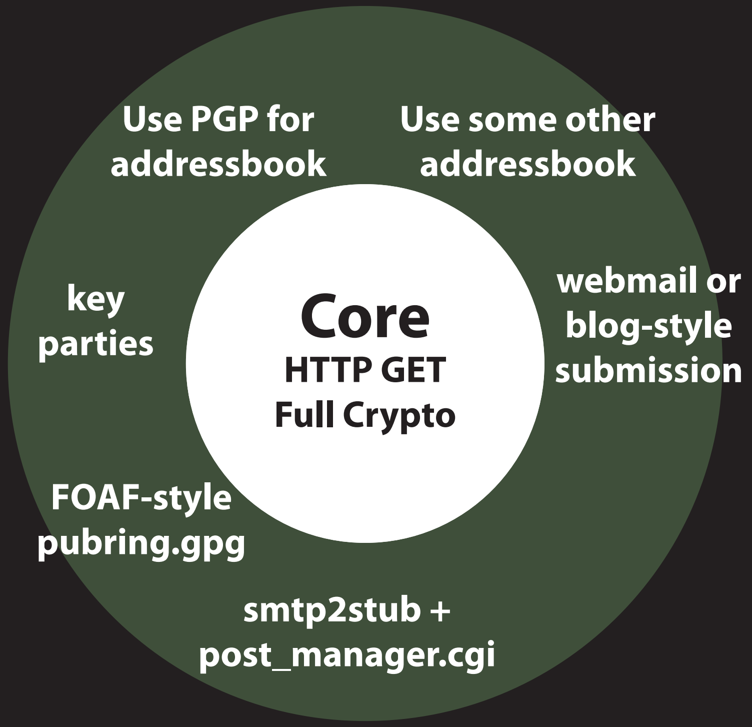
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



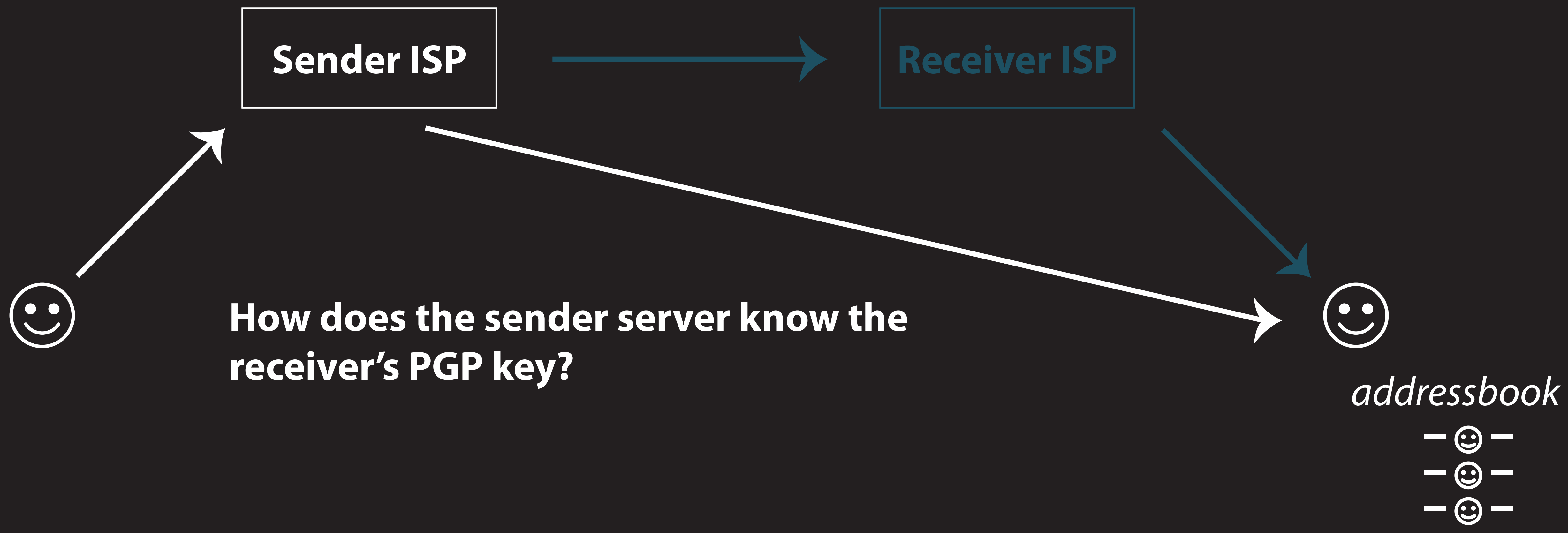
How does the sender send mail?
An SMTP proxy intercepts RFC2822 message and HTTP POSTs to an upload-managing CGI

Our Problem
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
key parties
FOAF-style pubring.gpg
smtp2stub + post_manager.cgi

Out of Scope
- how does the sender server know the receiver's PGP key?
- aren't you reinventing SSL?
- who decrypts the message?

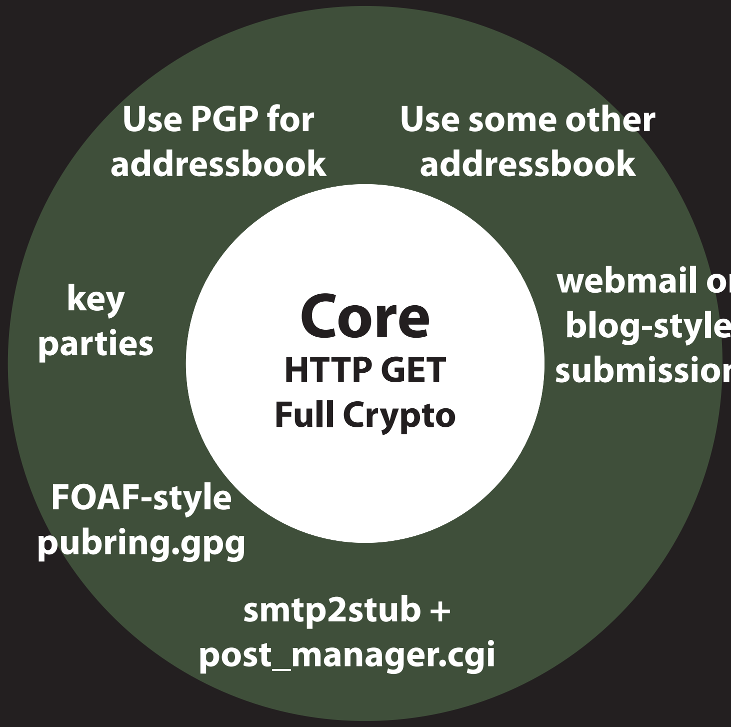
Simplest Possible Implementation



How does the sender server know the receiver's PGP key?

Our Problem

- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI
- how does the sender server know the receiver's PGP key?

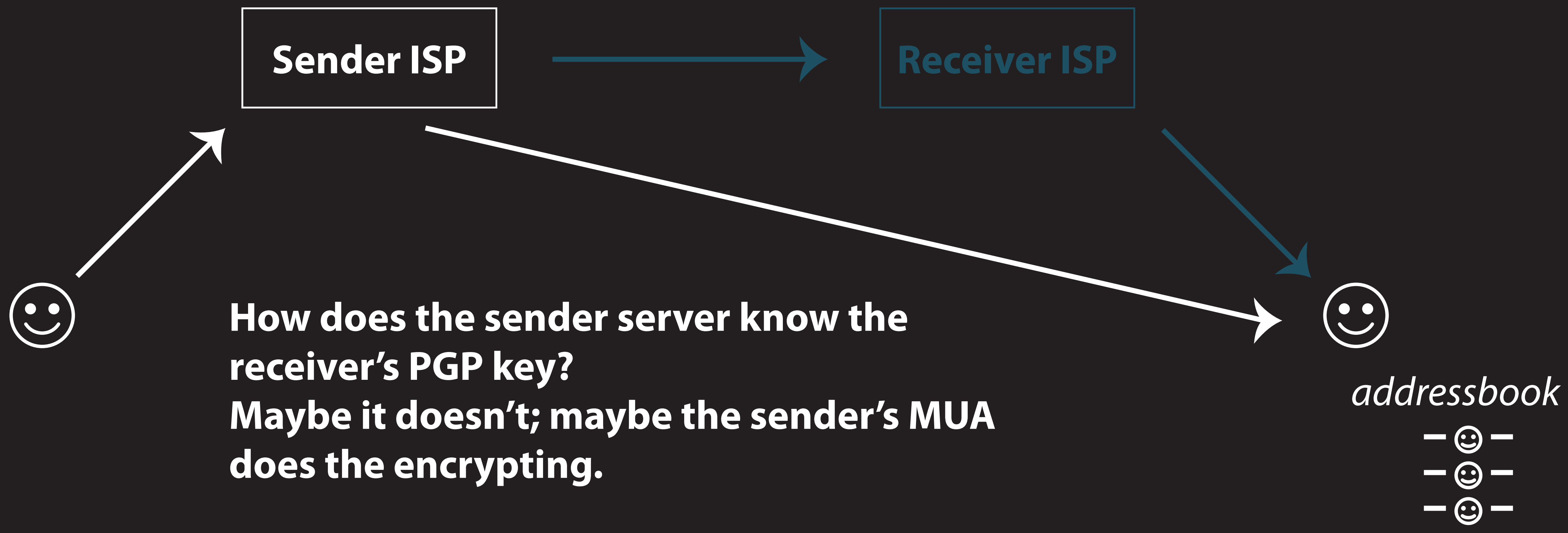


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- aren't you reinventing SSL?
- who decrypts the message?

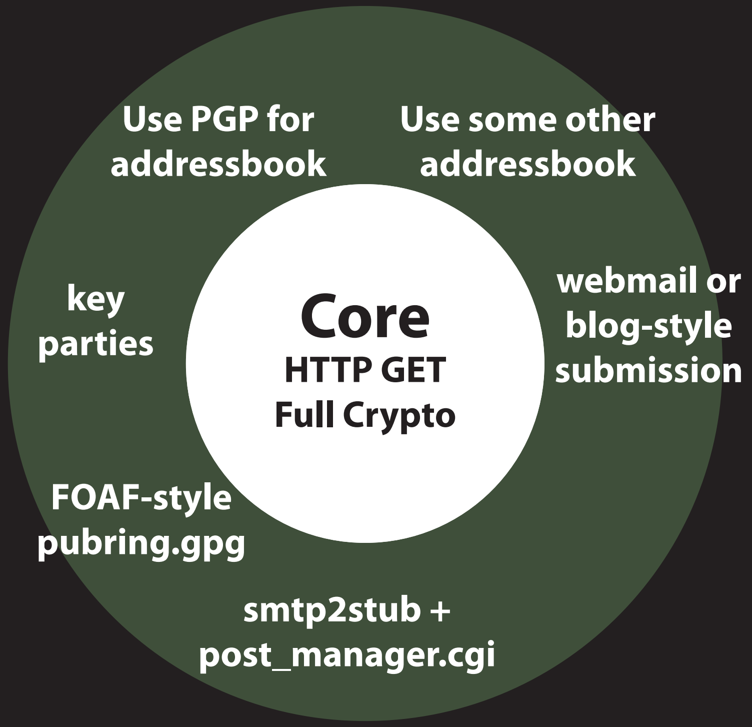
Simplest Possible Implementation



How does the sender server know the receiver's PGP key?
Maybe it doesn't; maybe the sender's MUA does the encrypting.

Our Problem

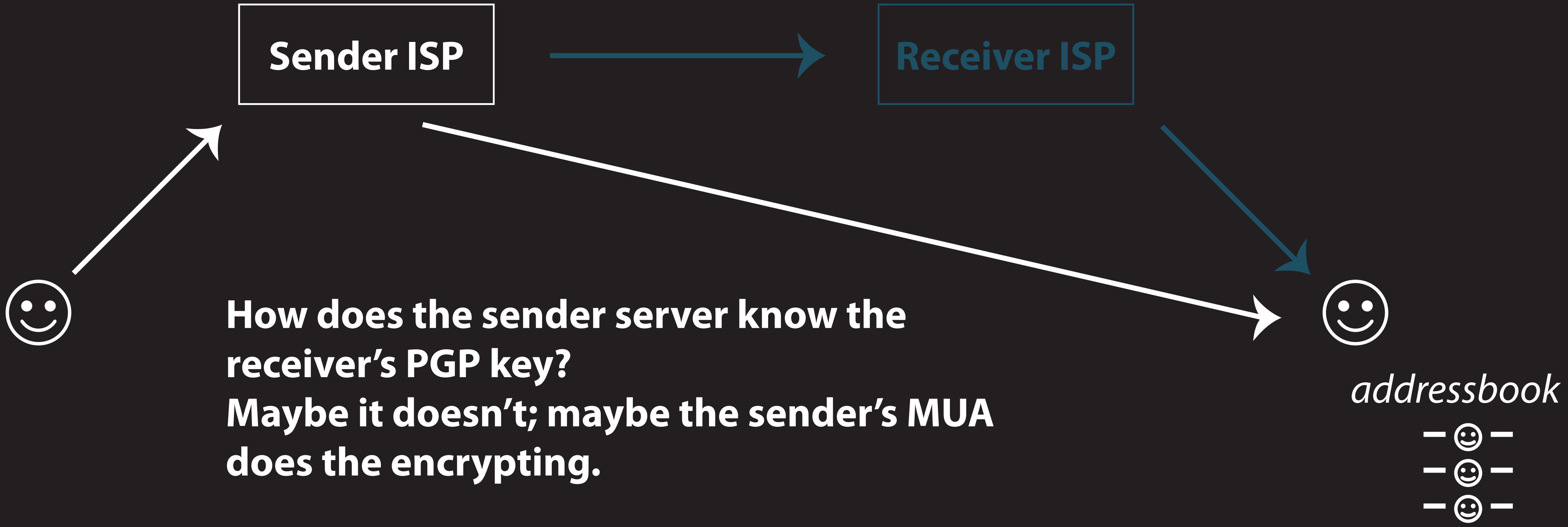
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI
- how does the sender server know the receiver's PGP key?


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
key parties
FOAF-style pubring.gpg
smtp2stub + post_manager.cgi

Out of Scope

- aren't you reinventing SSL?
- who decrypts the message?

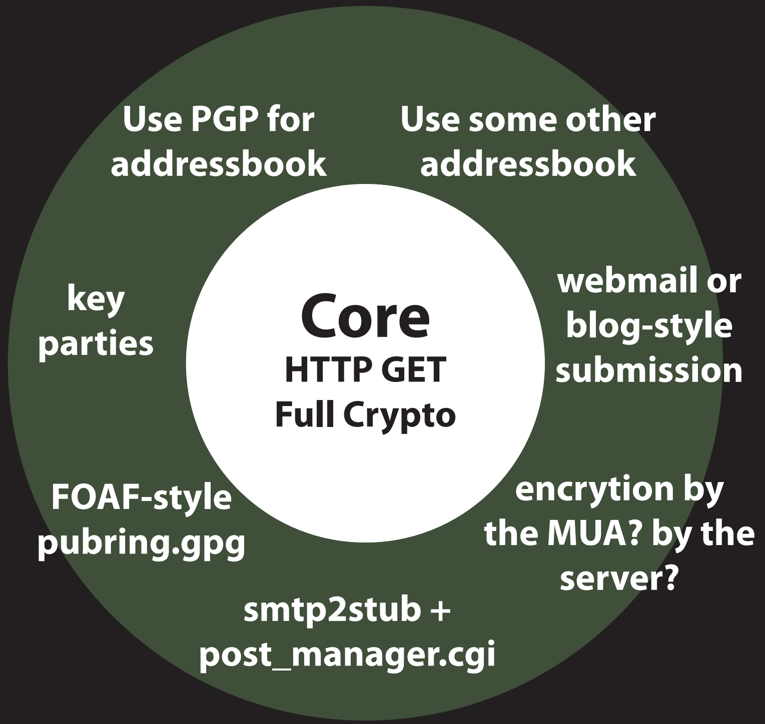
Simplest Possible Implementation



How does the sender server know the receiver's PGP key?
Maybe it doesn't; maybe the sender's MUA does the encrypting.

Our Problem

- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI
- maybe the sender MUA does the encrypting.

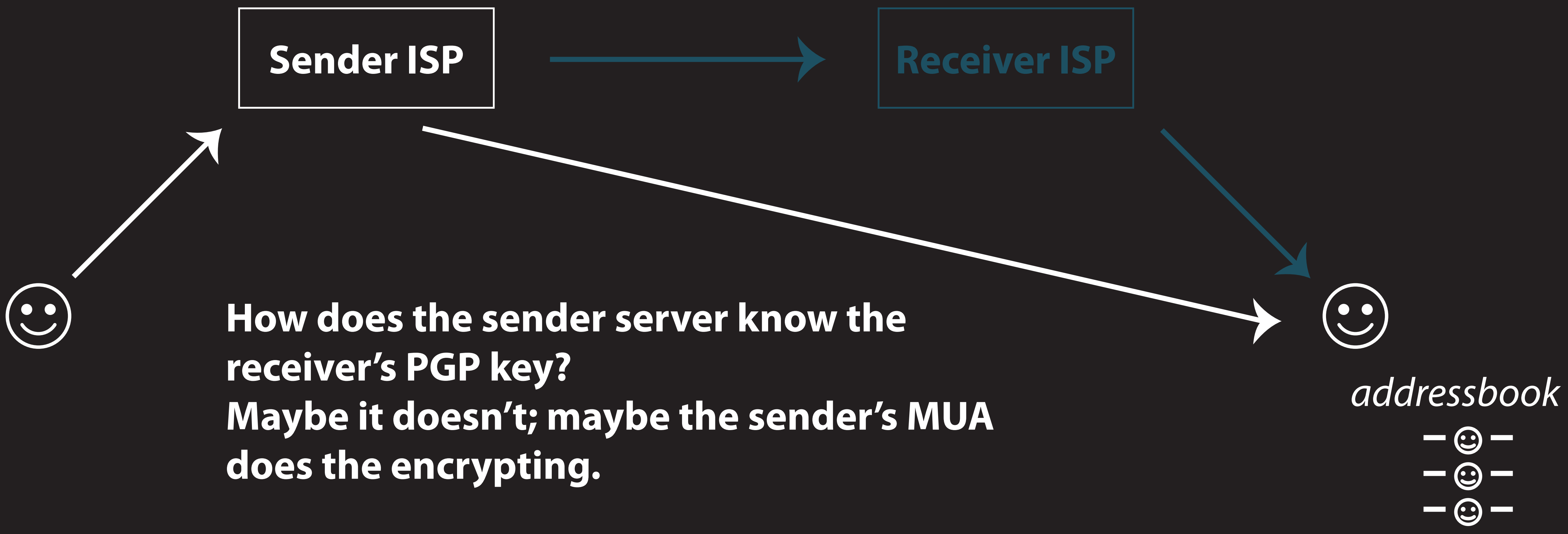


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- aren't you reinventing SSL?
- who decrypts the message?

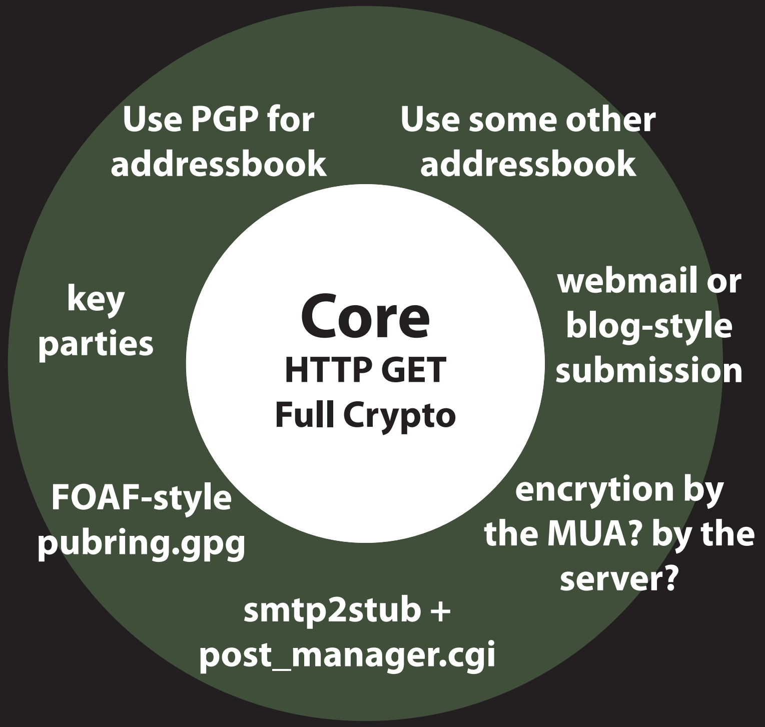
Simplest Possible Implementation



How does the sender server know the receiver's PGP key?
Maybe it doesn't; maybe the sender's MUA does the encrypting.

Our Problem

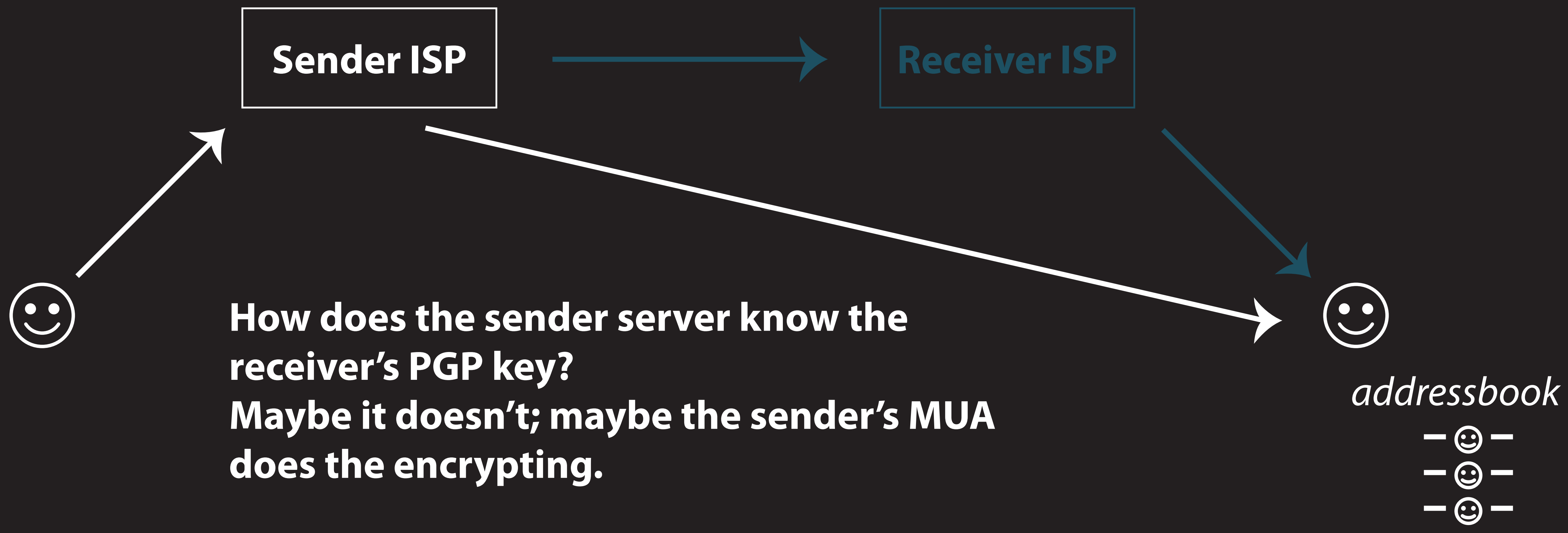
- given an outbox URL, retrieve messages
- PGP keyring contains the URLs
- PGP dorks hold keysigning parties at science fiction cons
- stealthier web of trust operating quietly in the background
- SMTP proxy + submission CGI
- maybe the sender MUA does the encrypting.


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

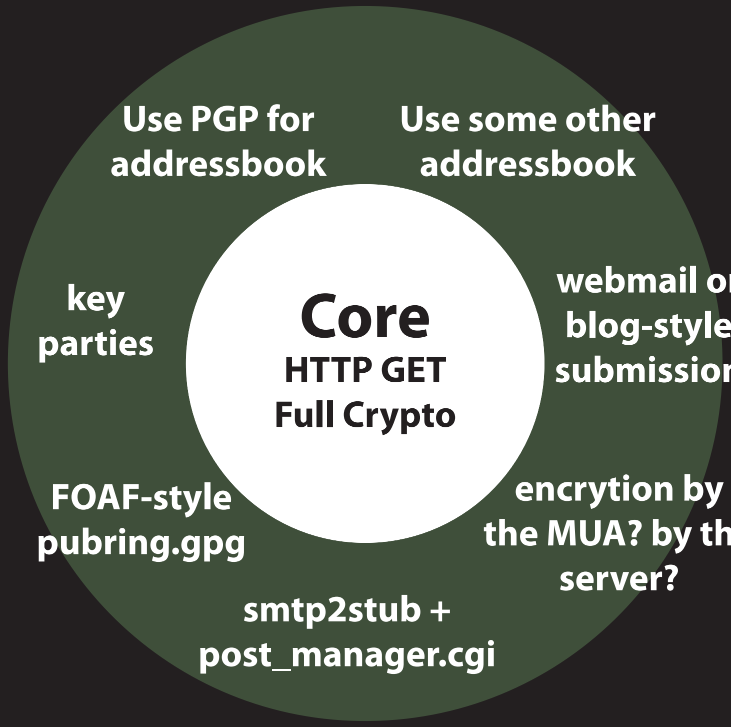
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



How does the sender server know the receiver's PGP key?
Maybe it doesn't; maybe the sender's MUA does the encrypting.

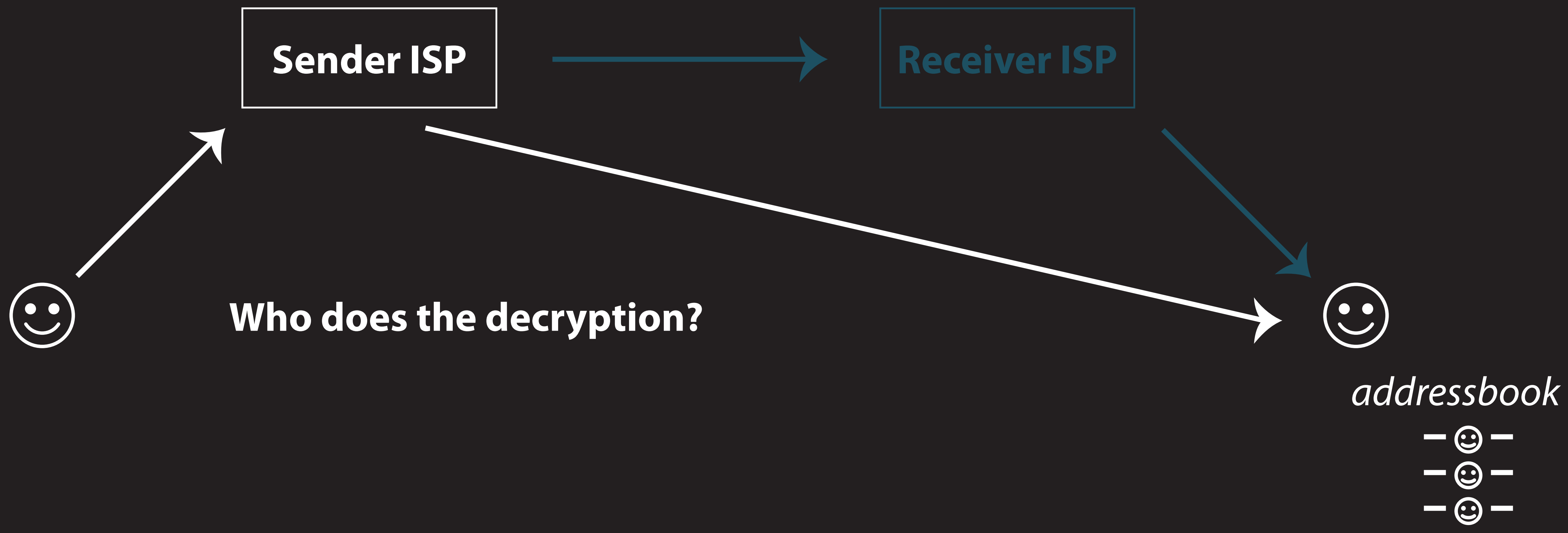
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?



Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

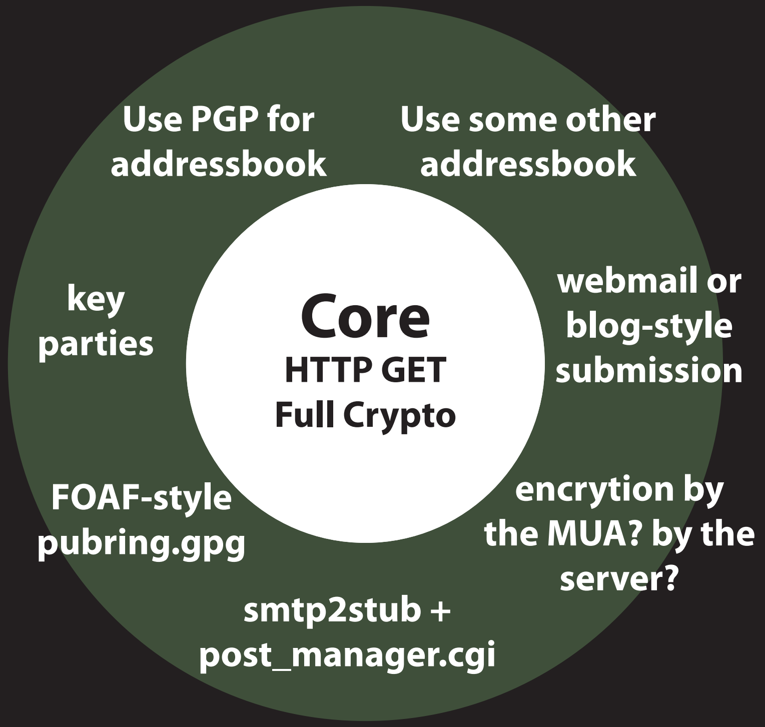
Out of Scope
- aren't you reinventing SSL?
- who decrypts the message?

Simplest Possible Implementation



Our Problem

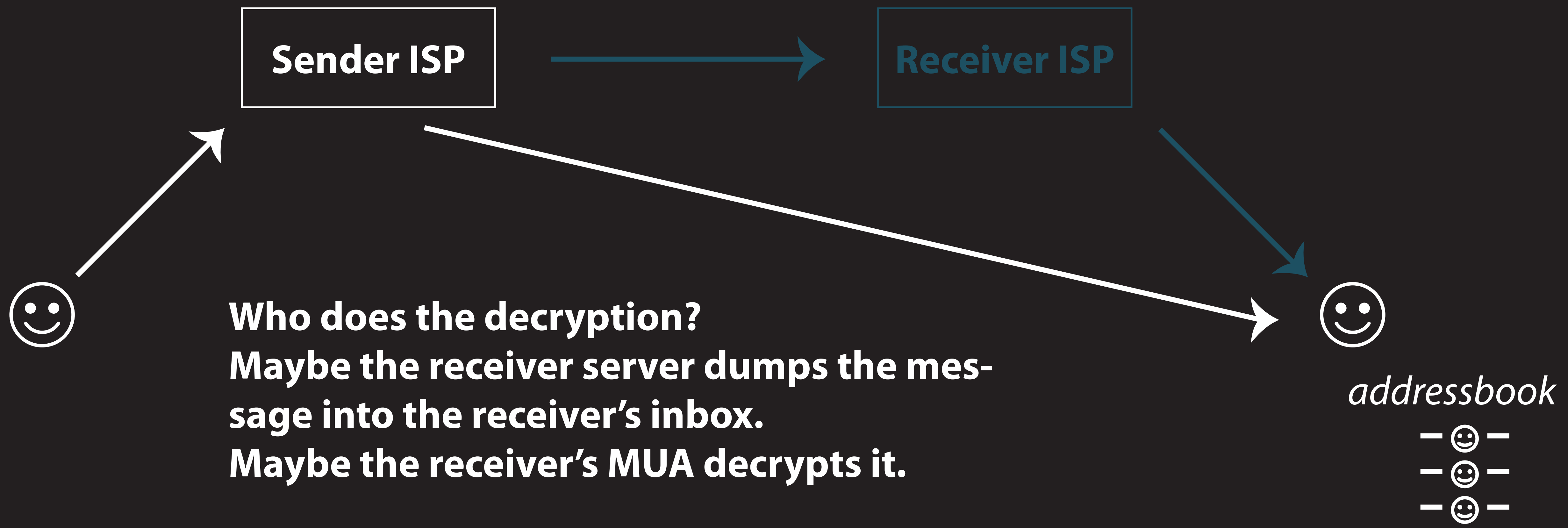
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- who decrypts the message?


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- aren't you reinventing SSL?

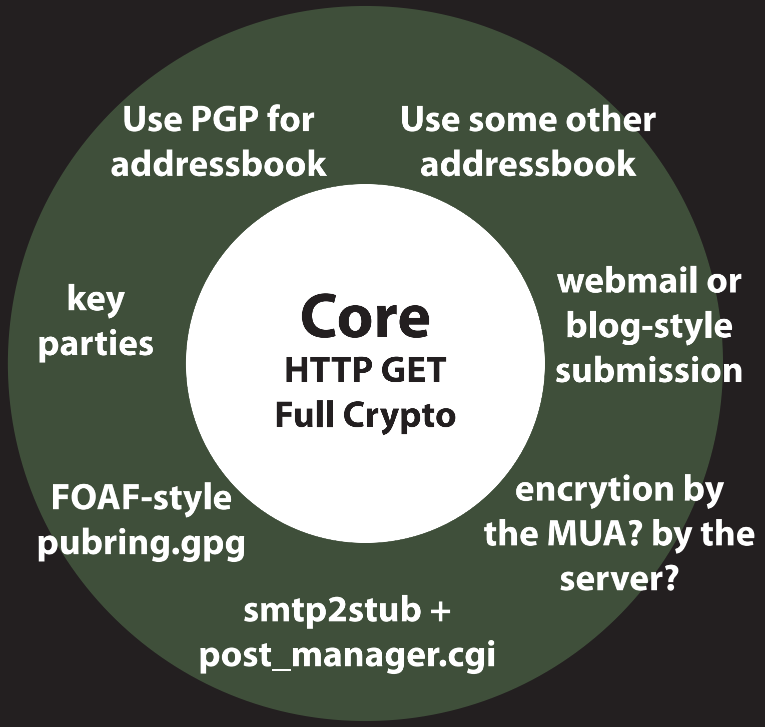
Simplest Possible Implementation



Who does the decryption?
Maybe the receiver server dumps the message into the receiver's inbox.
Maybe the receiver's MUA decrypts it.

Our Problem

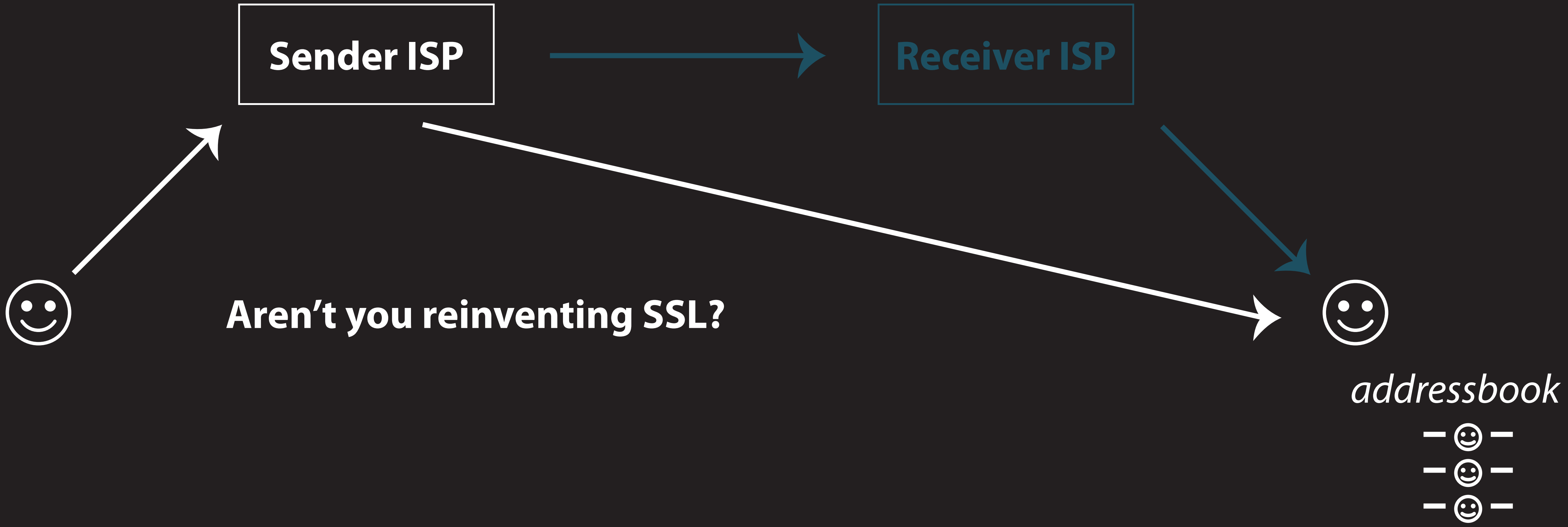
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- aren't you reinventing SSL?

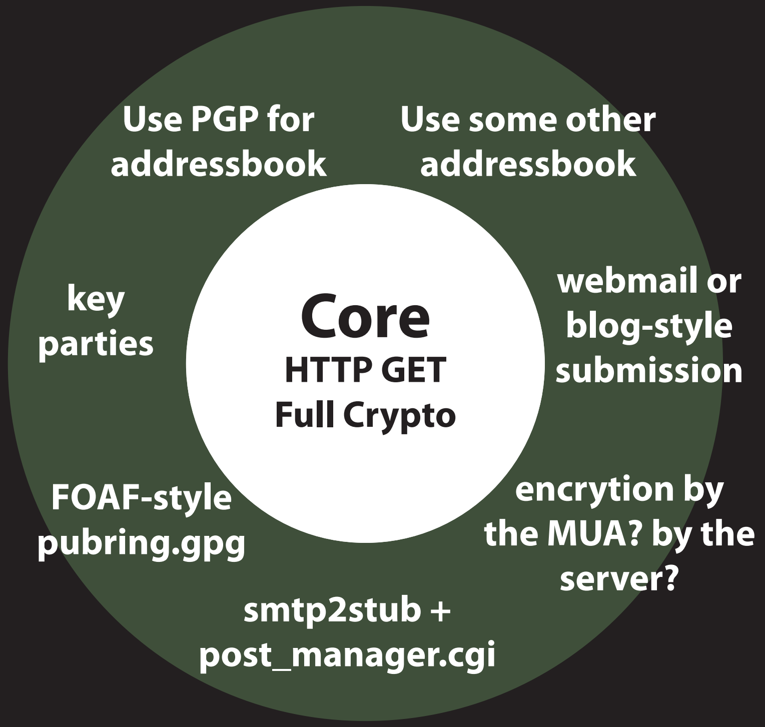
Simplest Possible Implementation



Aren't you reinventing SSL?

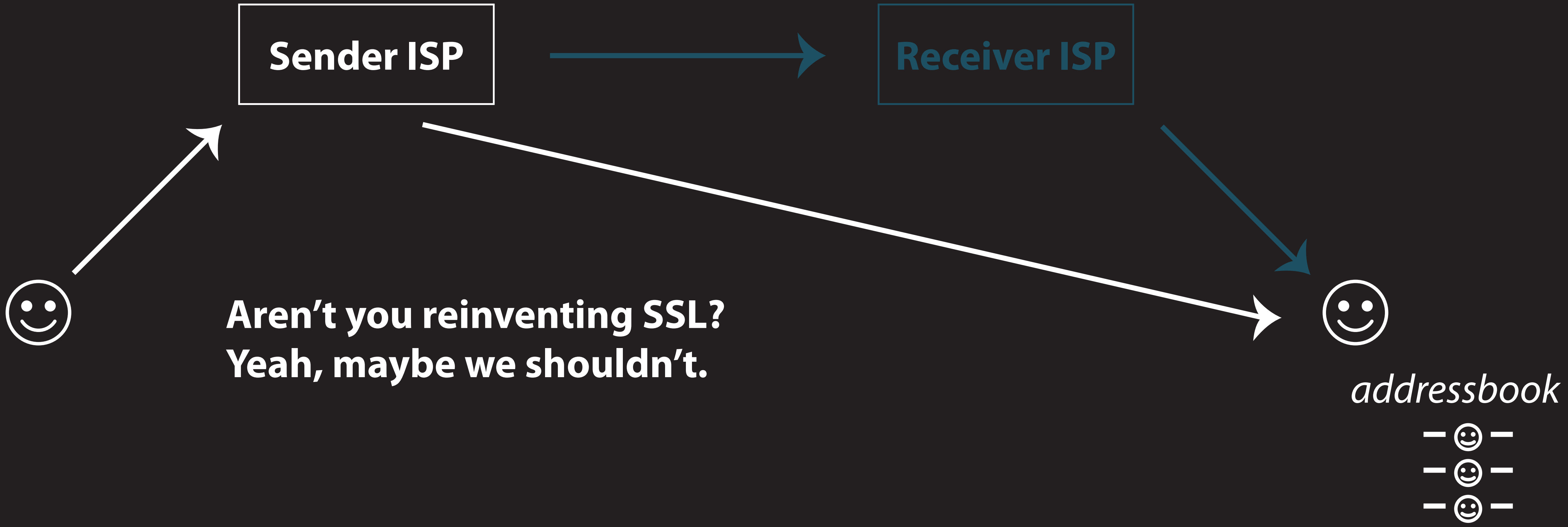
Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- aren't you reinventing SSL?


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

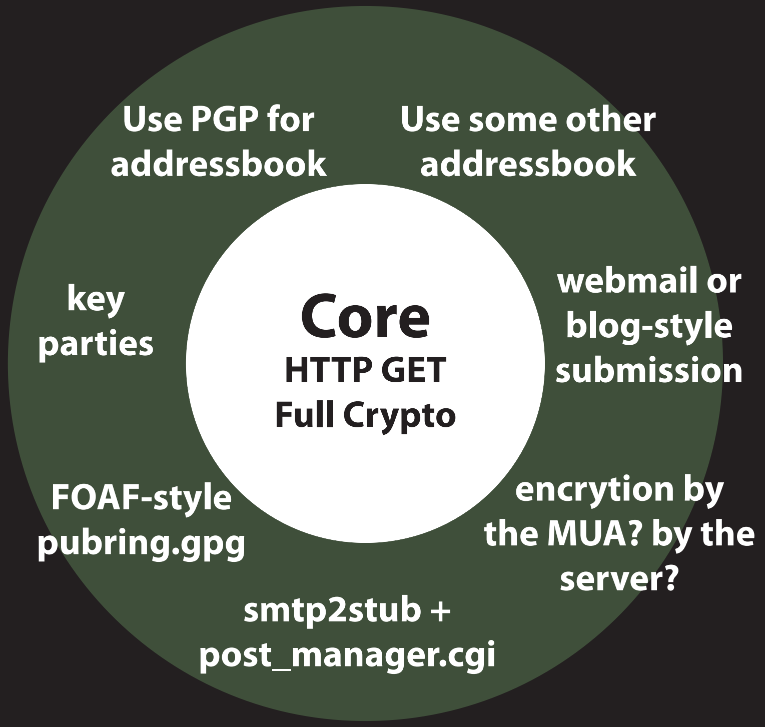
Simplest Possible Implementation



Aren't you reinventing SSL?
Yeah, maybe we shouldn't.

Our Problem

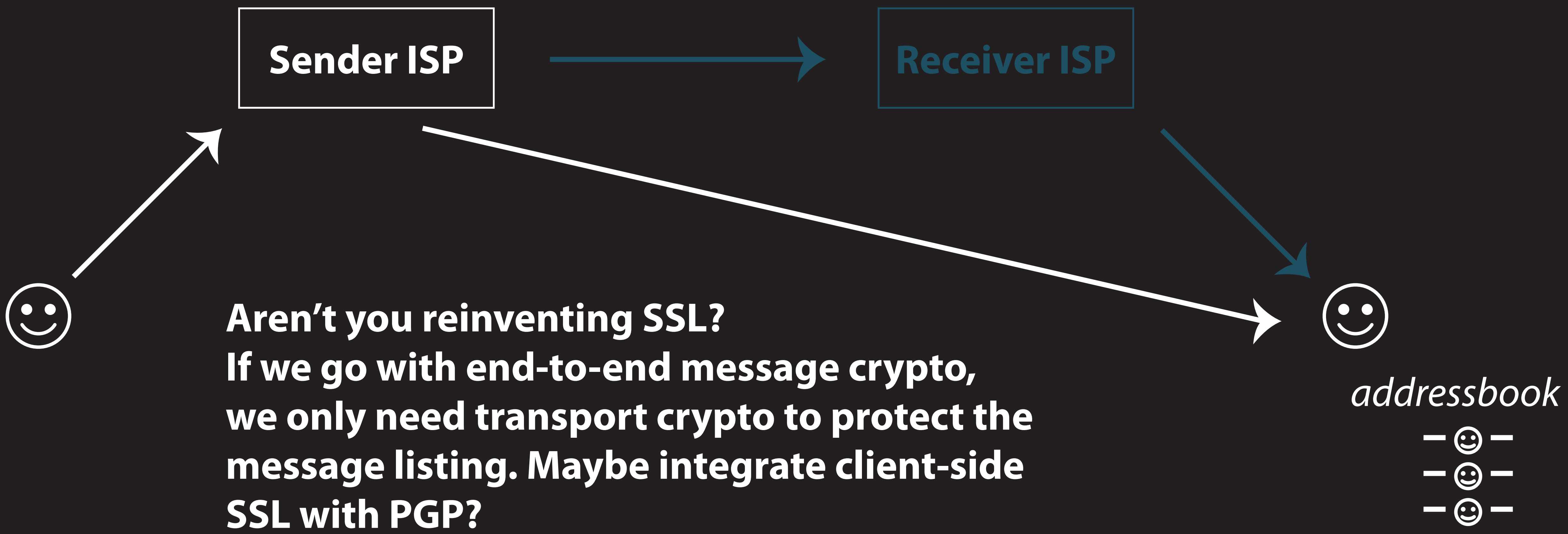
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- aren't you reinventing SSL?



Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

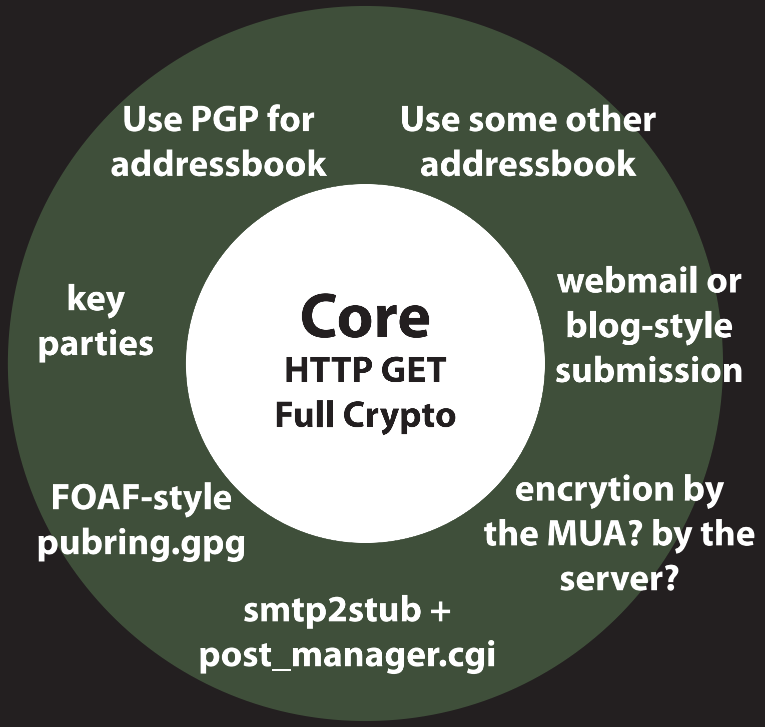
Simplest Possible Implementation



Aren't you reinventing SSL?
If we go with end-to-end message crypto, we only need transport crypto to protect the message listing. Maybe integrate client-side SSL with PGP?

Our Problem

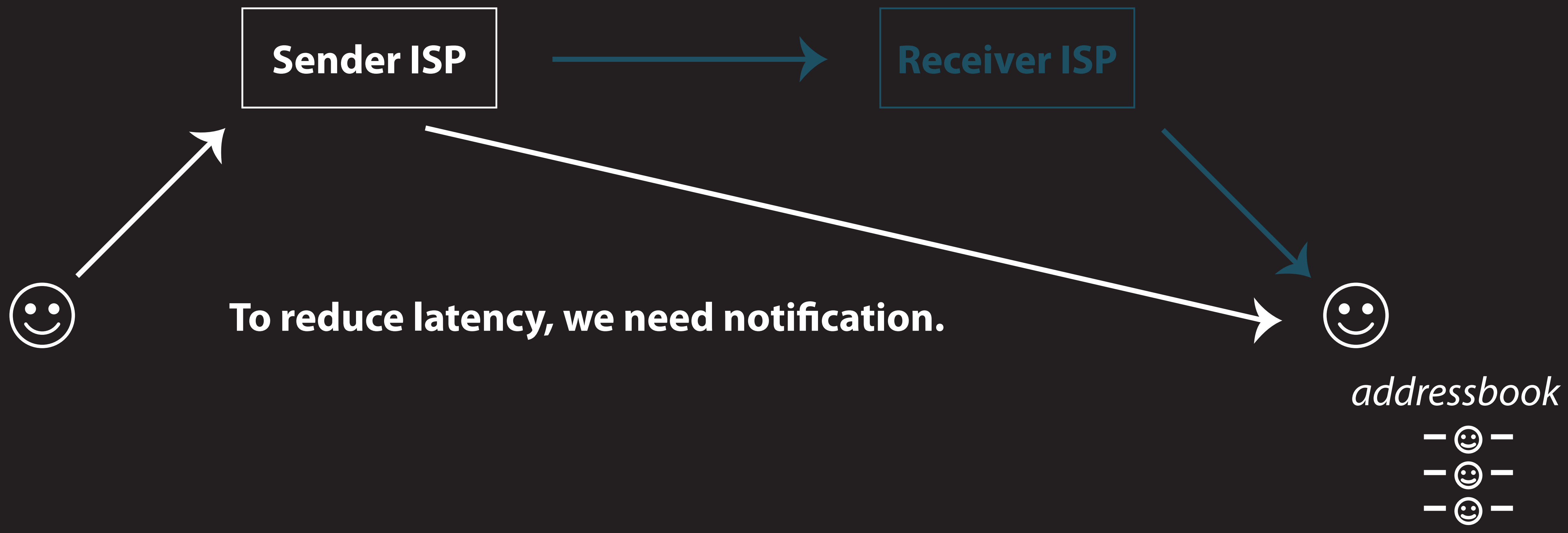
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?



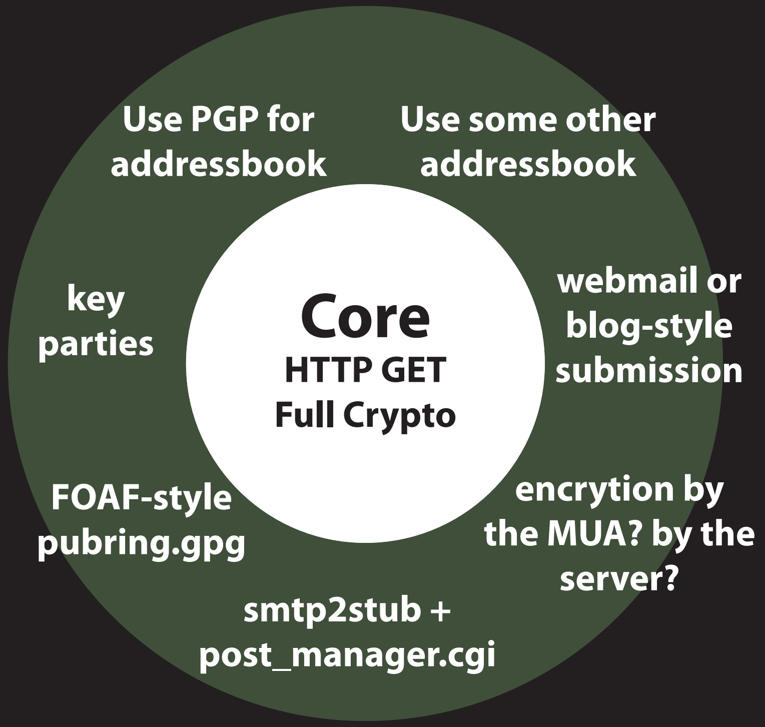
Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

Simplest Possible Implementation



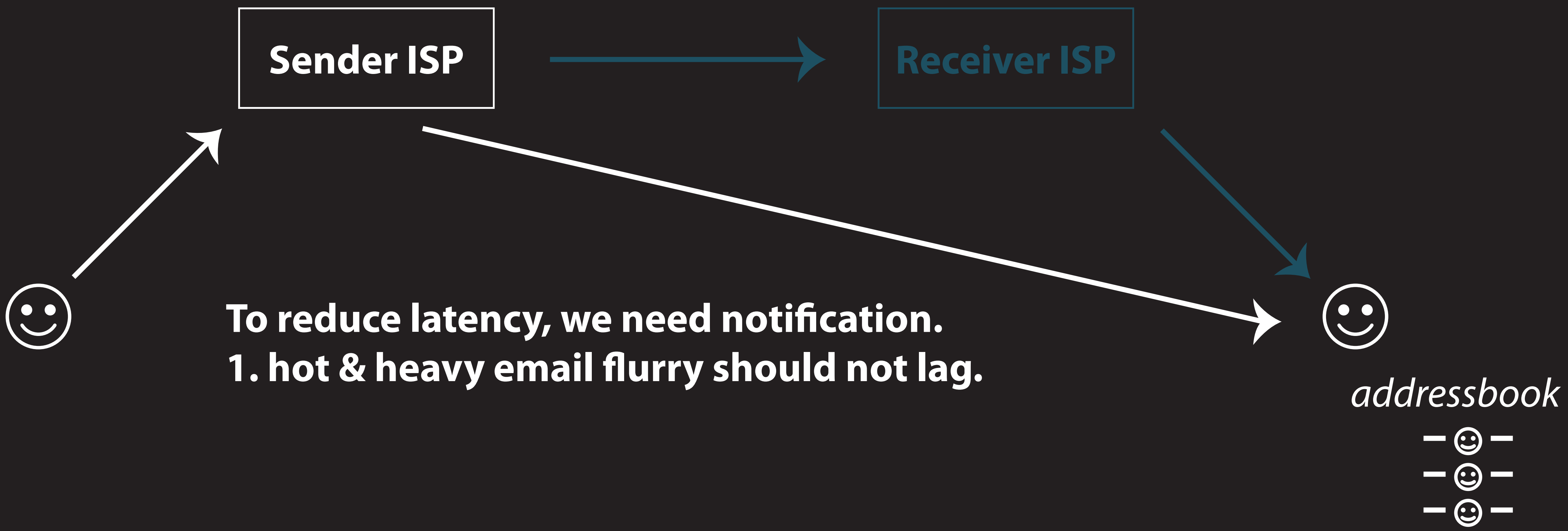
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope
- what about notification?
- what about polling?

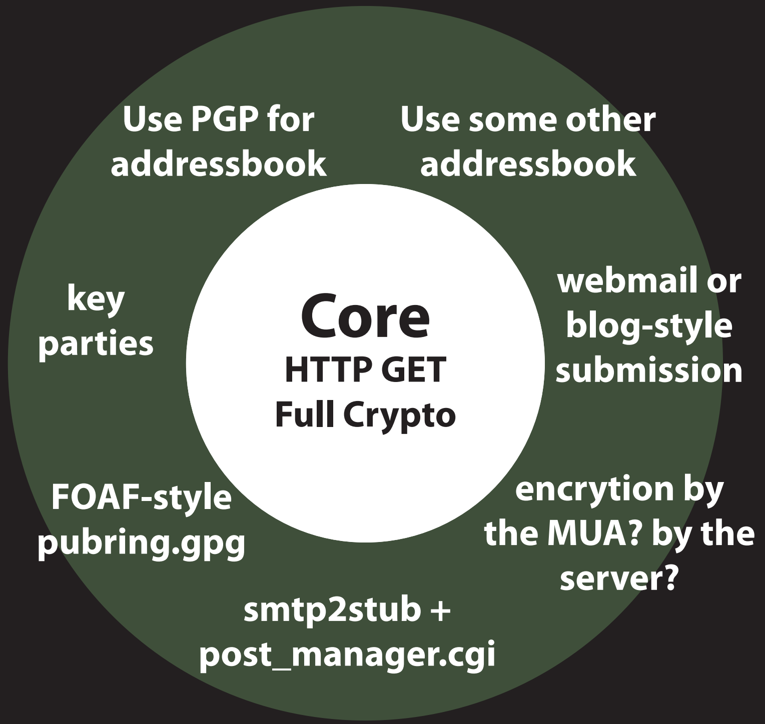
can we make them more lightweight?

Simplest Possible Implementation



To reduce latency, we need notification.
1. hot & heavy email flurry should not lag.

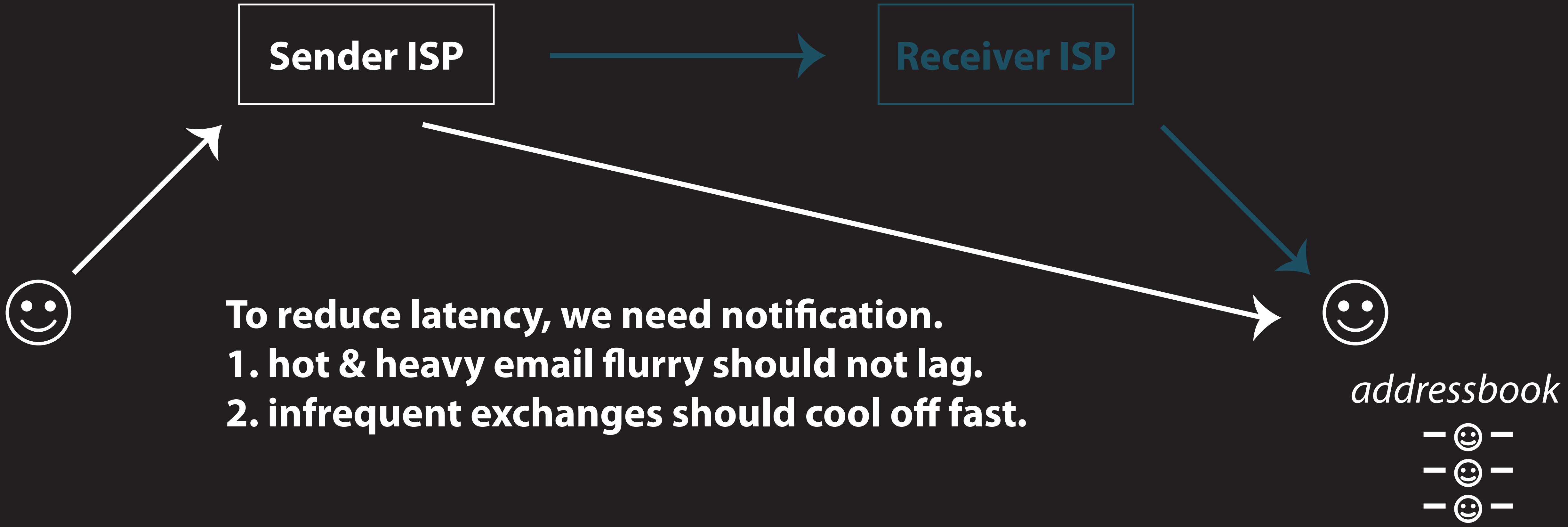
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?


Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope
- what about notification?
- what about polling?

can we make them more lightweight?

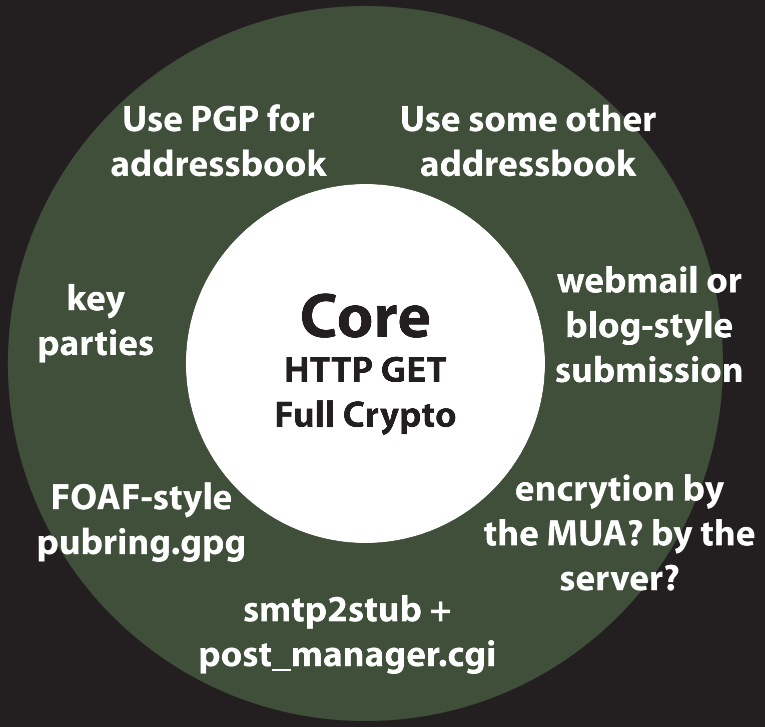
Simplest Possible Implementation



To reduce latency, we need notification.
1. hot & heavy email flurry should not lag.
2. infrequent exchanges should cool off fast.

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?

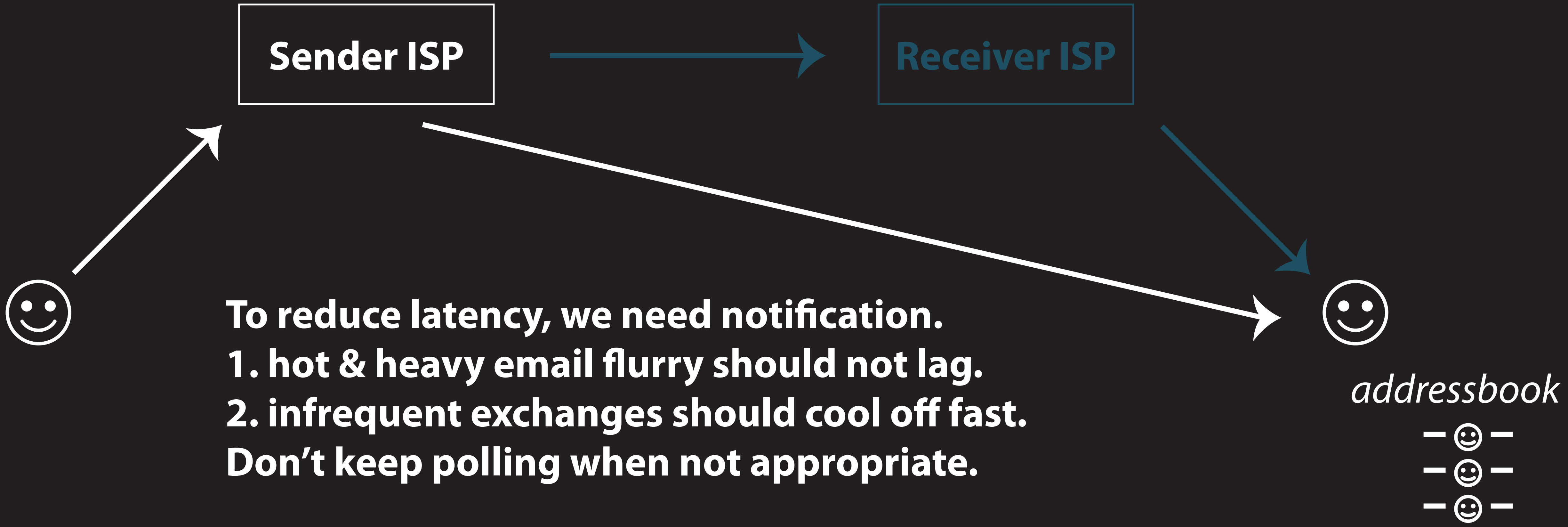

Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- what about notification?
- what about polling?

can we make them more lightweight?

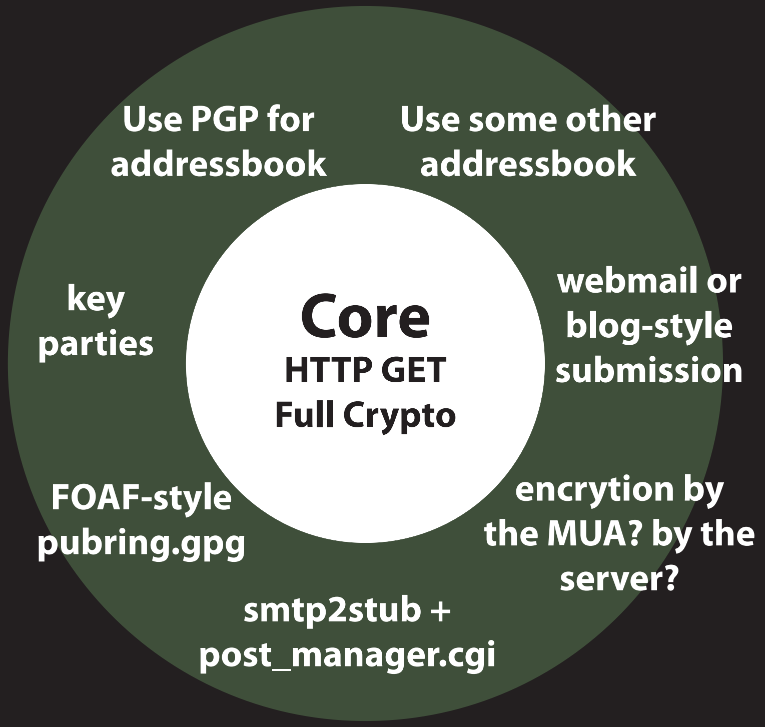
Simplest Possible Implementation



To reduce latency, we need notification.
1. hot & heavy email flurry should not lag.
2. infrequent exchanges should cool off fast.
Don't keep polling when not appropriate.

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?

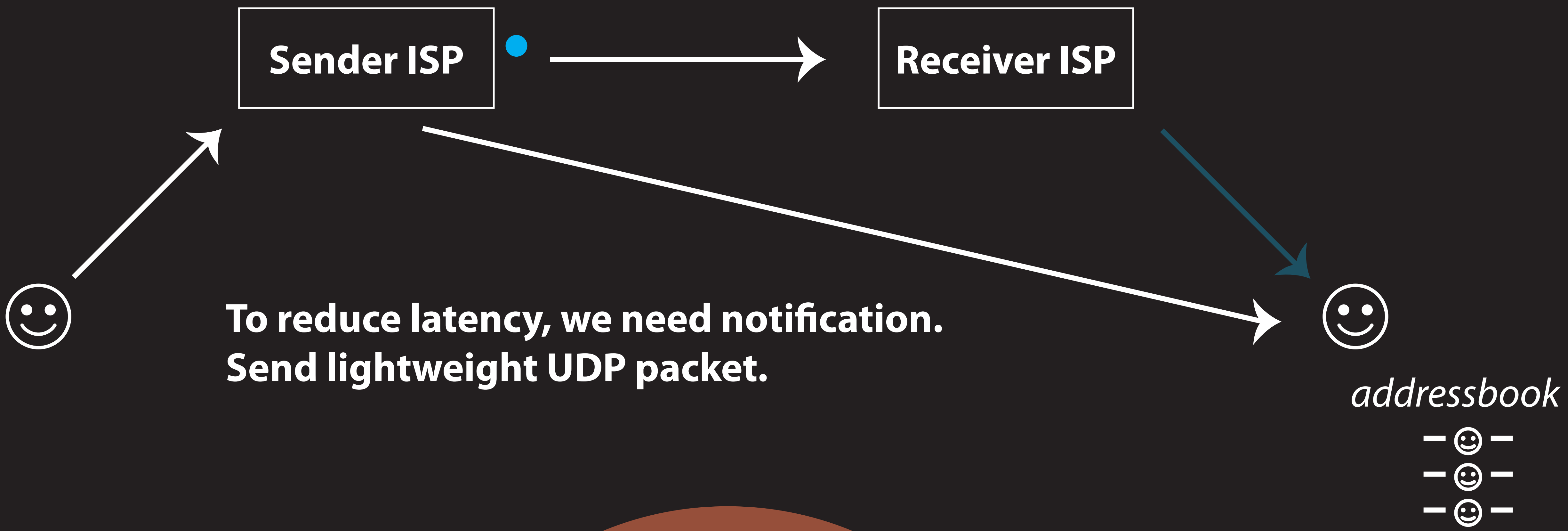

Core
HTTP GET
Full Crypto
Use PGP for addressbook
Use some other addressbook
webmail or blog-style submission
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi
FOAF-style pubring.gpg
key parties

Out of Scope

- what about notification?
- what about polling?

can we make them more lightweight?

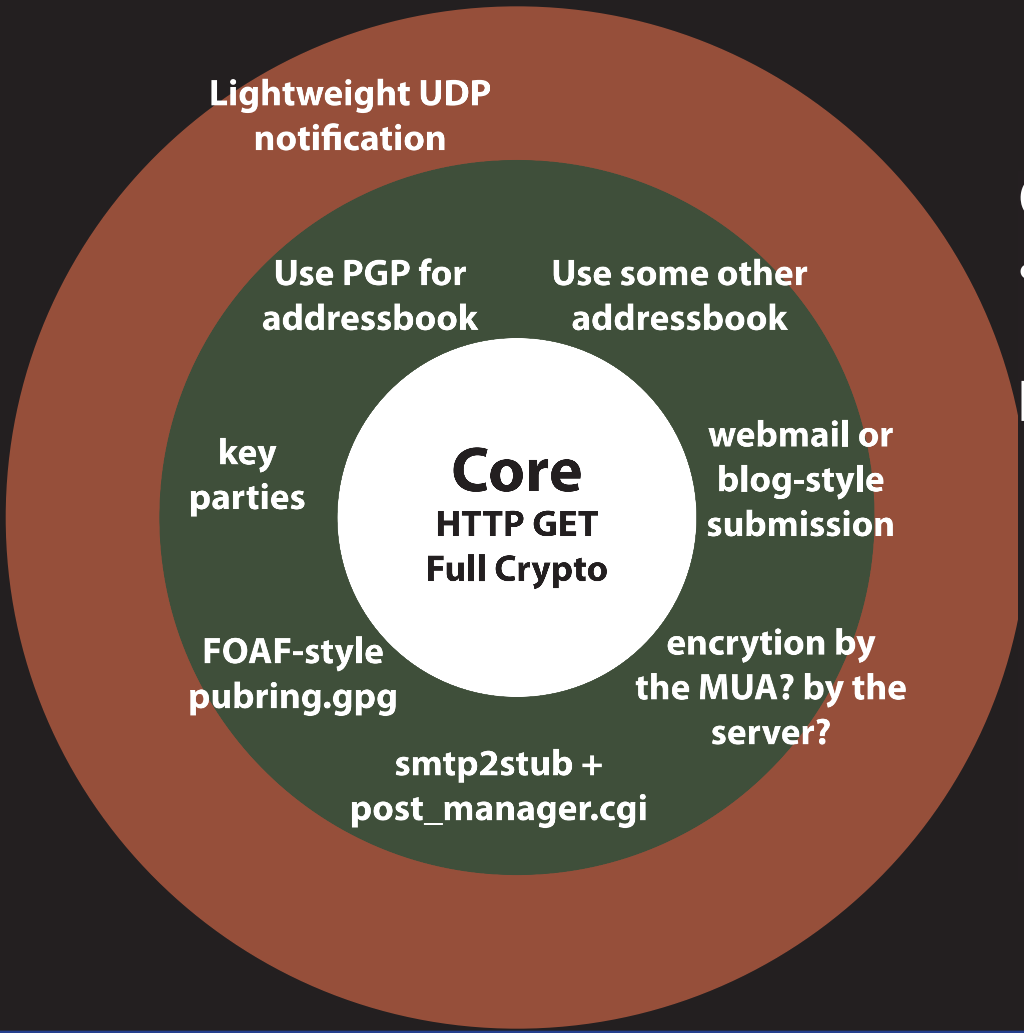
Simplest Possible Implementation



To reduce latency, we need notification.
Send lightweight UDP packet.

Our Problem

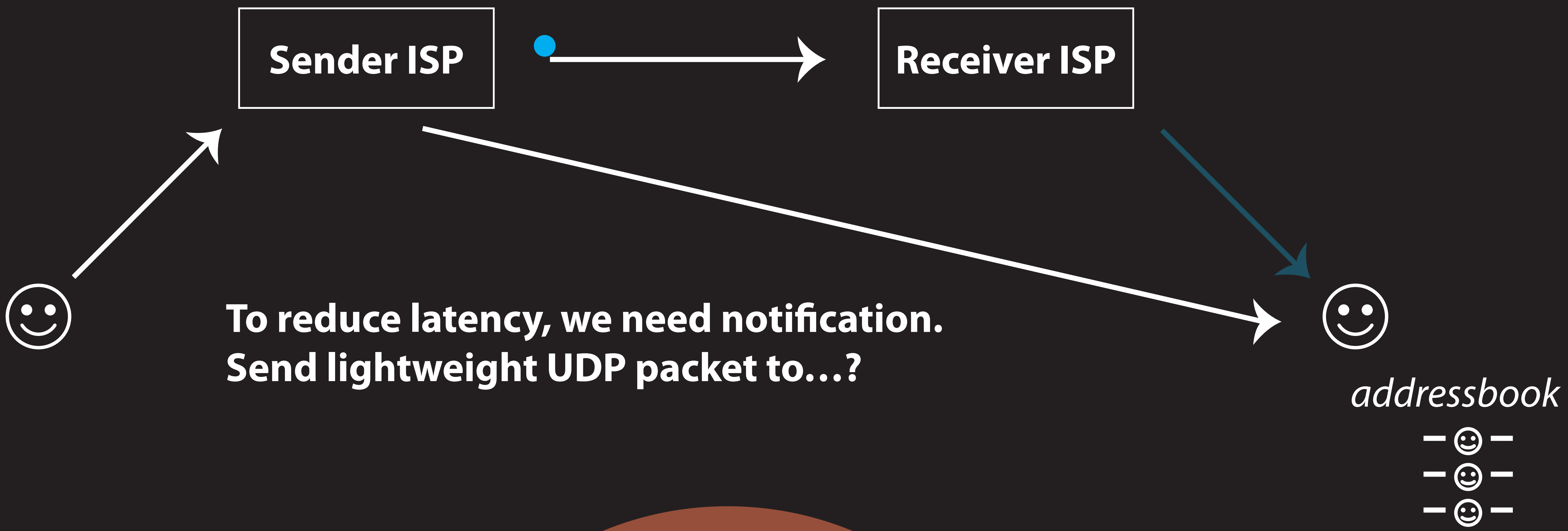
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?
- push notification with UDP.


Lightweight UDP notification
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

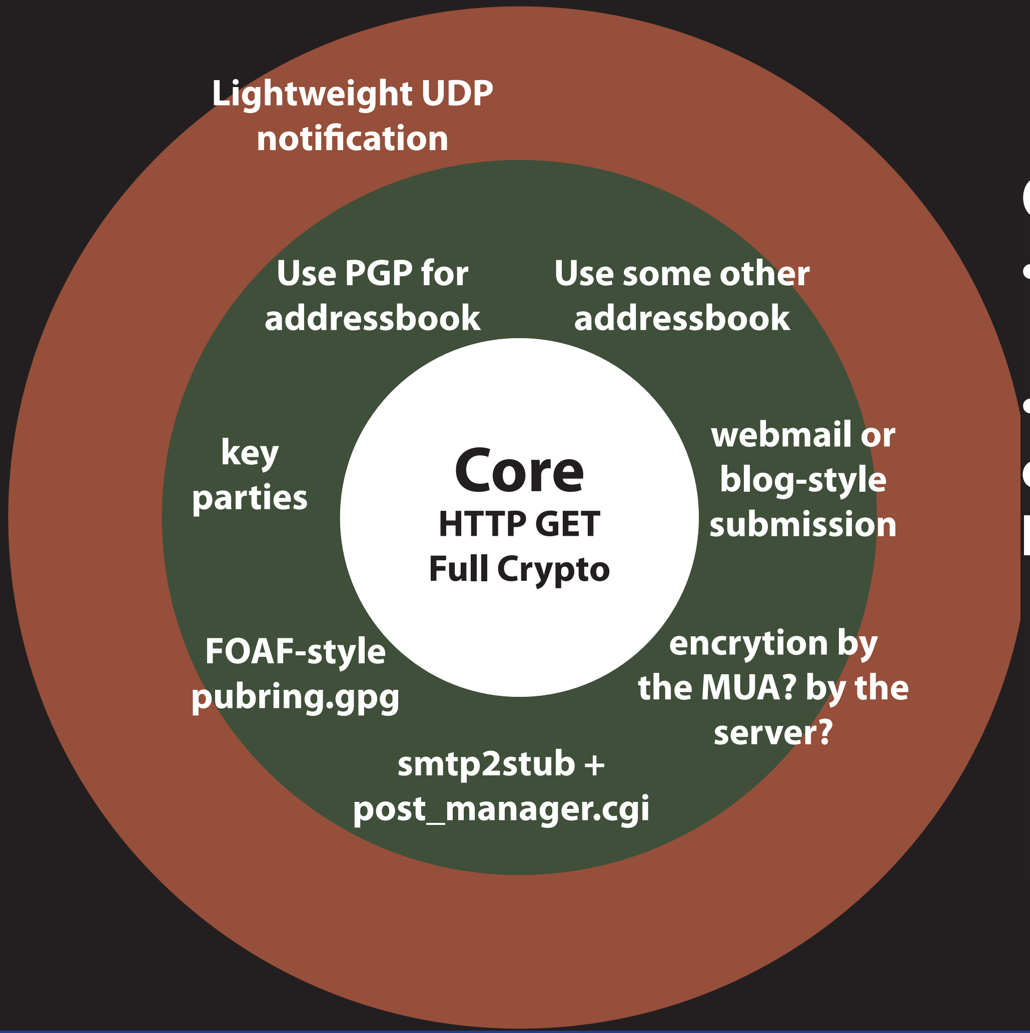
- what about polling?
 can we make them more
lightweight?

Simplest Possible Implementation



To reduce latency, we need notification.
Send lightweight UDP packet to...?

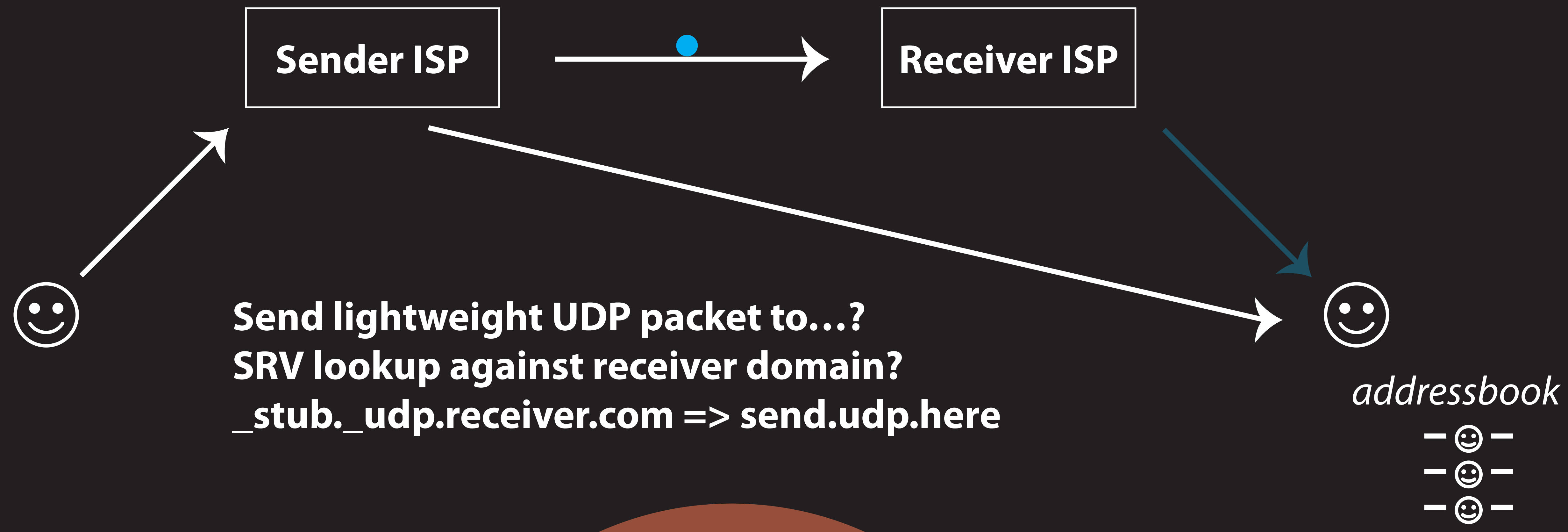
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?
- push notification with UDP.


Lightweight UDP notification
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope
- how do senders find out where UDP notifications go?
- what about polling?

can we make them more lightweight?

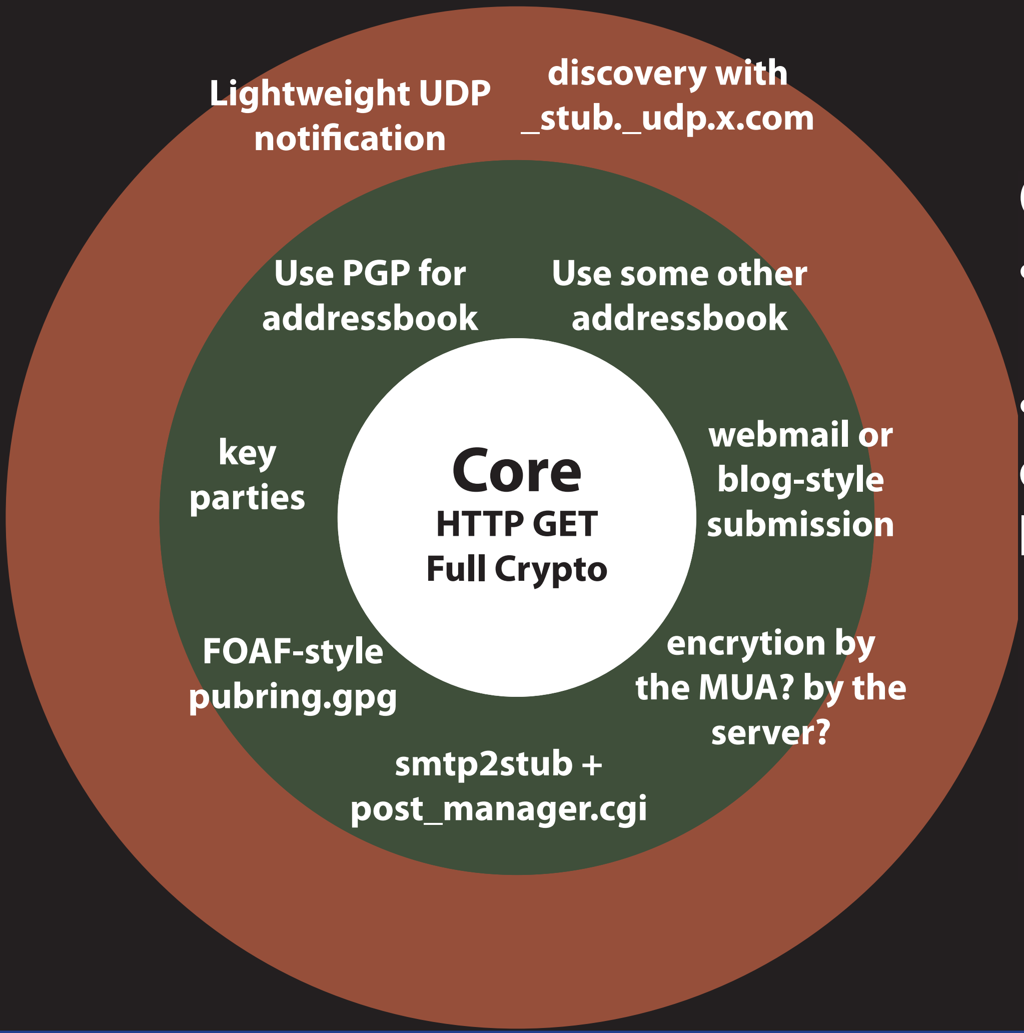
Simplest Possible Implementation



Send lightweight UDP packet to…?
SRV lookup against receiver domain?
_stub._udp.receiver.com => send.udp.here

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?
- push notification with UDP.
- UDP configured with DNS?



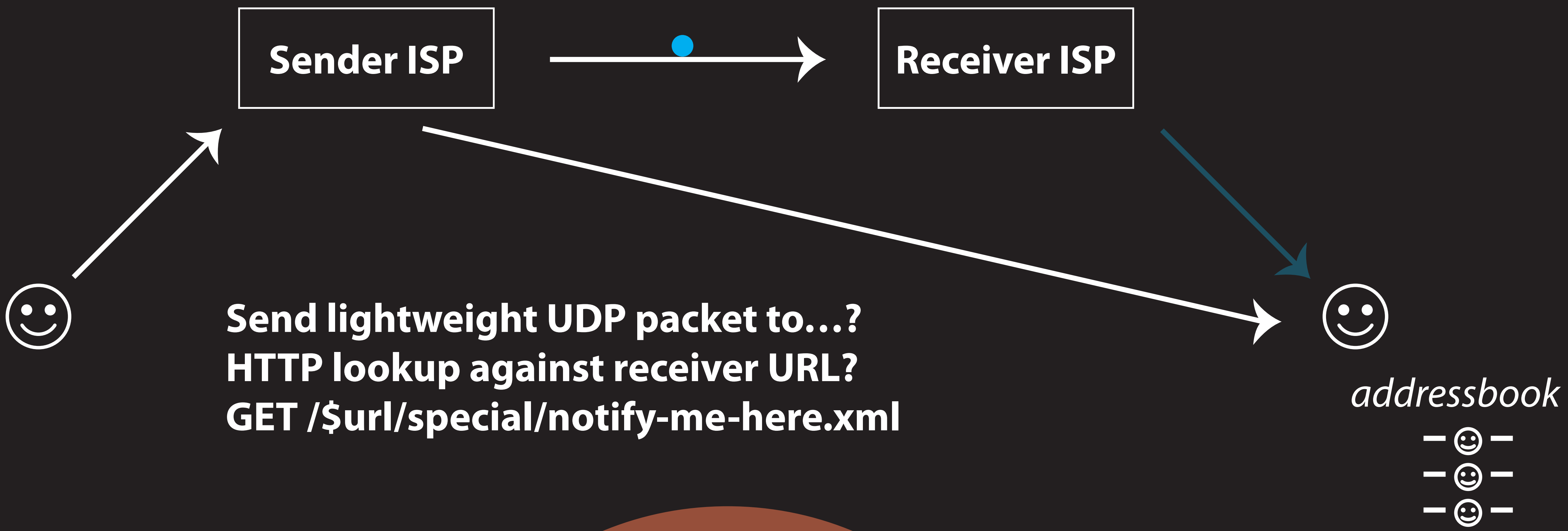
Lightweight UDP notification
discovery with _stub._udp.x.com
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

- how do senders find out where UDP notifications go?
- what about polling?

can we make them more lightweight?

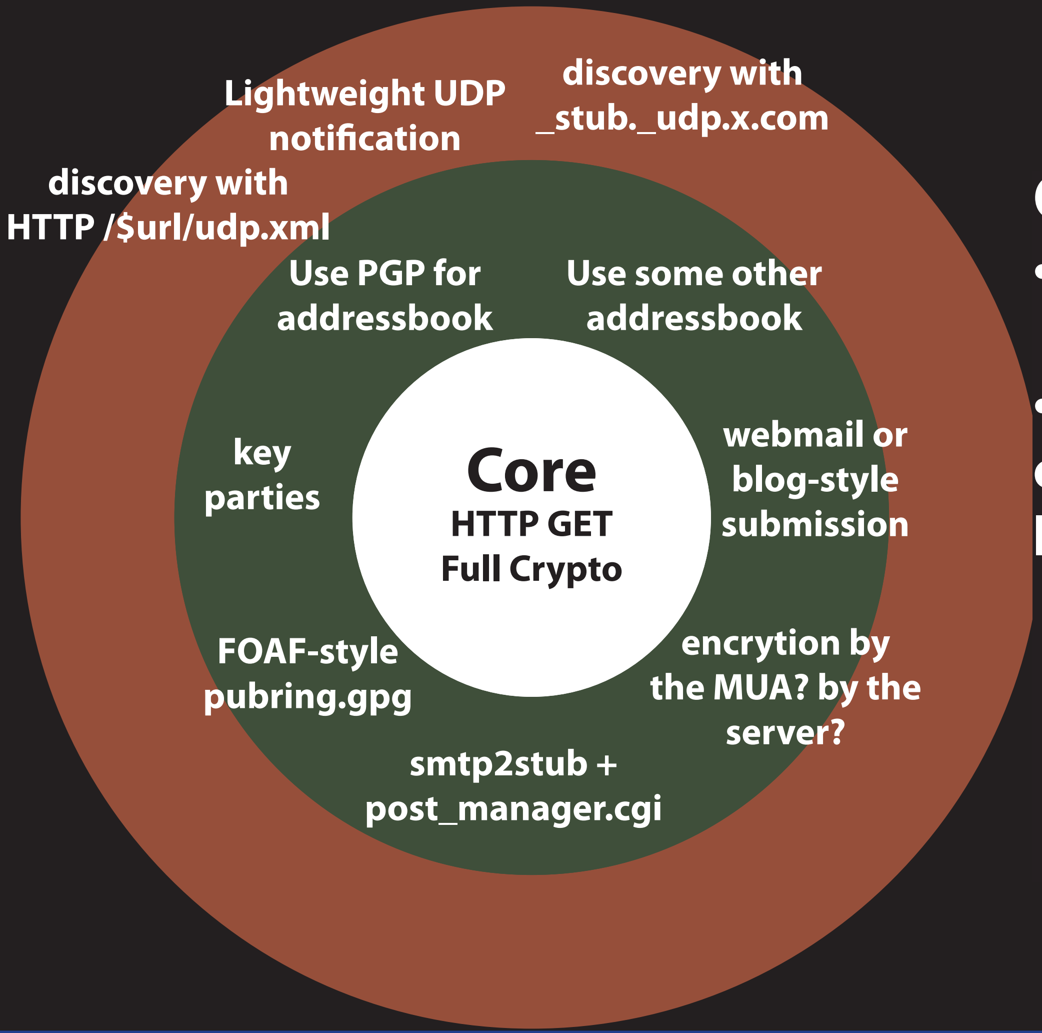
Simplest Possible Implementation



Send lightweight UDP packet to…?
HTTP lookup against receiver URL?
GET /$url/special/notify-me-here.xml

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?
- push notification with UDP.
- UDP configured with HTTP?

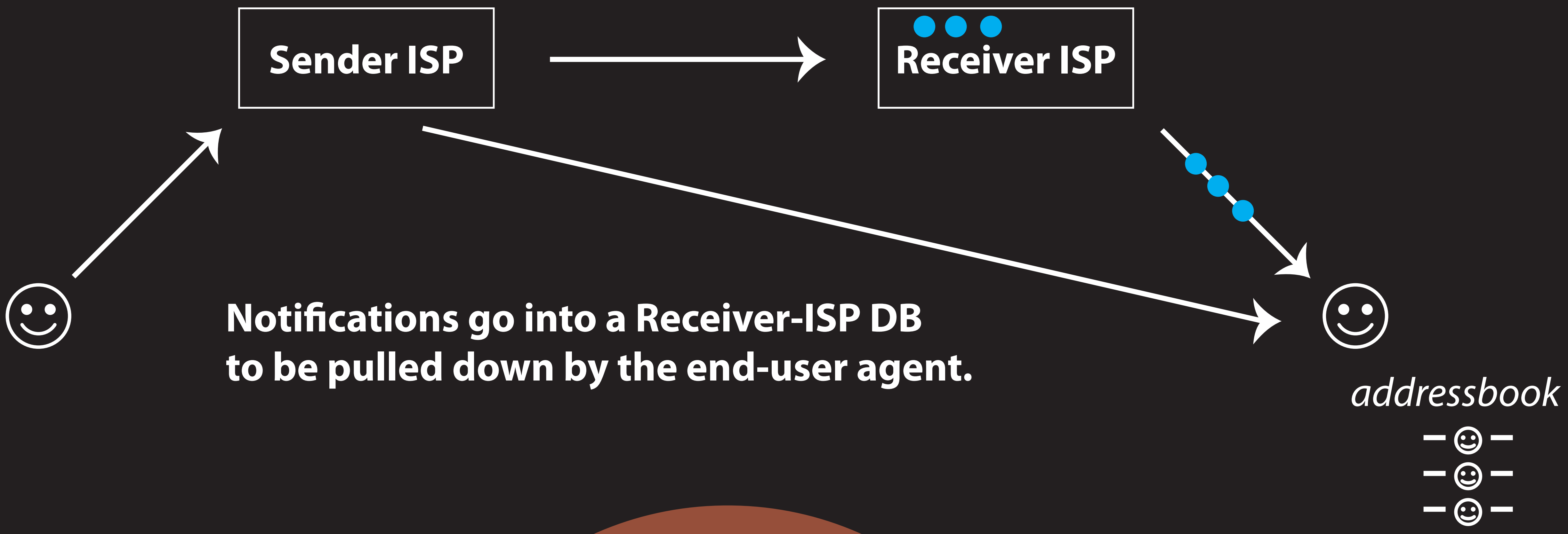

Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

- how do senders find out where UDP notifications go?
- what about polling?

can we make them more lightweight?

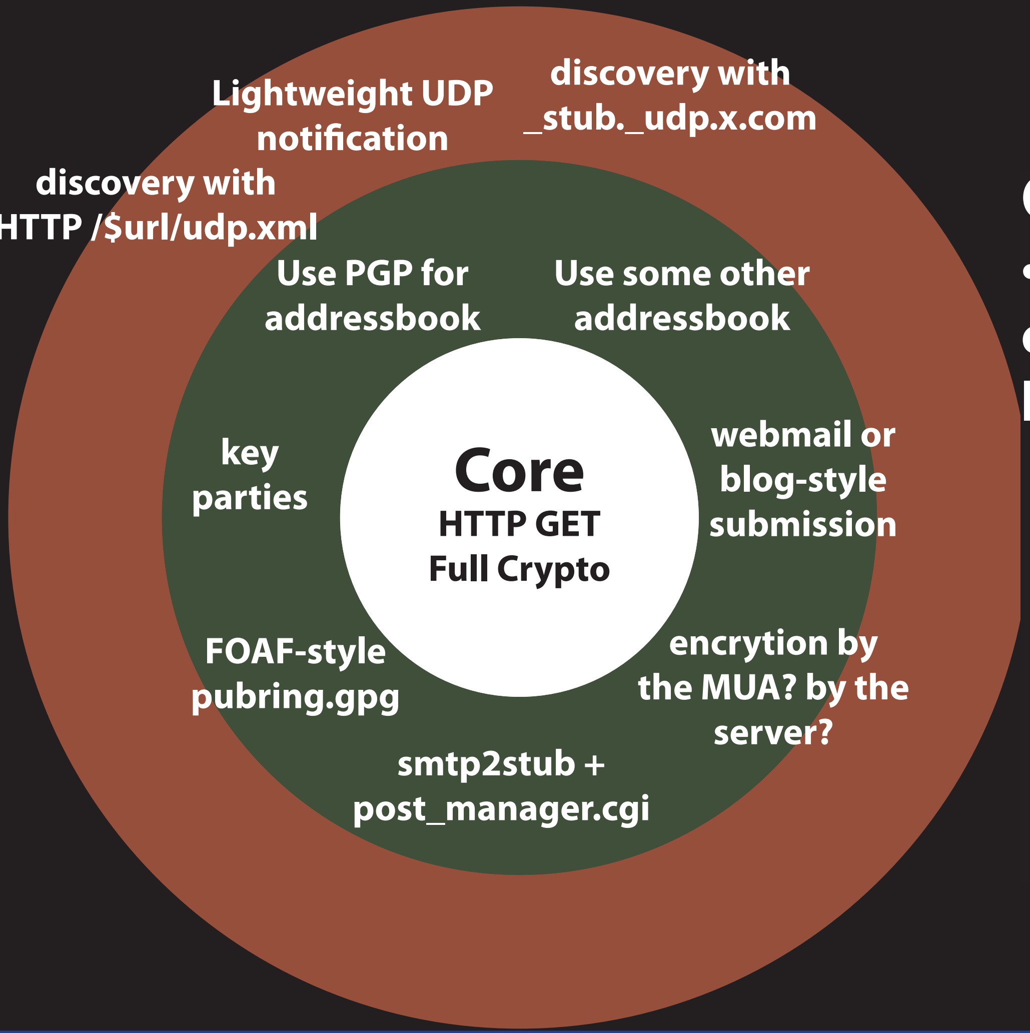
Simplest Possible Implementation



Notifications go into a Receiver-ISP DB to be pulled down by the end-user agent.

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- must we protect transport?
- push notification with UDP.
- UDP configured with HTTP?
- Receiver ISP stores “dirty list”.

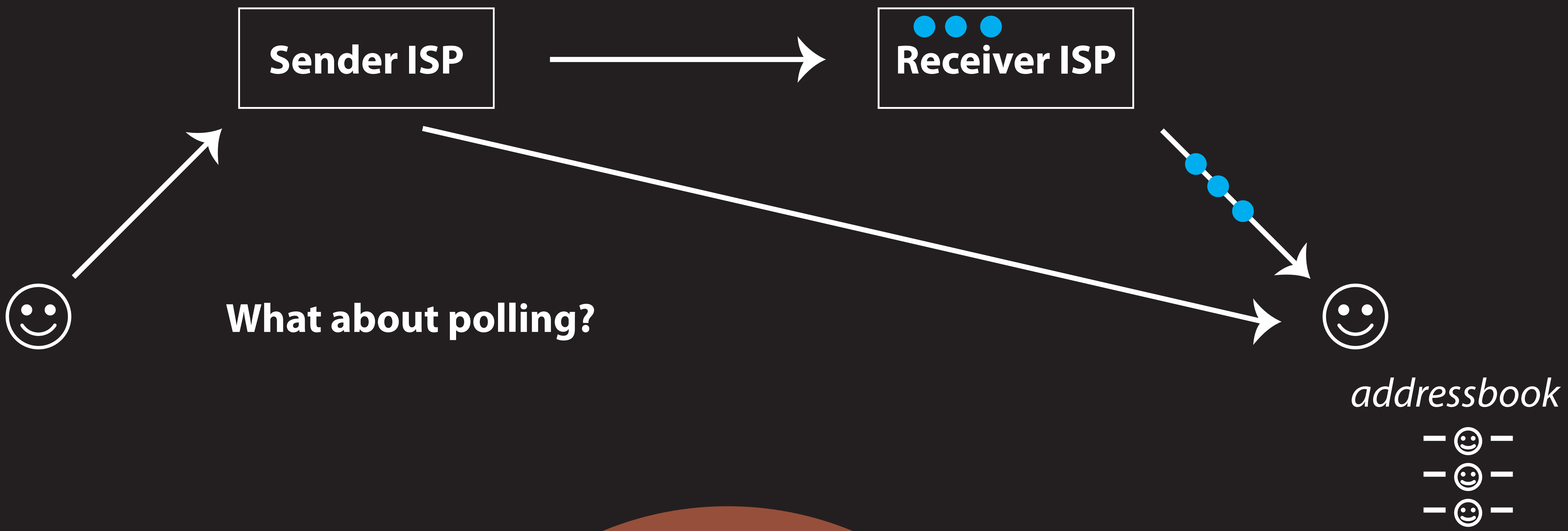

Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

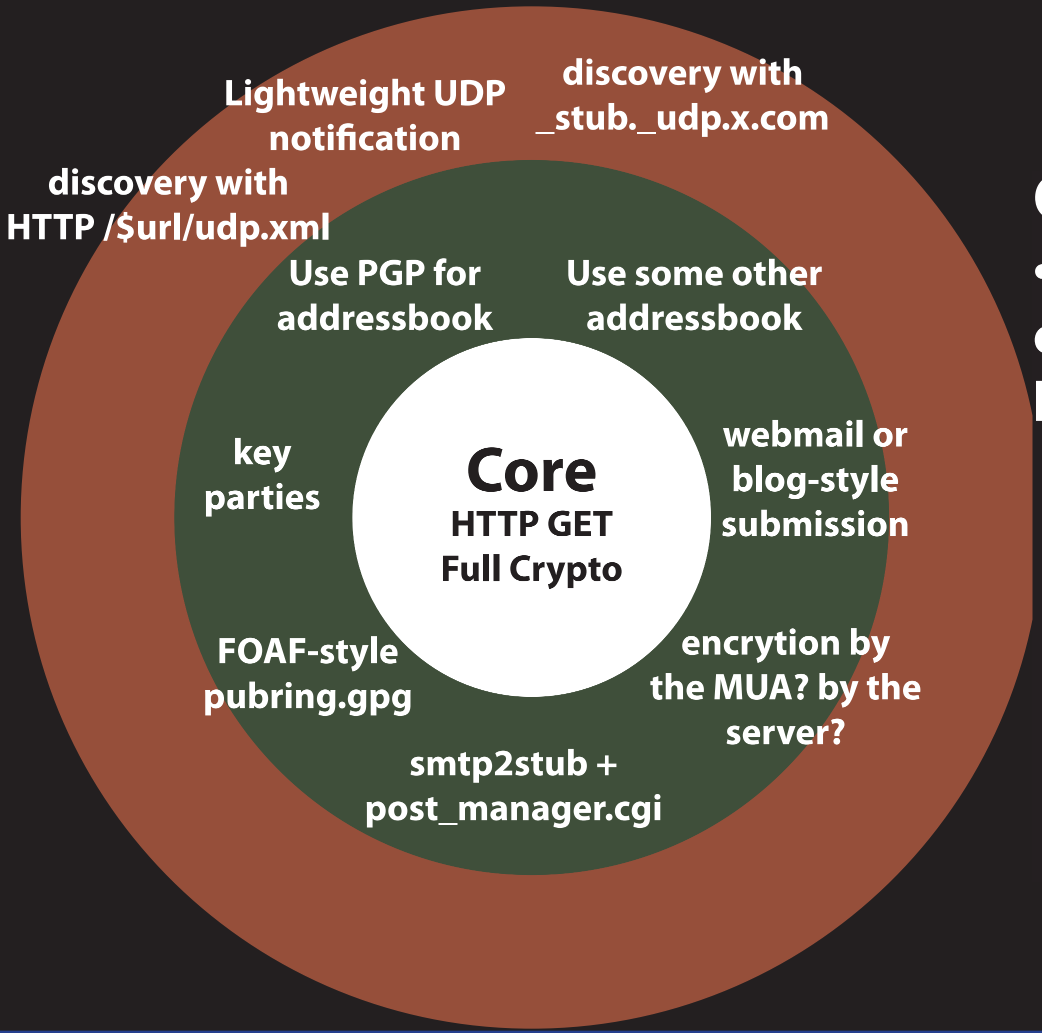
- what about polling?

can we make them more lightweight?

Simplest Possible Implementation



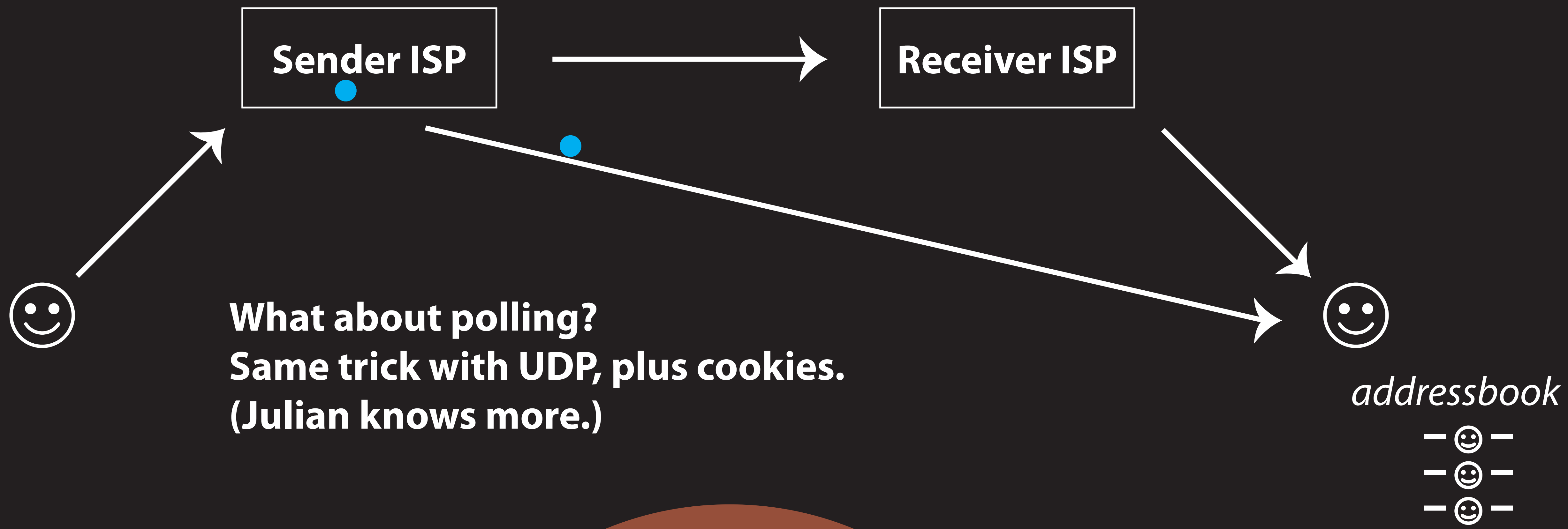
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- push notification with UDP.
- Receiver ISP stores "dirty list".


Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope
- what about polling?

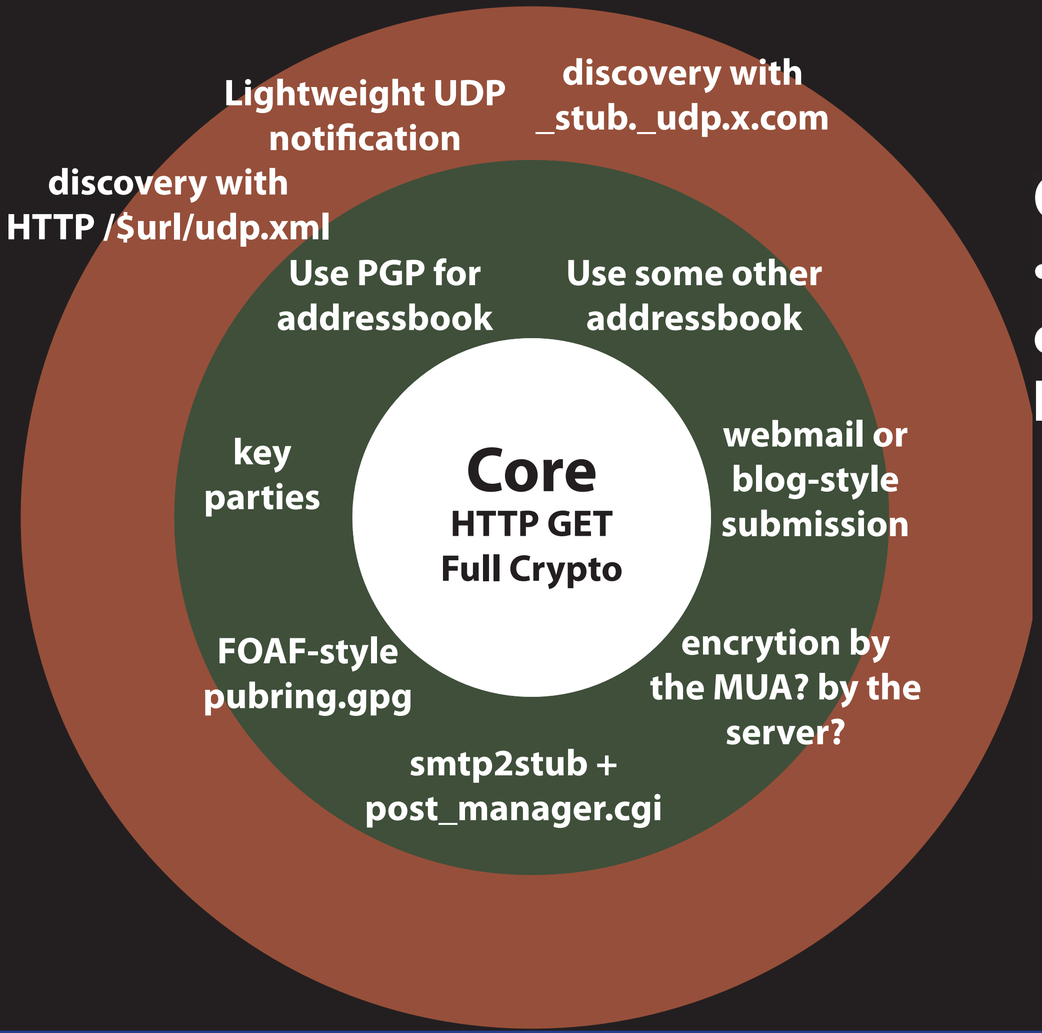
can we make them more lightweight?

Simplest Possible Implementation



What about polling?
Same trick with UDP, plus cookies.
(Julian knows more.)

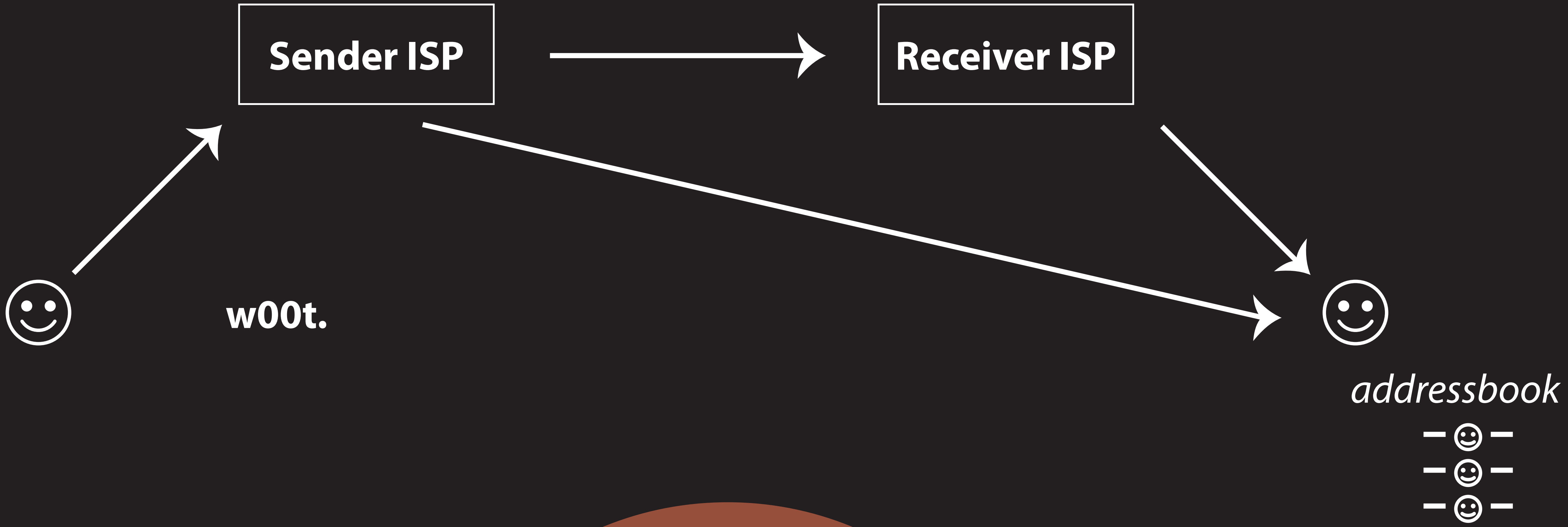
Our Problem
- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- push notification with UDP.
- Receiver ISP stores "dirty list".


Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope
- what about polling?

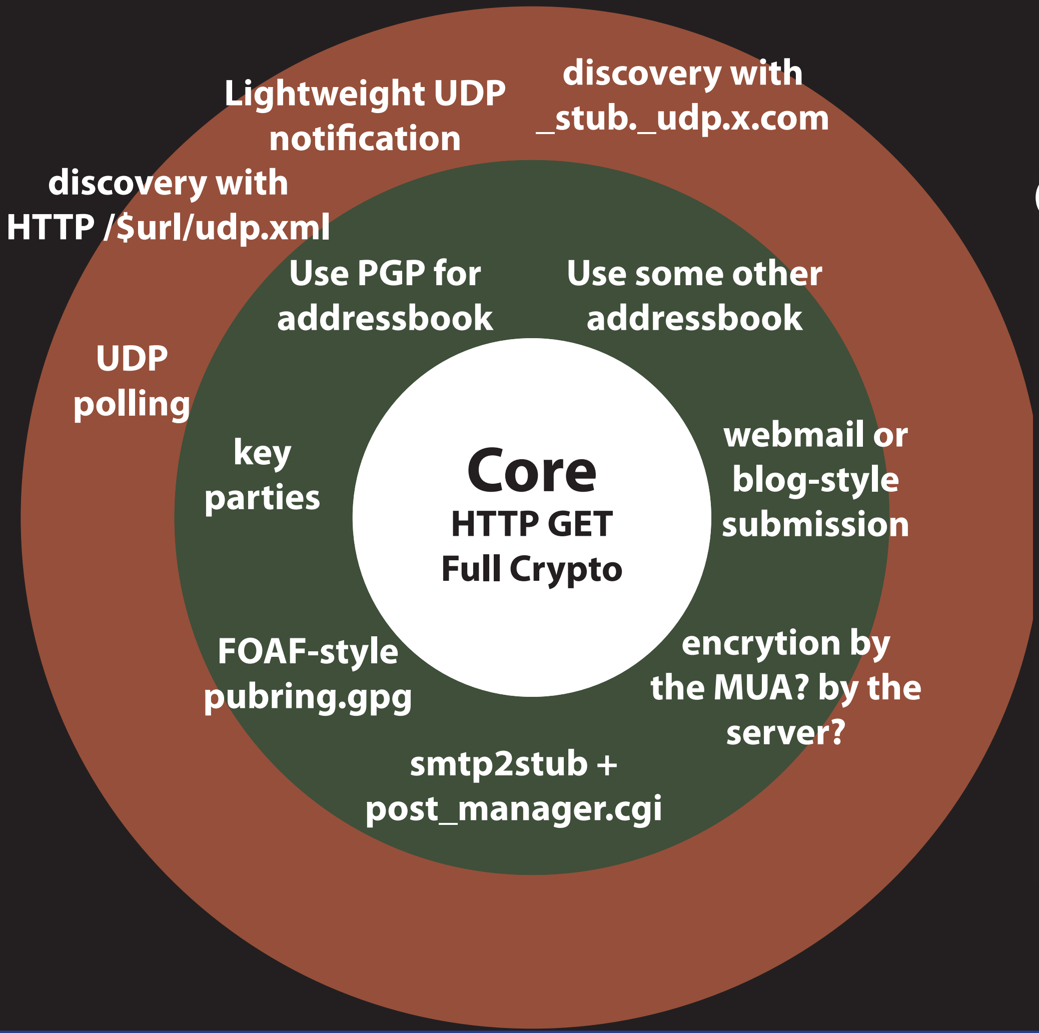
can we make them more lightweight?

Simplest Possible Implementation



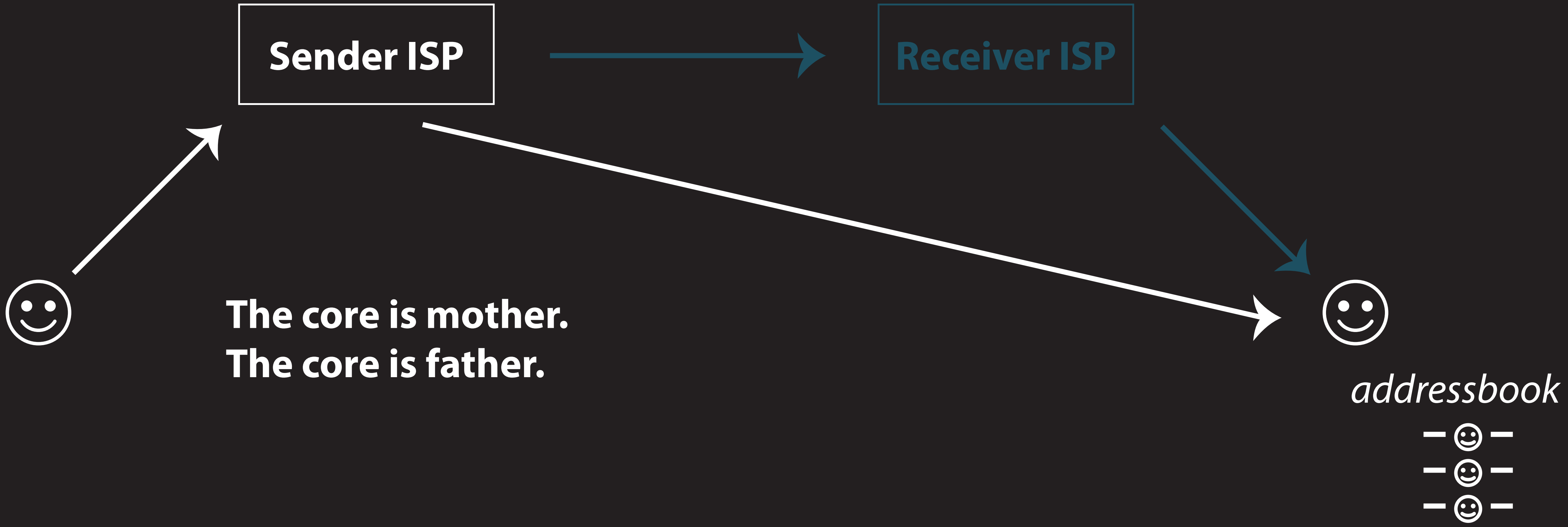
Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- push notification with UDP.
- Receiver ISP stores "dirty list".
- Polling with UDP also.

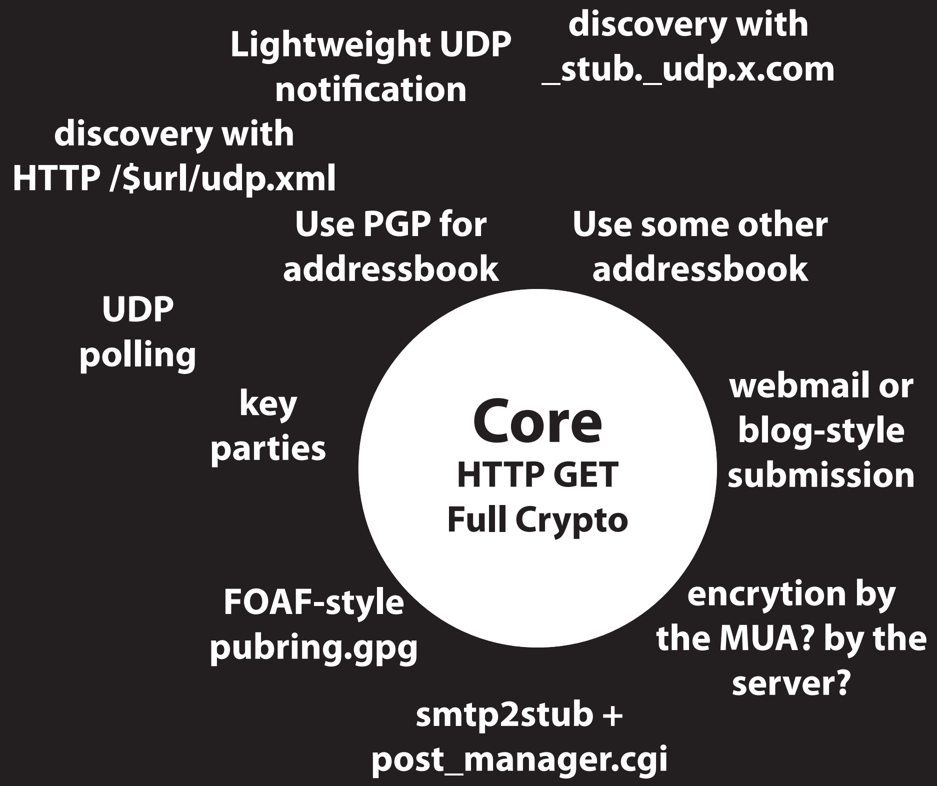

Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
UDP polling
Use PGP for addressbook
Use some other addressbook
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

Simplest Possible Implementation

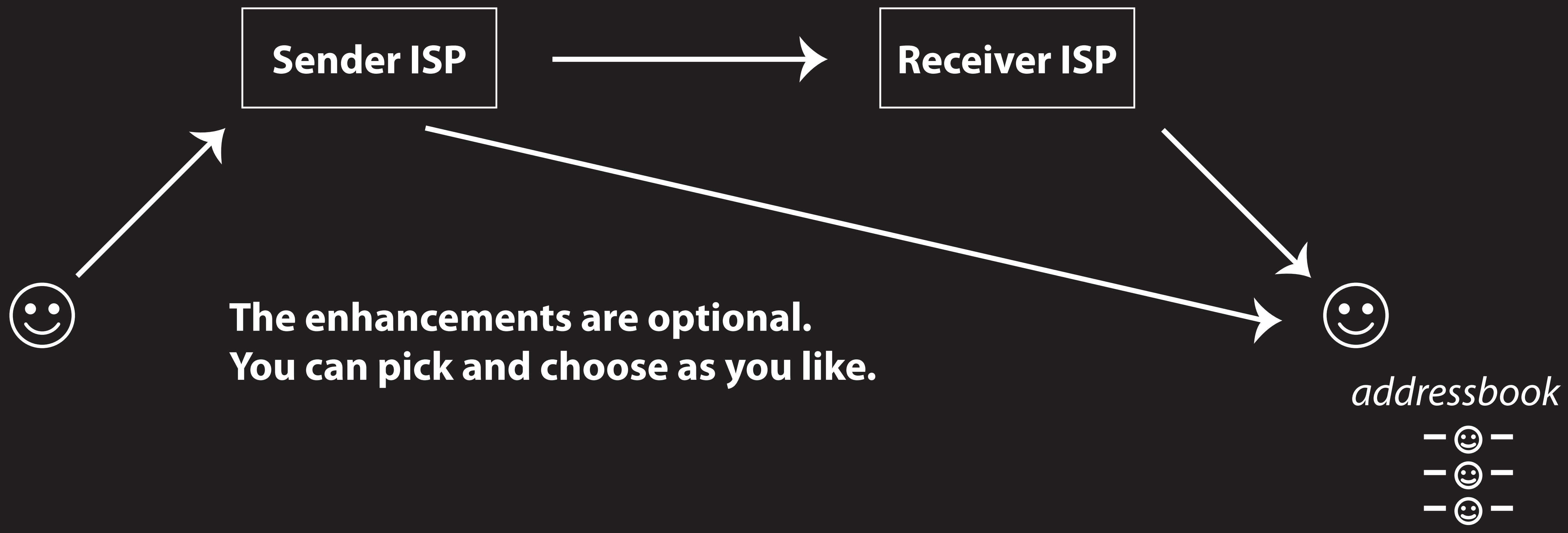


Our Problem
- **retrieve mail over HTTP GET**
- **PGP keyring contains the URLs**
- **SMTP proxy + submission CGI**
- **the sender MUA encrypts?**
- **receiver MTA or MUA decrypts.**
- **push notification with UDP.**
- **Receiver ISP stores "dirty list".**
- **Polling with UDP also.**

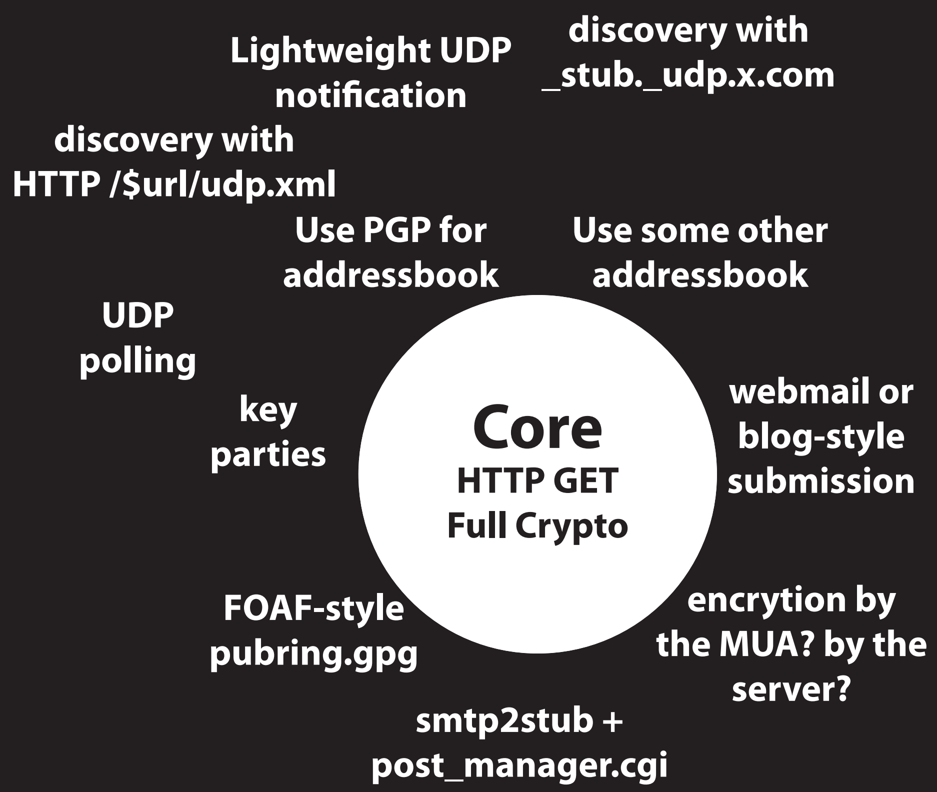

Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
UDP polling
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

Simplest Possible Implementation



The enhancements are optional.
You can pick and choose as you like.

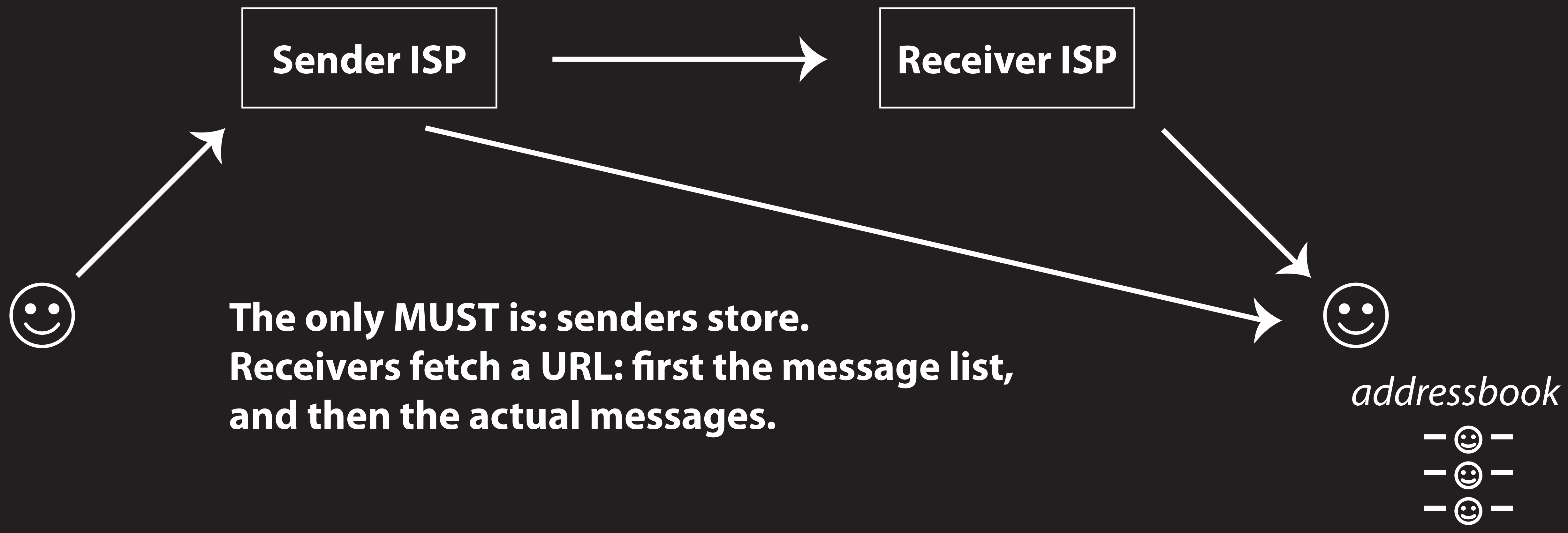

Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
UDP polling
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Our Problem

- retrieve mail over HTTP GET
- PGP keyring contains the URLs
- SMTP proxy + submission CGI
- the sender MUA encrypts?
- receiver MTA or MUA decrypts.
- push notification with UDP.
- Receiver ISP stores "dirty list".
- Polling with UDP also.

Out of Scope

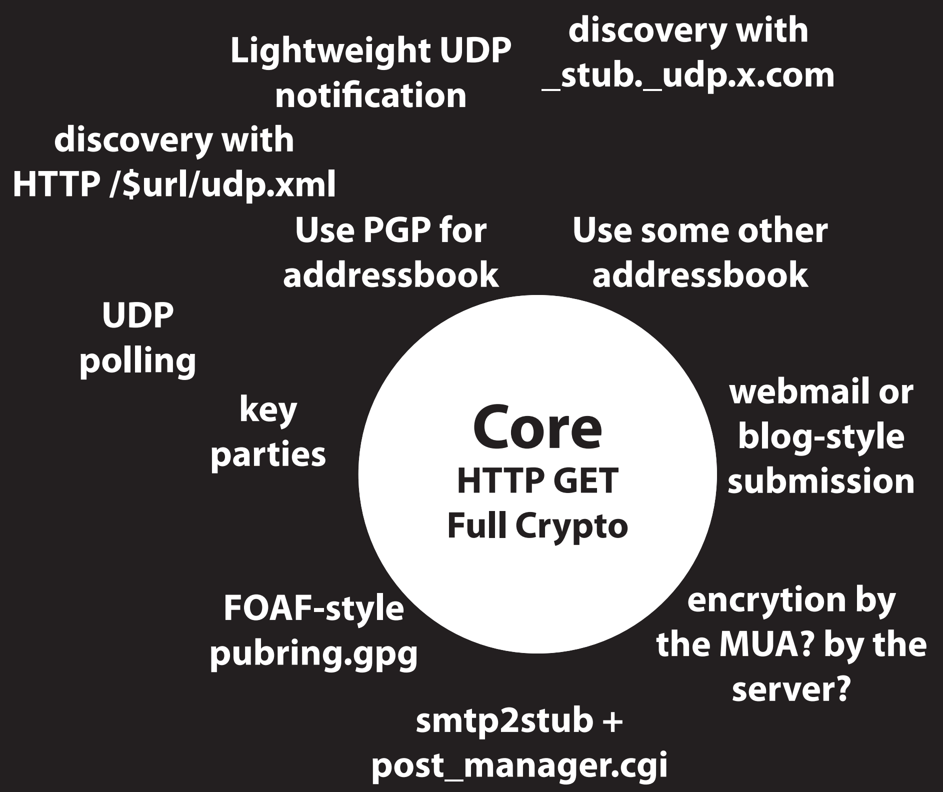
Simplest Possible Implementation



The only MUST is: senders store.
Receivers fetch a URL: first the message list, and then the actual messages.

Our Problem

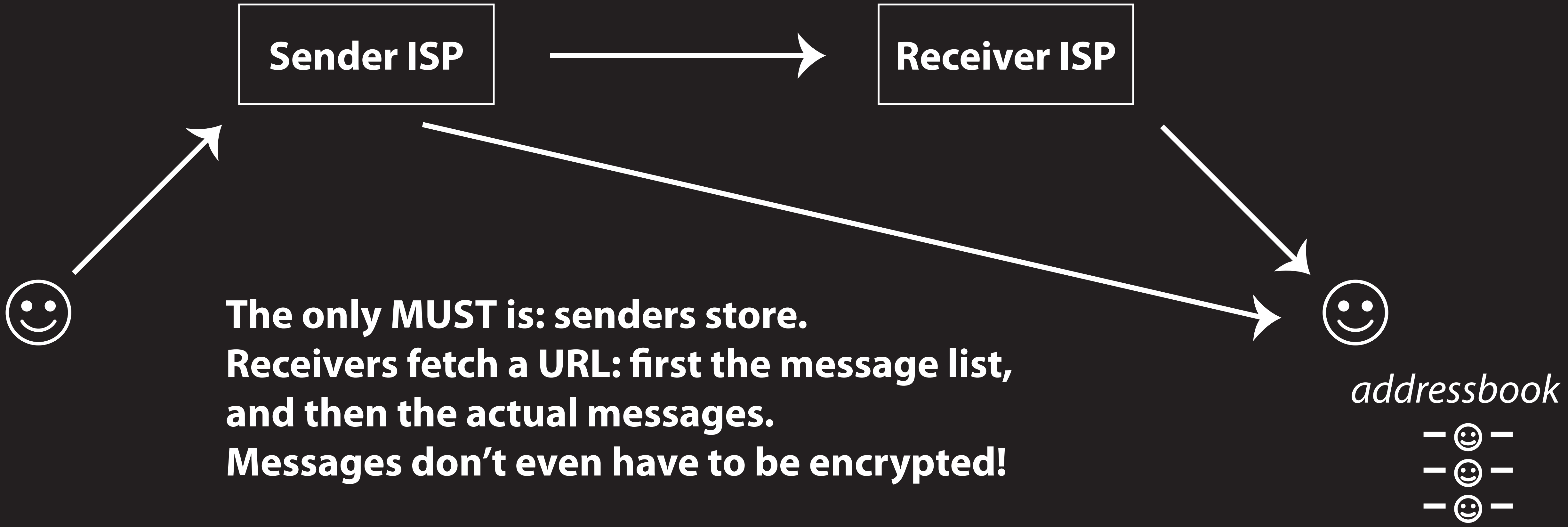
- **retrieve mail over HTTP GET**
- **PGP keyring contains the URLs**
- **SMTP proxy + submission CGI**
- **the sender MUA encrypts?**
- **receiver MTA or MUA decrypts.**
- **push notification with UDP.**
- **Receiver ISP stores "dirty list".**
- **Polling with UDP also.**



Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
UDP polling
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

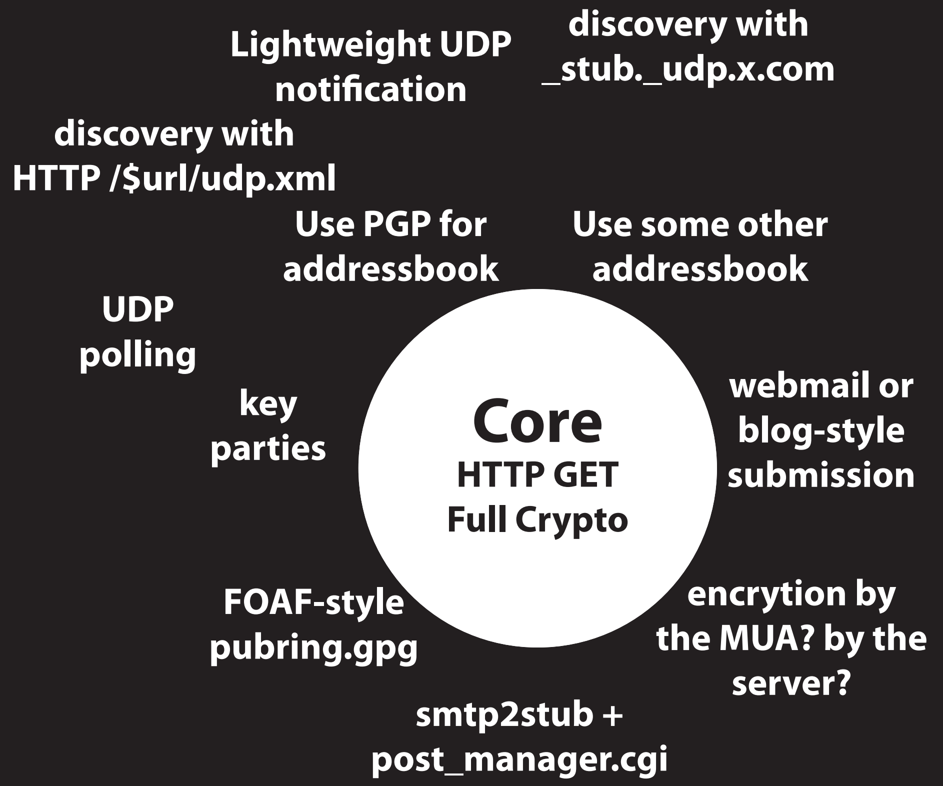
Simplest Possible Implementation



The only MUST is: senders store.
Receivers fetch a URL: first the message list, and then the actual messages.
Messages don't even have to be encrypted!

Our Problem

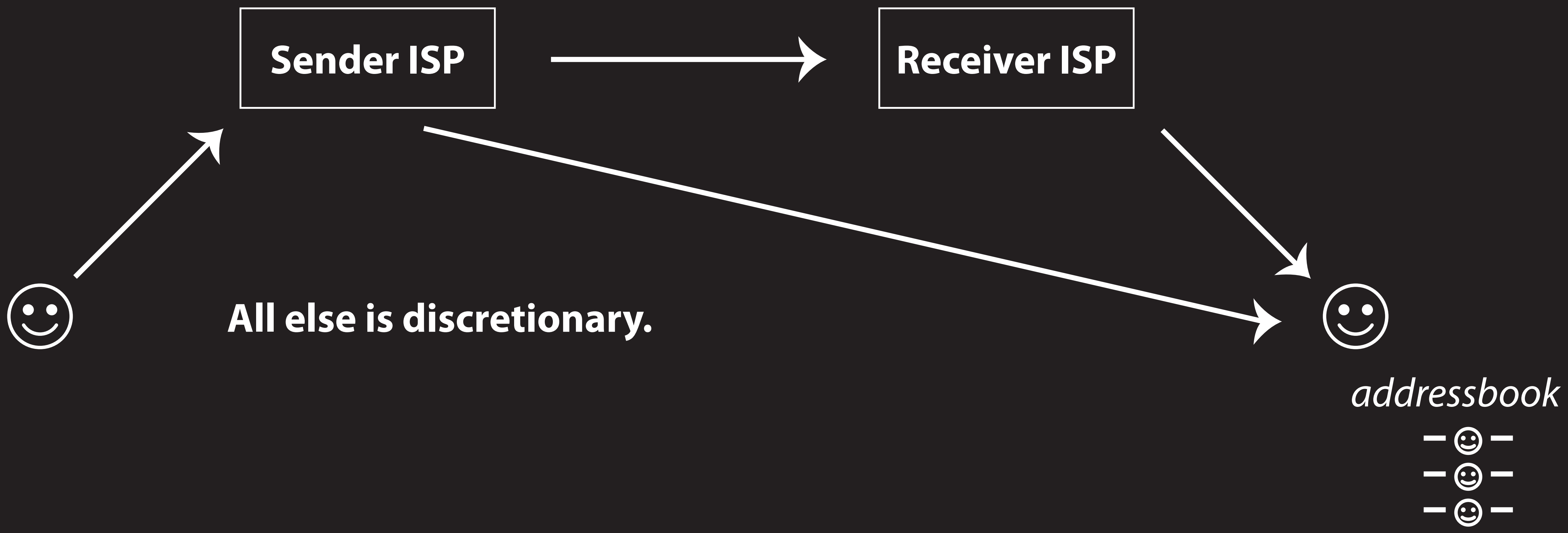
- **retrieve mail over HTTP GET**
- **PGP keyring contains the URLs**
- **SMTP proxy + submission CGI**
- **the sender MUA encrypts?**
- **receiver MTA or MUA decrypts.**
- **push notification with UDP.**
- **Receiver ISP stores "dirty list".**
- **Polling with UDP also.**



Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
UDP polling
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

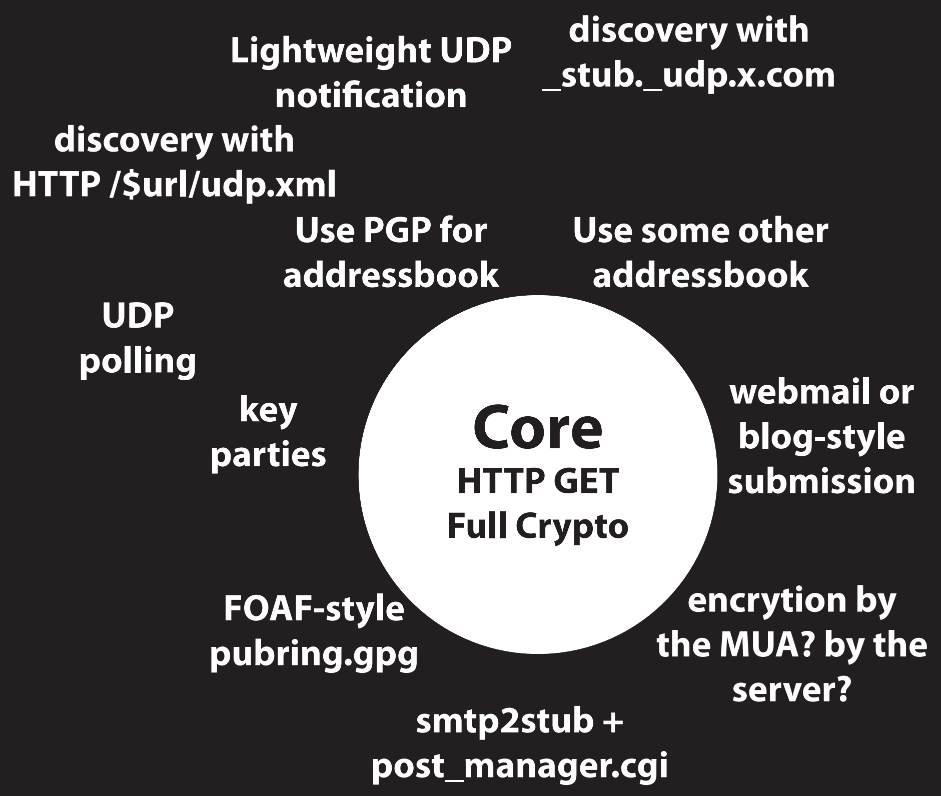
Simplest Possible Implementation



All else is discretionary.

Our Problem

- **retrieve mail over HTTP GET**
- **PGP keyring contains the URLs**
- **SMTP proxy + submission CGI**
- **the sender MUA encrypts?**
- **receiver MTA or MUA decrypts.**
- **push notification with UDP.**
- **Receiver ISP stores “dirty list”.**
- **Polling with UDP also.**


Lightweight UDP notification
discovery with _stub._udp.x.com
discovery with HTTP /$url/udp.xml
Use PGP for addressbook
Use some other addressbook
UDP polling
key parties
Core
HTTP GET
Full Crypto
webmail or blog-style submission
FOAF-style pubring.gpg
encrytion by the MUA? by the server?
smtp2stub + post_manager.cgi

Out of Scope

The Goal:

Meng and Julian can email each other without worrying about spam filtering.

The Goal:

Meng and Julian can email each other without worrying about spam filtering.

Done!

Project History:

April 8 2006.

Meng hosts a hackathon at his crib in Campbell.

- **Julian Haight**
- **Nathan Cheng**
- **Richard Soderberg**

Project History:

July 18 2006:

Meng writes to Julian:

Come here, Watson, I need to tell you that one small step for (a) man is one giant leap for mankind.

Project News:

July 19 2006
Tech Talk at Google

July 26 2006
lightning talk at OScon

Project Status:

totally still a prototype

no wiki yet

no mailing list

(but there should never be :)

volunteers welcome

More:
mengwong.com/rssemail/

Next Talk:
Open Reputation Systems or, Whuffies for the Net

Thanks to Nathan Cheng, Richard Soderberg, Andrei Freeman, Ali Farschian, Nat Torkington, Andy Newton, djb, Leslie Hawthorn, Kathleen Downing

Produced in Adobe InDesign CS2