Cryptography opportunities in Tor

Nick Mathewson
The Tor Project
21 January 2013

Summary

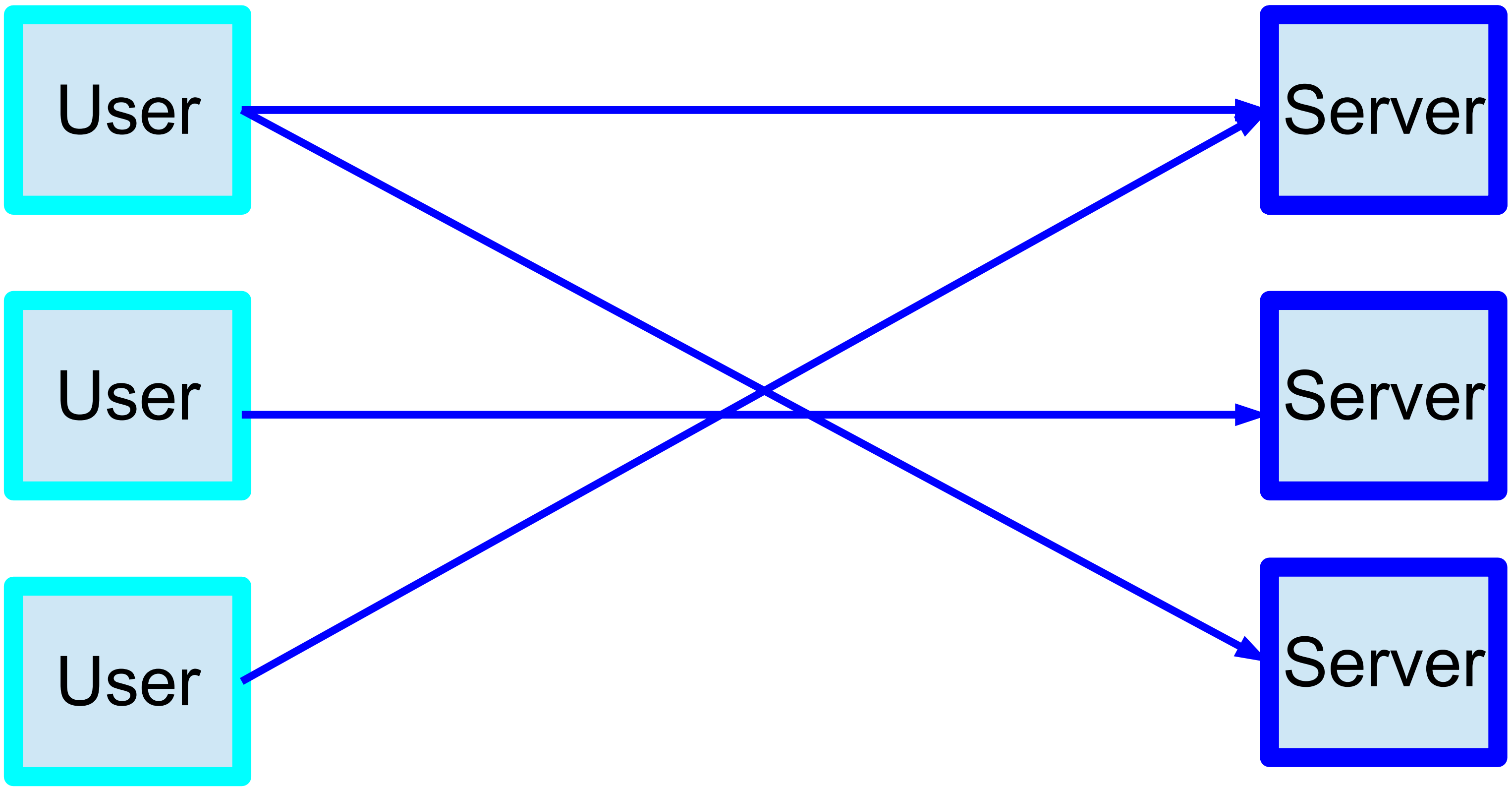
- Very quick Tor overview
- Tor's cryptography, and how it's evolving
- Various opportunities for more Tor crypto work

Disclaimer:

This is not exhaustive; these are only our most interesting crypto needs, not all of them; these are not our most urgent needs in general.

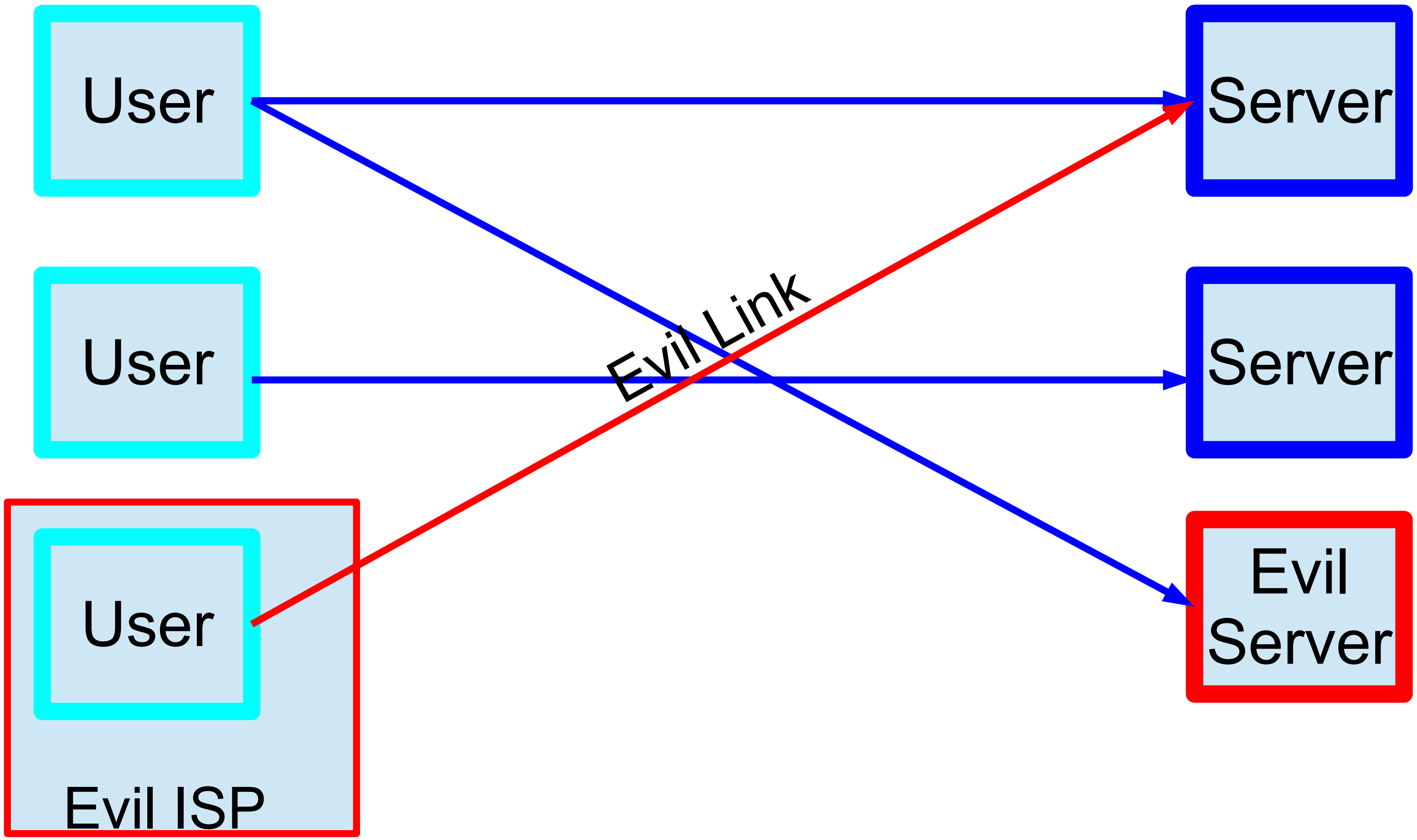
Part 1: Tor overview

Ordinarily, traffic analysis and censorship are easy.

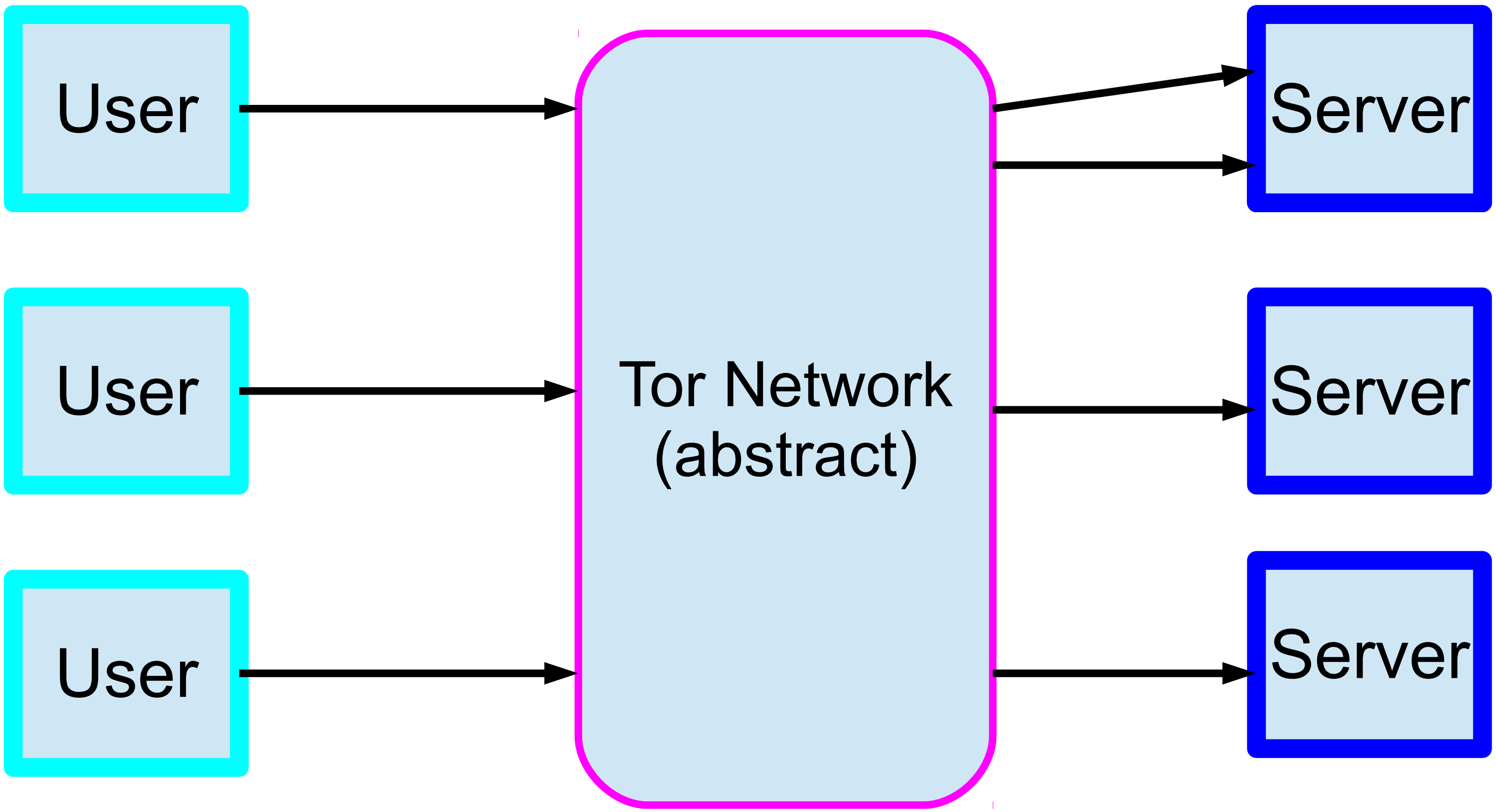


+

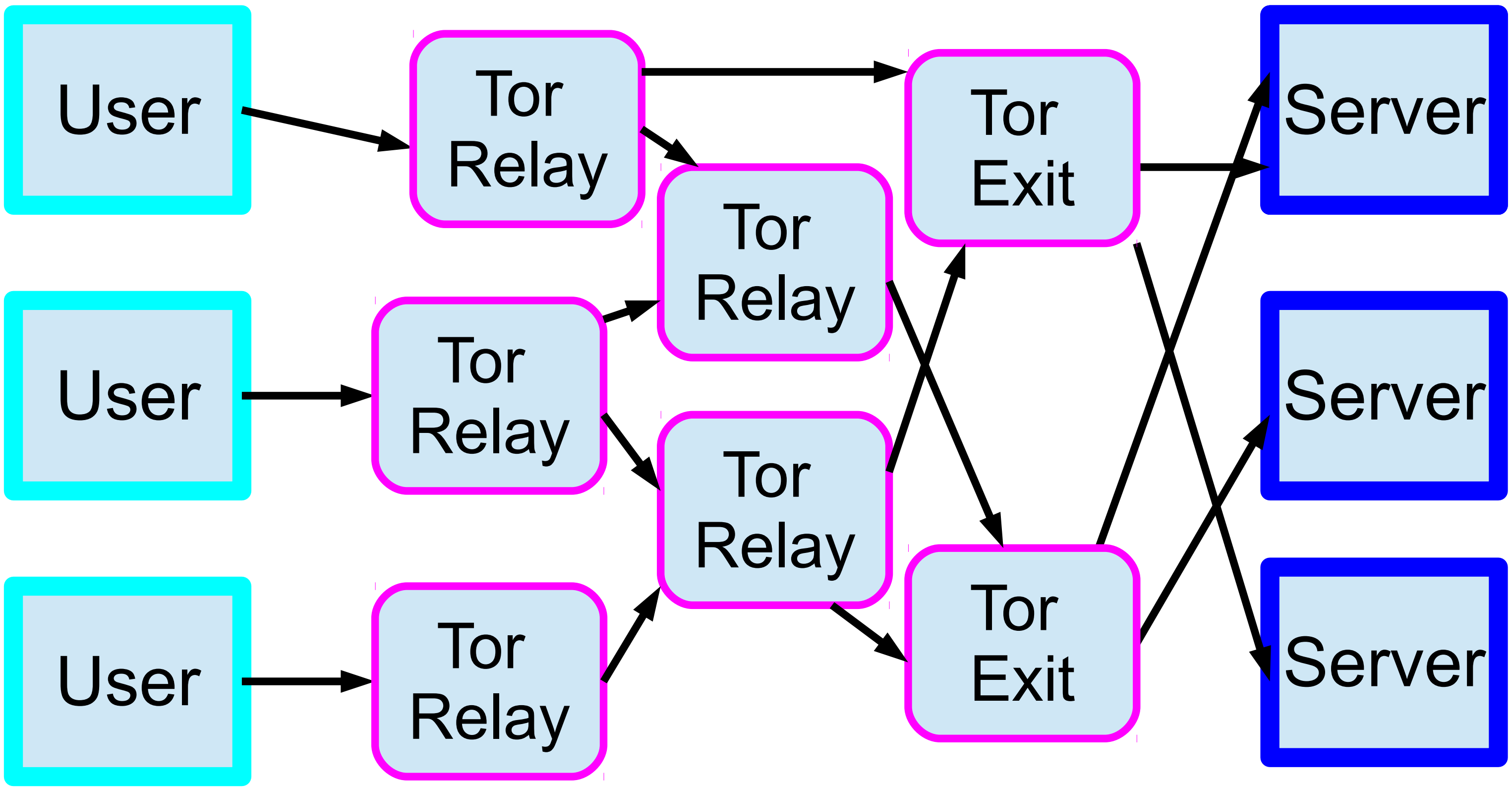
Ordinarily, traffic analysis and censorship are easy.



Tor makes traffic analysis and censorship harder...

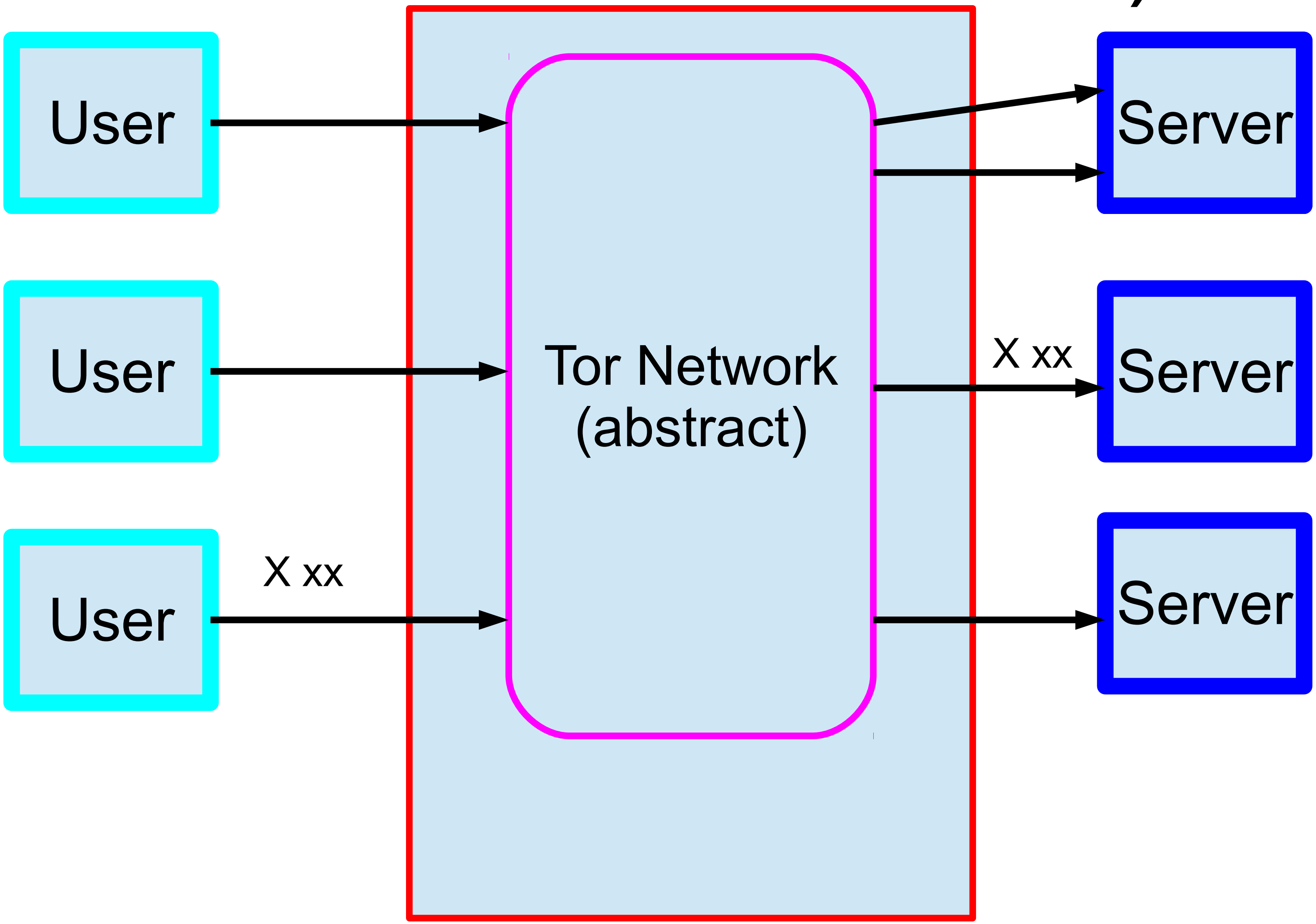


...by using a network of relays to anonymize traffic.



(Use non-public entry relays to resist censorship.)

(But an end-to-end traffic correlation attack still works.)

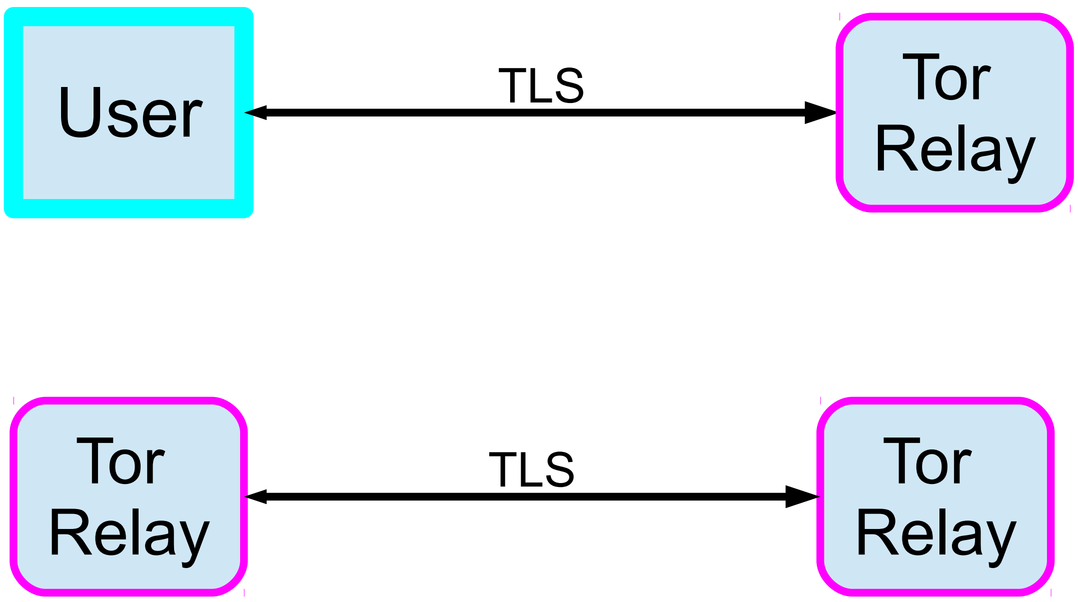


Tor is the largest deployed network of its kind

- 3000 relays
- 1000 public bridges
- > 2 GiB/sec
- > 500,000 users each day (estimated)
  - (With a pretty broad diversity of interest)

Part 2: Tor could use better crypto

Tor uses TLS for its link protocol...

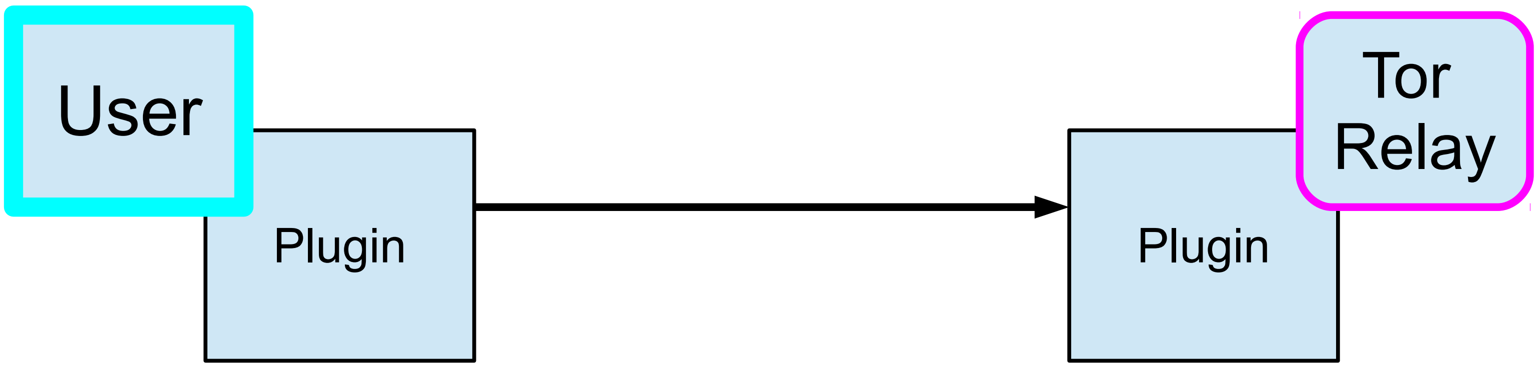


… with all the problems that entails.

- Easy to detect TLS variants based on:
  - Cipher choice
  - Certificate structure
  - List of extensions
- More secure: less common. Can't use any unpopular TLS feature.

(Did you know I have an effective veto over any new TLS features?)

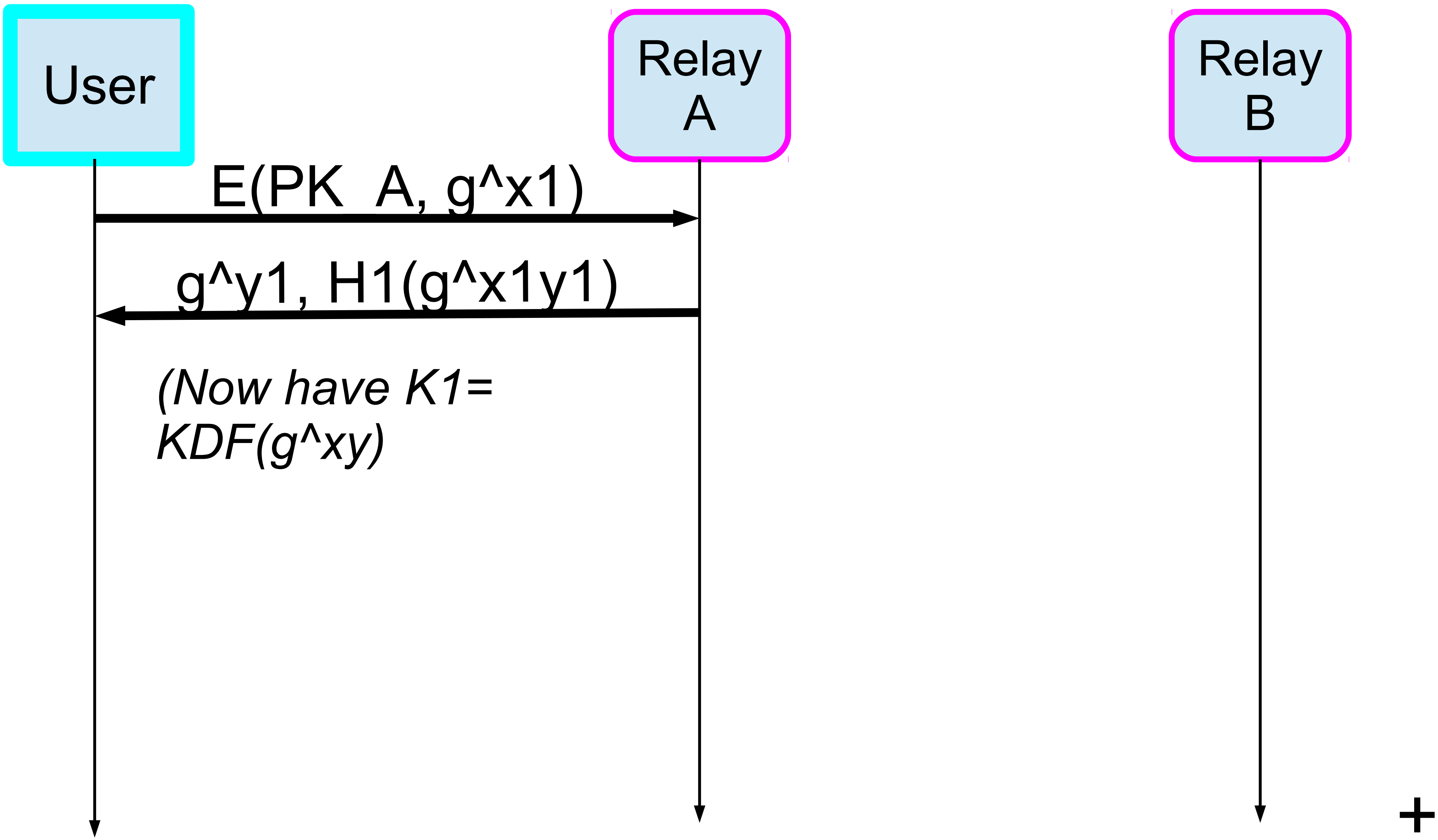
Maybe other link protocols are better for anticensorship?



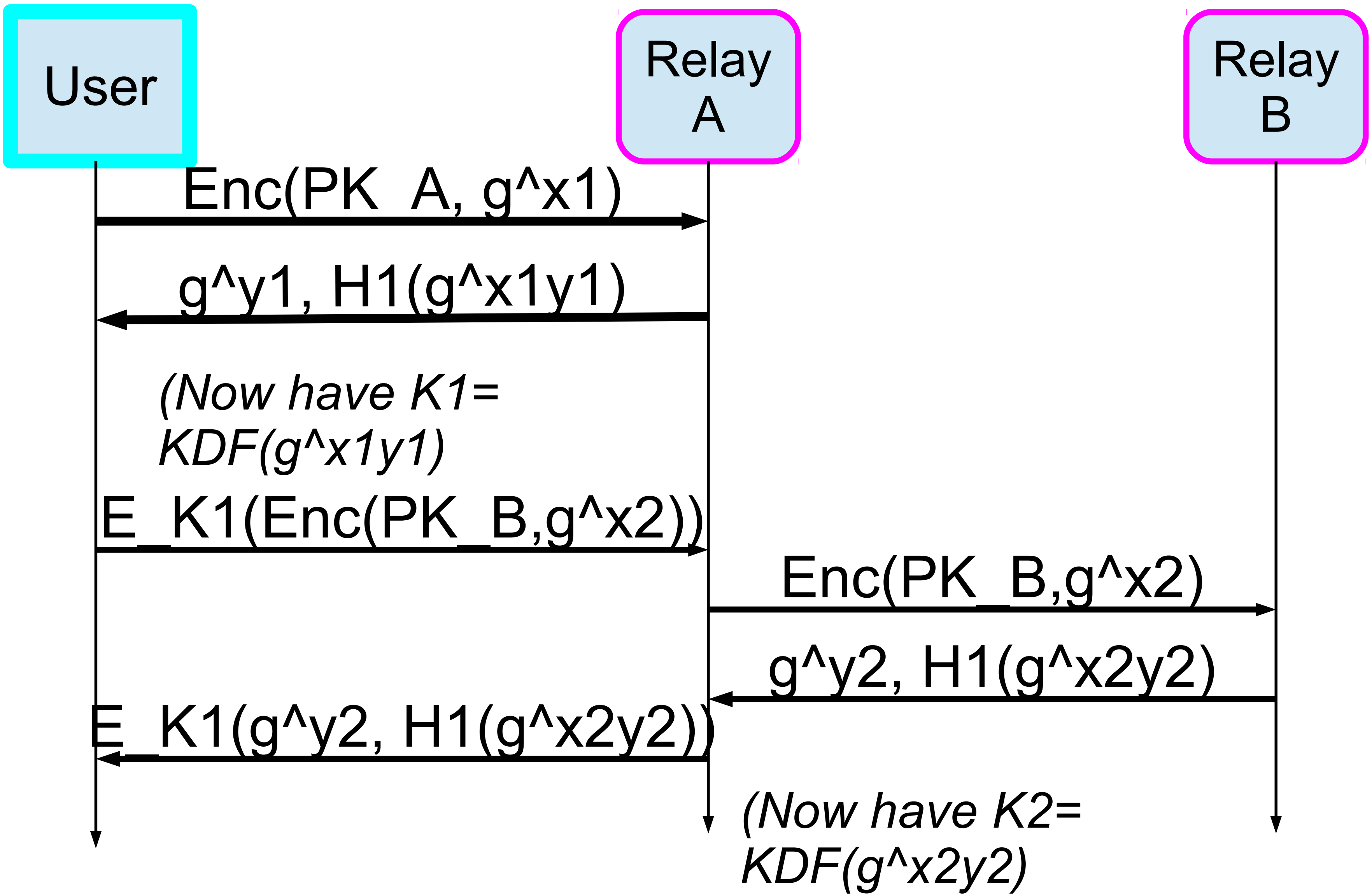
There are a number of these "Pluggable Transports" in development, but we need even more. *Even weak stego can help*.

...Do we still need "normal-looking" TLS?
(If so, hack OpenSSL? port to NSS?)

Tor needs a one-way-authenticated handshake to build circuits



Tor needs a one-way-authenticated key exchange to build circuits



We're replacing this protocol...

- Original protocol (“TAP”) did hybrid encryption with RSA,DH-1024, badly. [Goldberg 2006]
- Enc(PK,g^x) was:
  - Let K = random 128-bit AES key.
  - Split 1024-bit g^x into 70-byte X1, 58-byte X2
  - Result is: RSA1024_OAEP_ENC(K||X1) || AES_CTR(K,X2)
- Note 1024-bit PK; note malleability on 2nd part.

We're replacing this protocol...

- Replacement (“ntor”) does ***approximately***
- Client: (given server public key B)
  - Generate keypair x, X=g^x
  - Send B, g^x
- Server: (given server private key b)
  - Generate y, Y=g^y. Let secret = X^y || X^b || ID || B || X || Y || PROTOID
  - Let auth = H_verify(secret) || ID || B || Y || X || PROTOID || “Server”
  - Send Y, H_mac(auth). Derive keys.
- Client: Compute secret, auth.

  [Goldberg, Stebila, Ustaoglu 2011]

  (We're using DJB's curve25519 for DH group)

...and could optimize it more...

- Replacement ("ntor") does ***approximately***
- Client: (given server public key B)
  - Generate keypair x, X=**g^x**
  - Send B, g^x
- Server: (given server private key b)
  - Generate y, Y=**g^y**. Let secret = X^y || X^b || ID || B || X || Y || PROTOID
  - Let auth = H_verify(secret) || ID || B || Y || X || PROTOID || "Server"
  - Send Y, H_mac(auth). Derive keys.
- Client: Compute secret, auth.

Fixed basepoint!

[Goldberg, Stebila, Ustaoglu 2011]

(We're using DJB's curve25519 for DH group)

...and could optimize it more...

- Replacement ("ntor") does ***approximately***
- Client: (given server public key B)
  - Generate keypair x, X=g^x
  - Send B, g^x
- Server: (given server private key b)
  - Generate y, Y=g^y. Let secret = **X^y** || **X^b** || ID || B || X || Y || PROTOID
  - Let auth = H_verify(secret) || ID || B || Y || X || PROTOID || "Server"
  - Send Y, H_mac(auth). Derive keys.
- Client: Compute secret, auth.

  [Goldberg, Stebila, Ustaoglu 2011]

  (We're using DJB's curve25519 for DH group)

Simultaneous, same base. Use batch exponentiation?

...and might even do better!

- Alternative (“ace”) does approximately:

  Client:
  - Send X1=g^x1, X2=g^x2

  Server:
  - Send Y=g^y
  - Compute S = (X1^b) (X2^y) = g^[b(x1) + y(x2)]
- Client:
  - Compute S= (B^x1)(y^x2) = g^[b(x1) + y(x2)]

  [Backes, Kate, Mohammedi 2012]

  (Is this better? Are the optimizations worth it?)

We should replace our old relay cell protocol...

- Used for symmetric crypto once we have shared keys.

Zeros (2)	Bad “MAC” (4)	Payload (503)

+

We should replace our old relay cell protocol...

- Used for symmetric crypto once we have shared keys.

<table>
<tr><td>Zeros (2)</td><td>Bad “MAC” (4)</td><td>Payload</td></tr>
<tr><td colspan="3">AES_CTR(Key1)</td></tr>
<tr><td colspan="3">AES_CTR(Key2)</td></tr>
<tr><td colspan="3">AES_CTR(Key3)</td></tr>
</table>

We should replace our old relay cell protocol...

- Used for symmetric crypto once we have shared keys.

Zeros (2)	Bad “MAC” (4)	Payload
AES_CTR(Key1)		
AES_CTR(Key2)		
AES_CTR(Key3)		

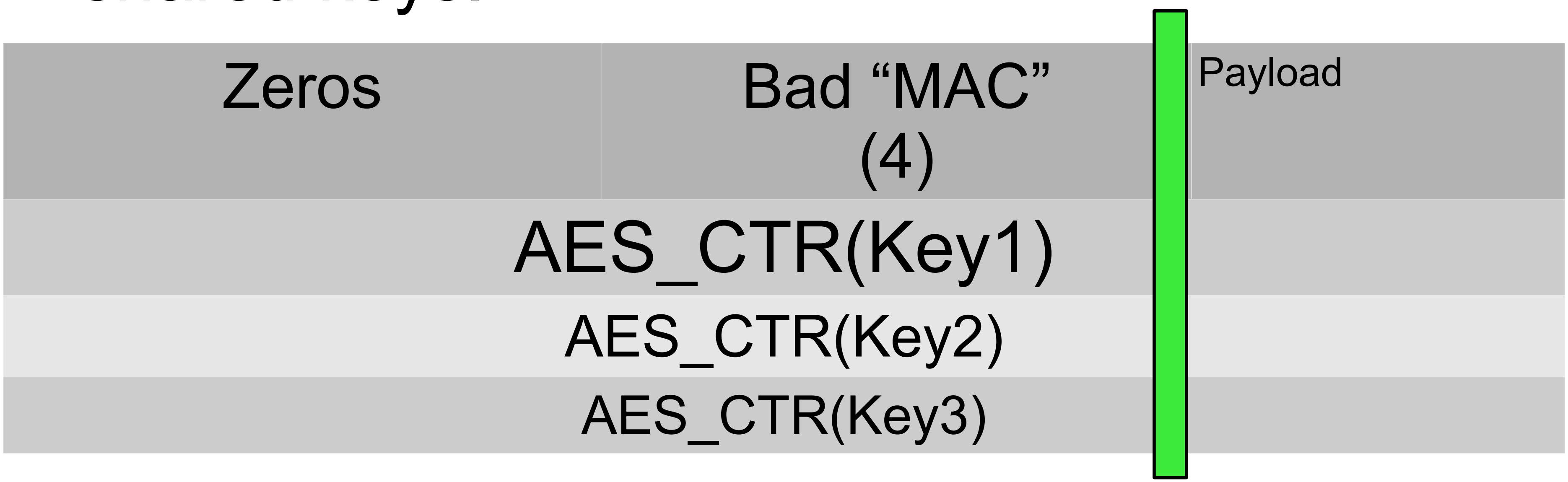
To handle a cell:

- Remove a layer of encryption.
- If Zeros == 0, and “MAC” = H(Key3_M, Previous cells | Payload):

  This cells is for us!
- Else, relay the cell

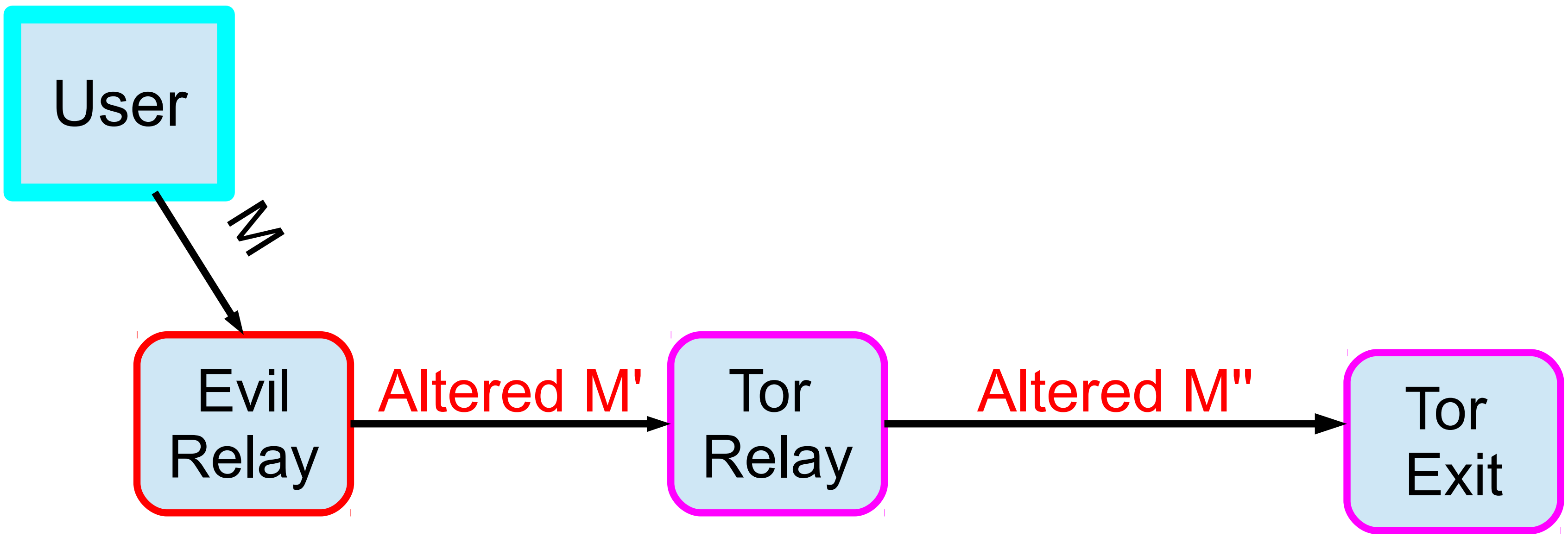
We should replace our old relay cell protocol...

- Used for symmetric crypto once we have shared keys.



But this is malleable!

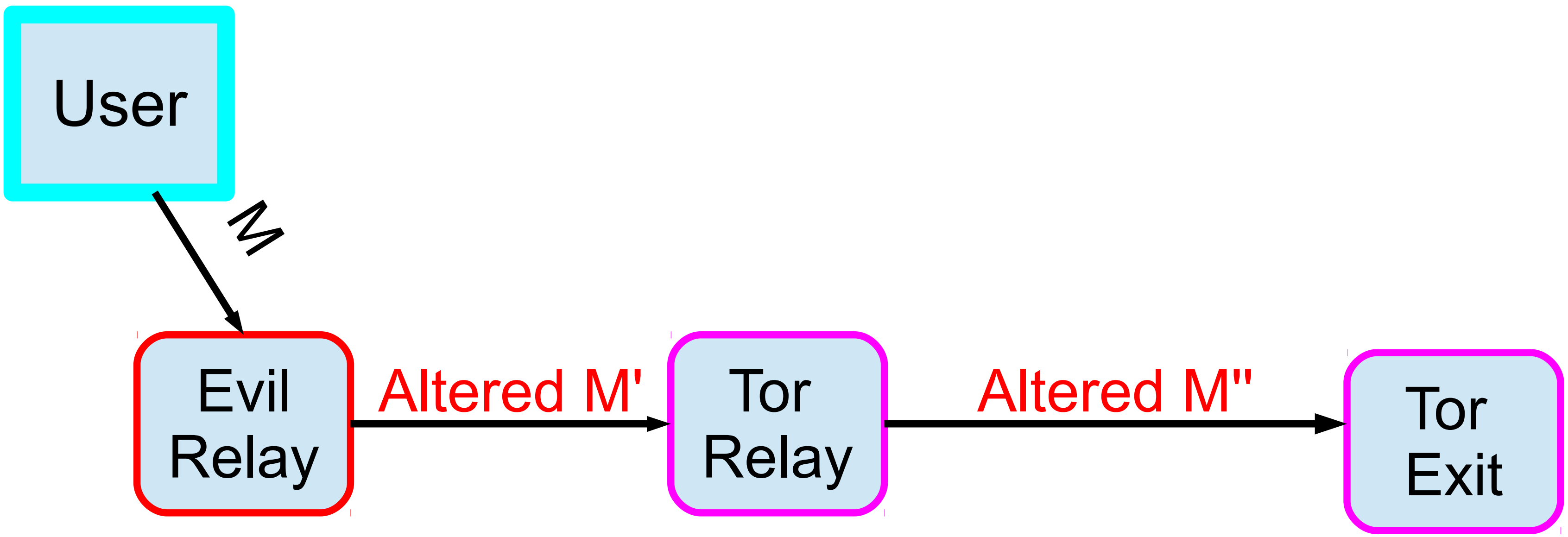
Hang on, does it matter that it's malleable?



- Honest exit (probably) rejects M''
- Evil exit detects tag, but could just as easily do traffic correlation, for same result at less risk of detection.
- So, don't worry? (Dingledine, Mathewson, Syverson 2004)

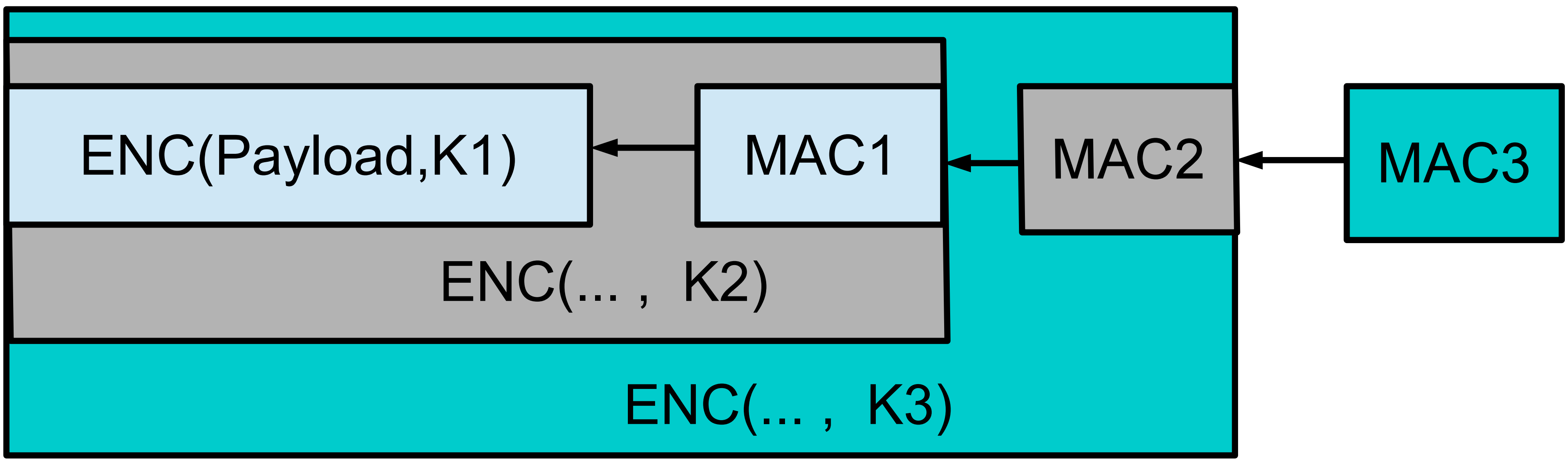
+

Hang on, does it matter that it's malleable?



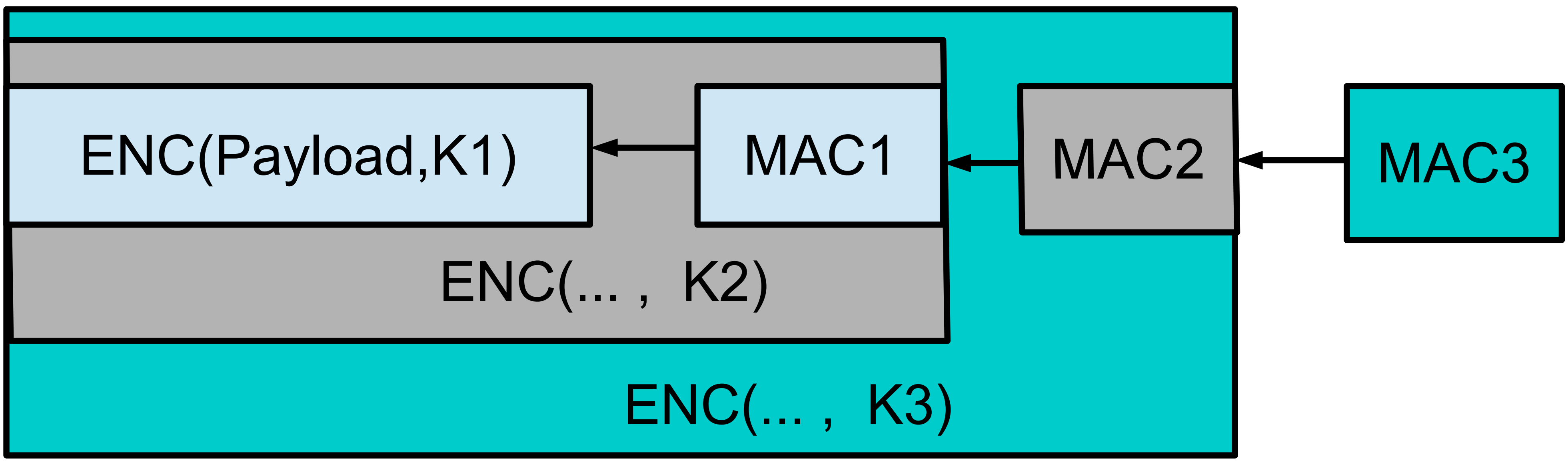
- Honest exit (probably) rejects M''
- Evil exit detects tag, but could ~~*just as easily*~~ do traffic correlation, ~~*for same result*~~ at less risk of detection.
- *Actually, it's not so clear-cut.*

We could use an encrypt-and-mac structure



+

We could use an encrypt-and-mac structure



But that requires one MAC per hop, and leaks path length.

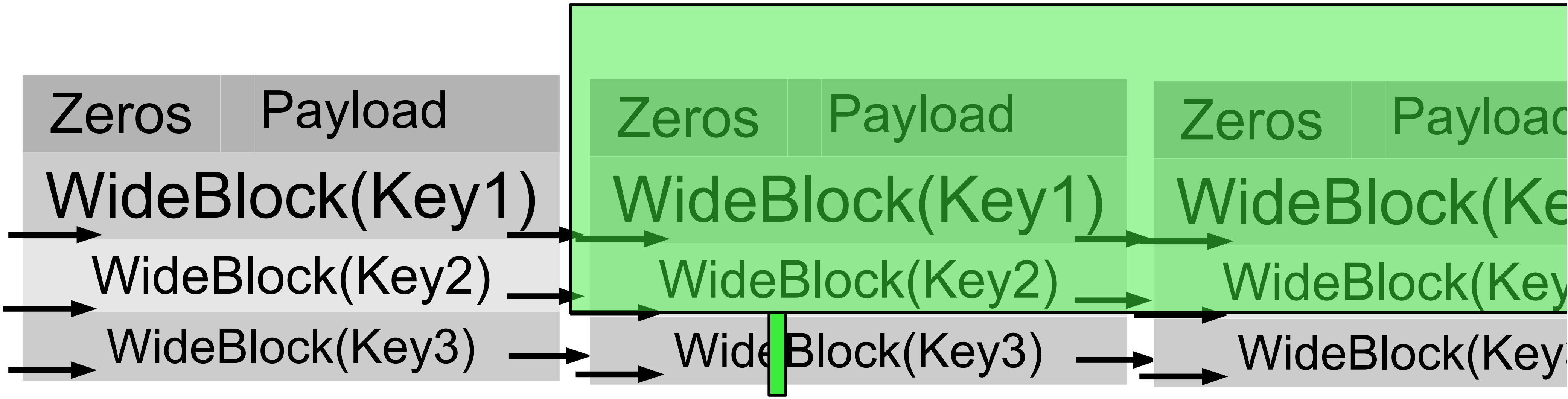
A chained wide-block cipher seems like a much better idea!

+

A chained wide-block cipher seems like a much better idea!

Any attempt to change the block renders the whole block unrecoverable...

A chained wide-block cipher seems like a much better idea!



Any attempt to change one block renders the whole circuit unrecoverable...

What wide-block cipher to use?

- Not enough time to discuss all of them (LIONESS, CMC, XCB, HCTR, XTS, XEX, HCH, TET)
- Needs to be fast, proven, secure, easy-to-implement, non-patent-encumbered, side-channel-free,...
- One promising approach in progress by Bernstein, Sarkar, and Nandi – HFFH Feistel structure, fast, not yet finished.
- CAESAR may produce more.
- Other ideas?

There are more crypto issues in Tor

- Directory protocol
- Hidden service protocol
- Link protocol
- Better DOS resistance (SSL is teh sux)
- SHA1, RSA1024 for node identity

Questions?

- See https://www.torproject.org/ for links to documentation, specifications, and more info about various Tor issues.
- See http://freehaven.net/anonbib/ for an incomplete but nonetheless useful anonymity bibliography.
- Grab me during a break for non-crypto Tor questions