ETH zürich

Counter Galois Onion (CGO): Fast Non-Malleable Onion Encryption for Tor

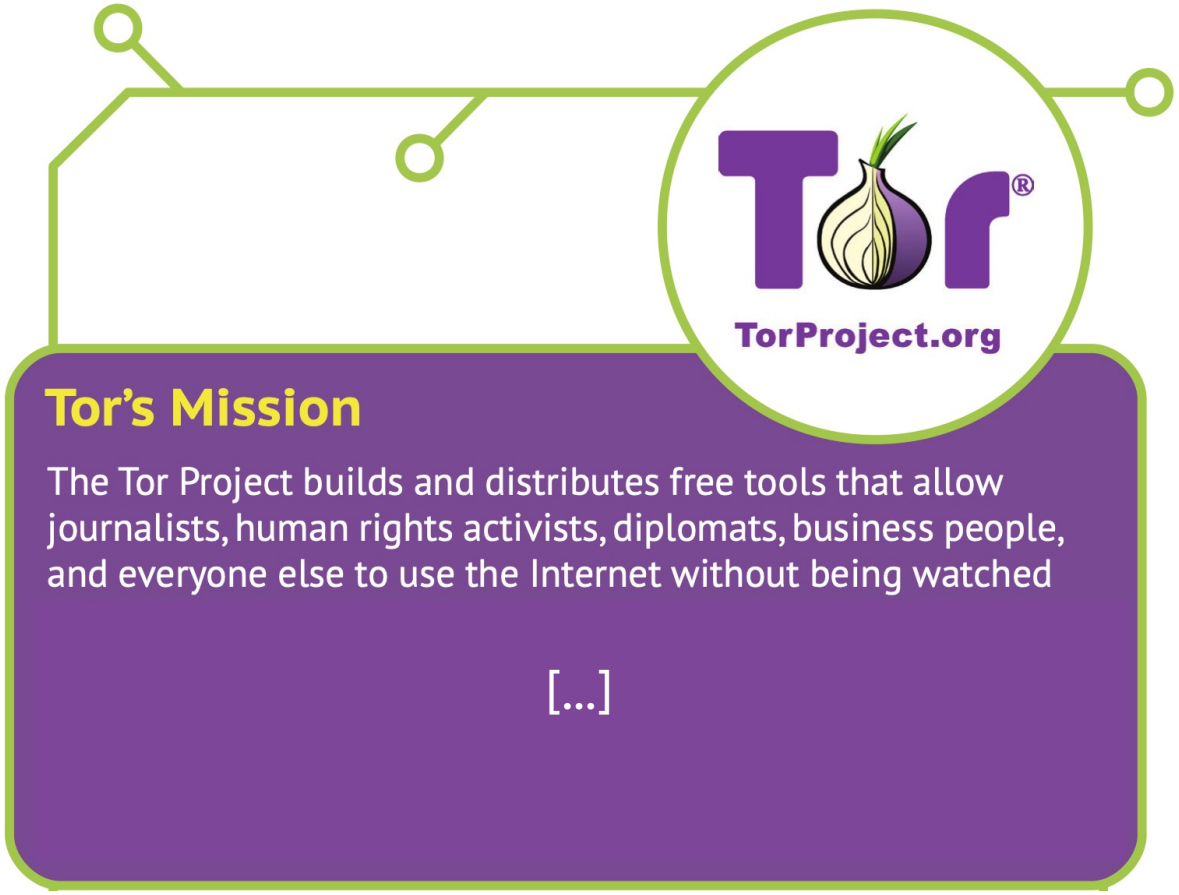
Jean Paul Degabriele, Alessandro Melloni, Jean-Pierre Münch, and *Martijn Stam*

Davide Fiocchi

Roadmap

1. **Motivation**
2. Tor refresher
3. What we have now, and why it’s bad
4. Cryptographic preliminaries
5. Counter Galois Onion
6. Conclusions

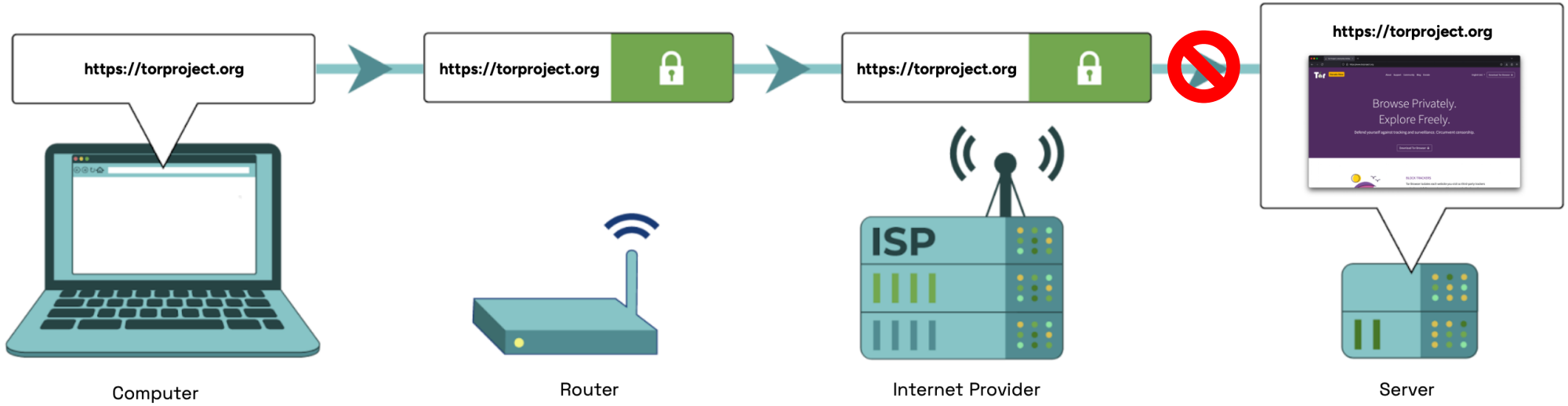
What the paper is about



- 2002: Tor alpha released
- 2012: Tor design proposal 202 expresses the need to upgrade Tor's onion encryption scheme

“We kill people based on metadata”

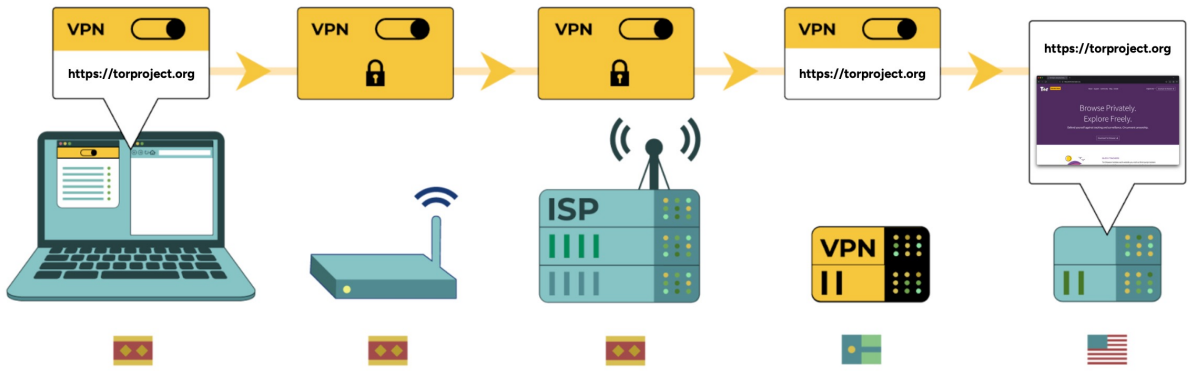
- An Internet Service Provider can view, store and block:
  - IP addresses ⇒ TCP/IP blocking
  - DNS requests ⇒ DNS tampering (hijacking/spoofing)
  - SNIs ⇒ SNI filtering
  - Packets ⇒ Deep packet inspection



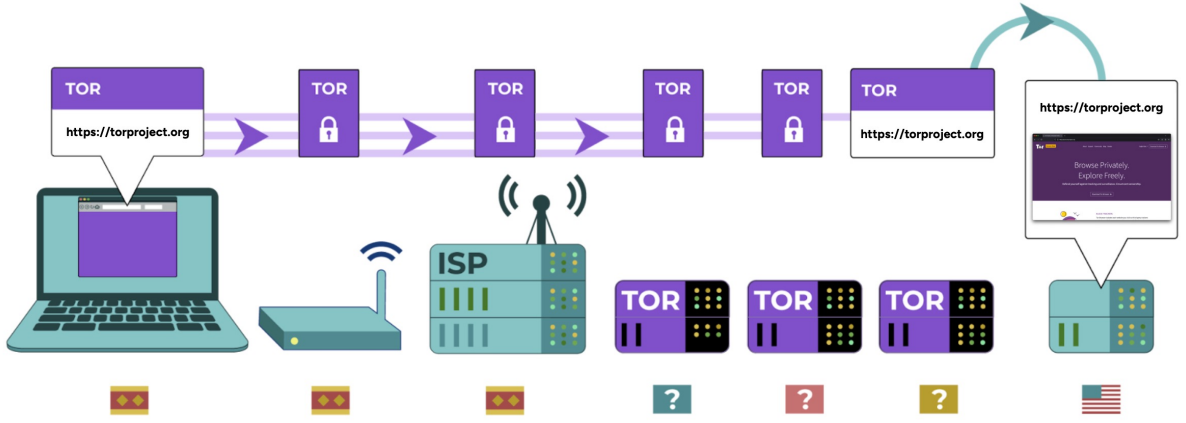
Avoiding censorship (3 “easy” steps)

1. Appear as if you're intending to visit an unblocked location,
2. protect your traffic along the way,
3. access the intended destination through the proxy.

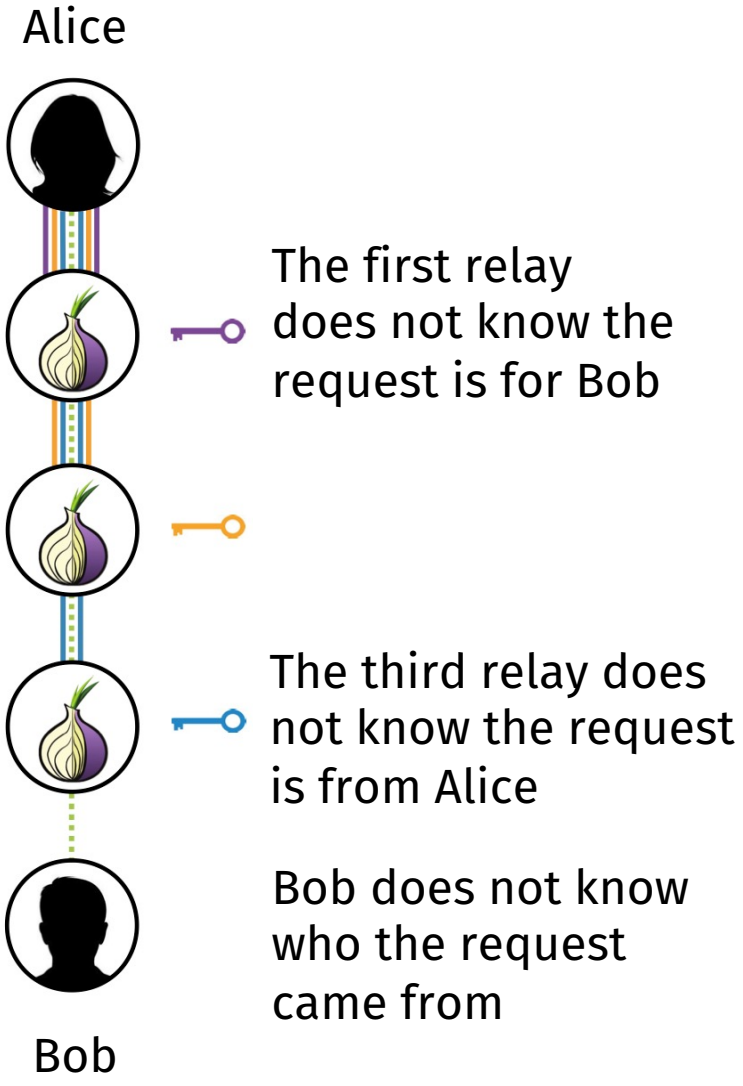
Encrypted Tunnels (e.g. VPNs)



Anonymizing Tunnels (e.g. Tor)



From torproject.org



Key Points on Anonymizing Tunnels

- You **don't need to trust** the node operators, or even Tor itself.
- [...]

Roadmap

1. Motivation
2. **Tor refresher**
3. What we have now, and why it’s bad
4. Cryptographic preliminaries
5. Counter Galois Onion
6. Conclusions

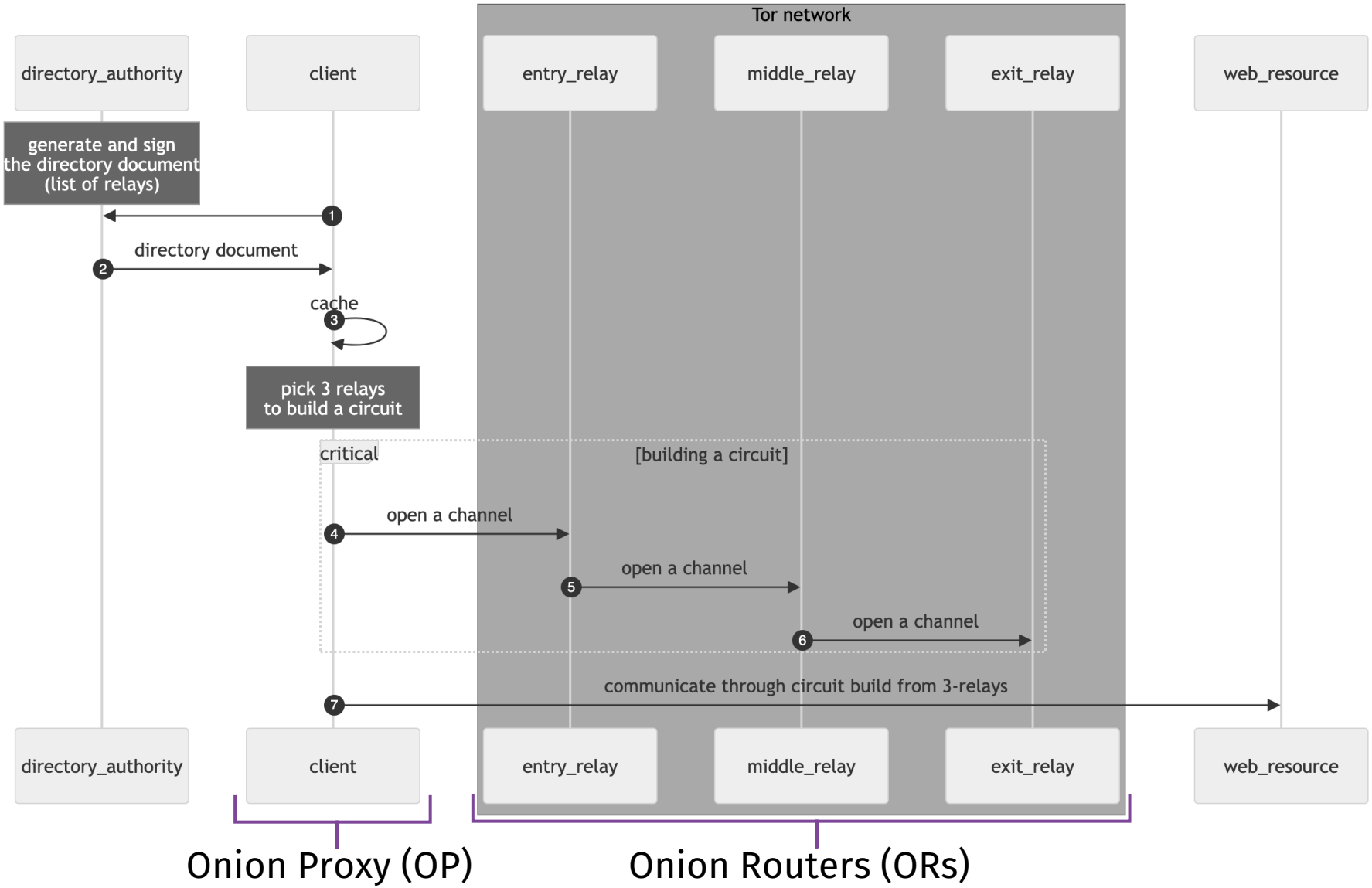
The Onion Routing 101

Tor aims to provide a

low-latency, circuit-based, bidirectional secure channel

between two parties, through a network of onion routers with the aim of obscuring exactly who is talking to whom, even in the presence of adversaries controlling part of the network [5].

The Onion Routing 101

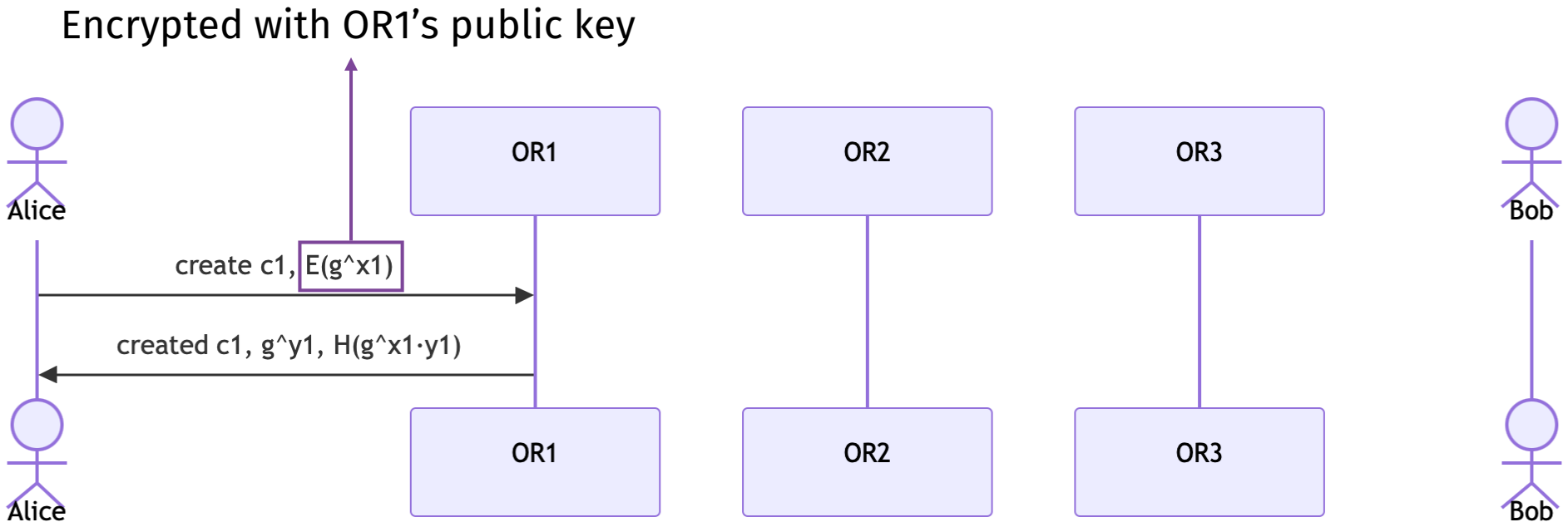


The Onion Routing 101

- **LINK** protocol (section 4)
  - secures communication between any pair of nodes
  - per-hop TLS connection “channels”
  - version and parameters negotiation
- **CIRCUIT EXTEND** protocol (section 5)
  - establishes multi-hop tunnels called **circuits**
  - uses public-key cryptography to exchange key material
- **RELAY** protocol (section 6)
  - uses the established key material to perform onion encryption
- **STREAM** protocol (section 6.2)
  - operates over the relay protocol
  - multiplexes multiple TCP connections (“streams”) over a single circuit

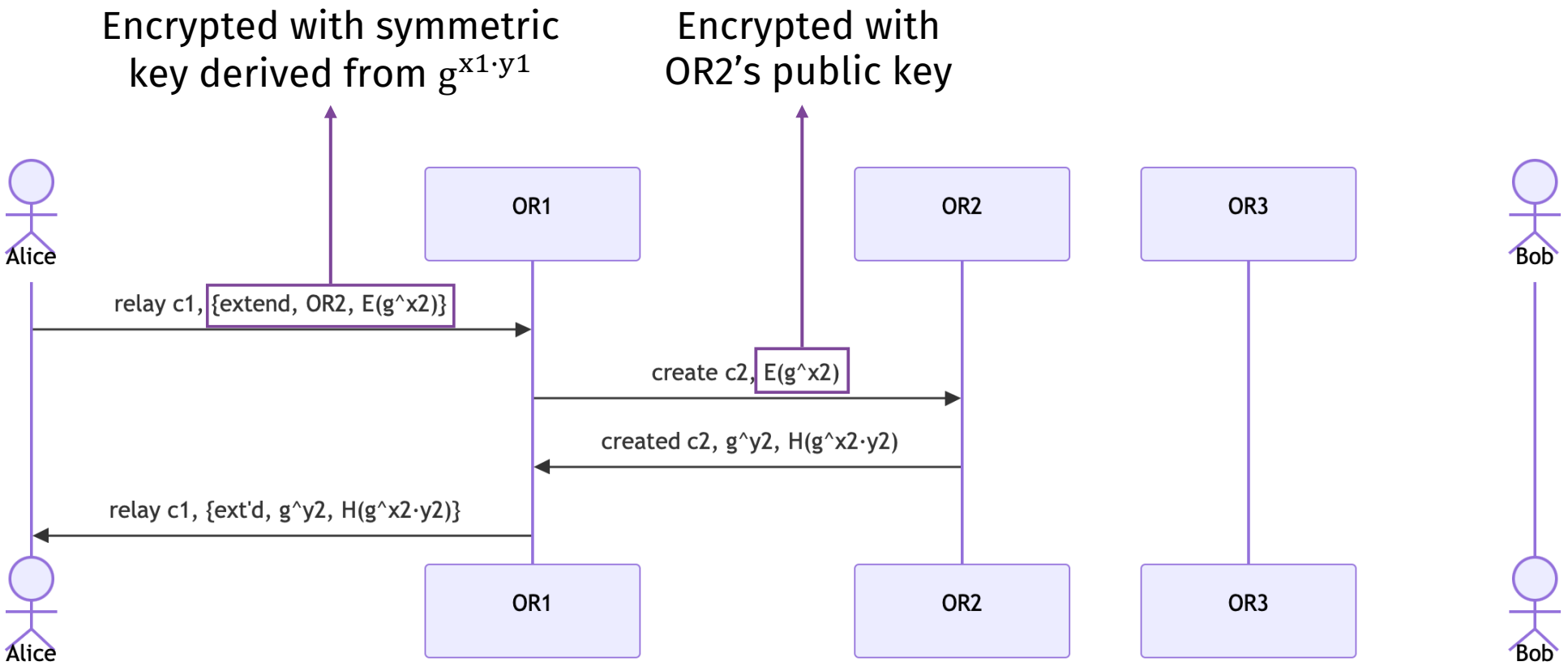
Questi dettagli si possono nascondere

Anatomy of an onion



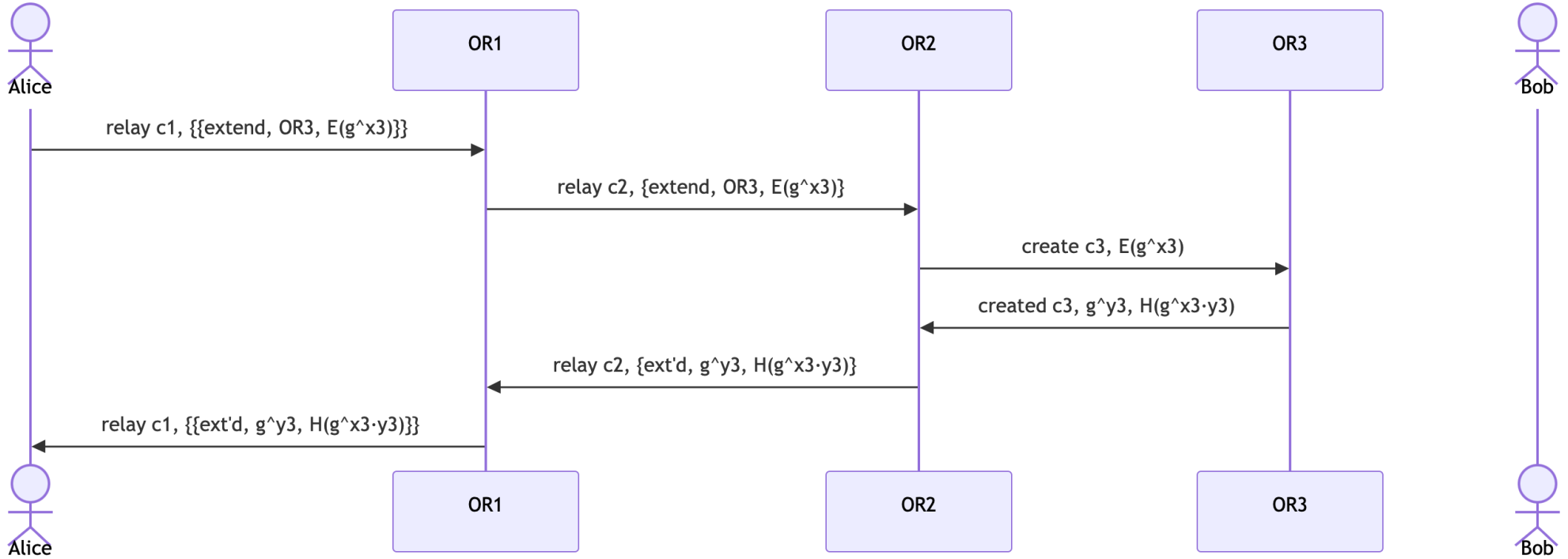
NOTE: this is the old “TAP” handshake, modern “ntor” handshake differs slightly

Anatomy of an onion



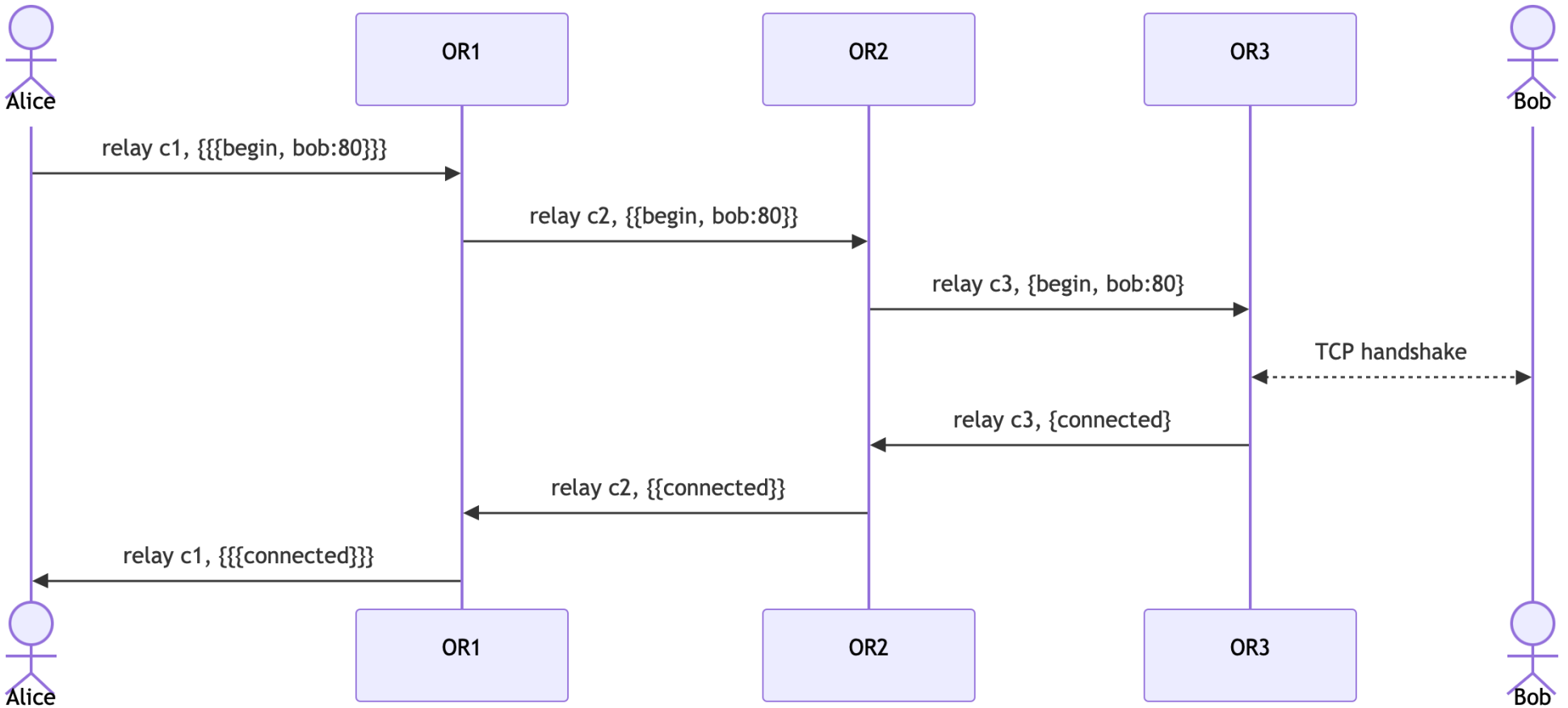
NOTE: this is the old “TAP” handshake, modern “ntor” handshake differs slightly

Anatomy of an onion

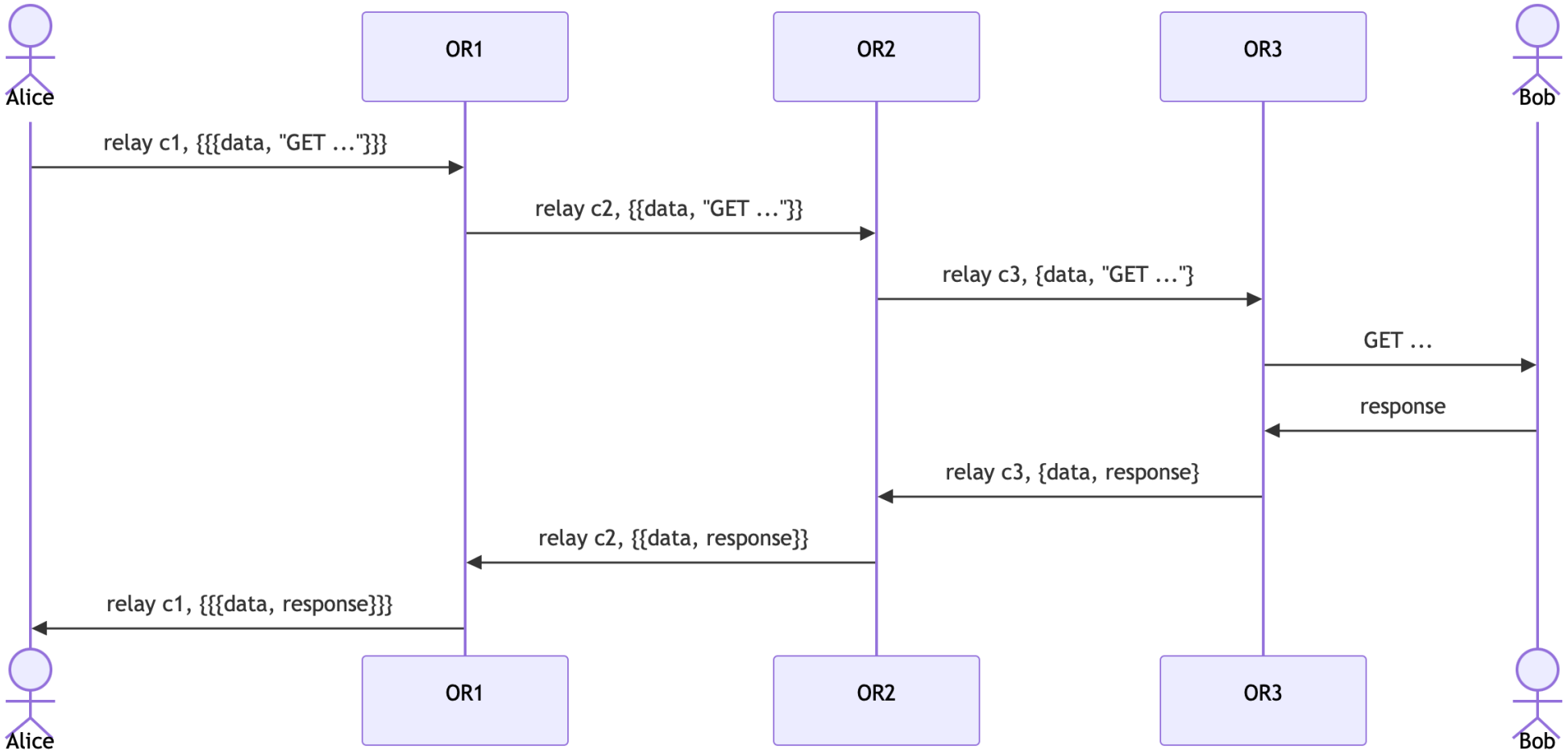


NOTE: this is the old “TAP” handshake, modern “ntor” handshake differs slightly

Anatomy of an onion

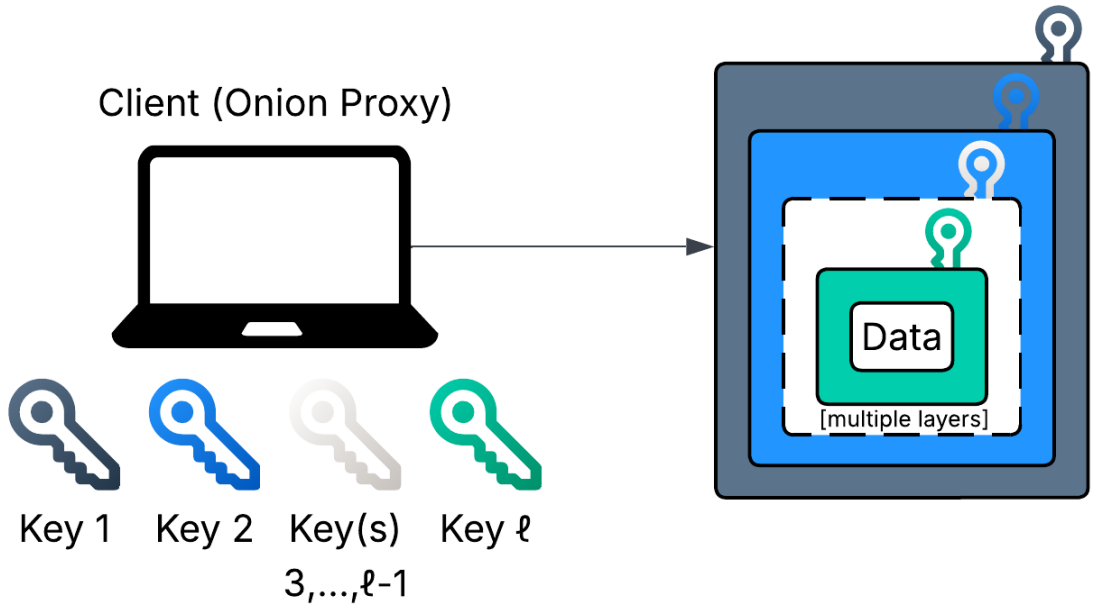


Anatomy of an onion



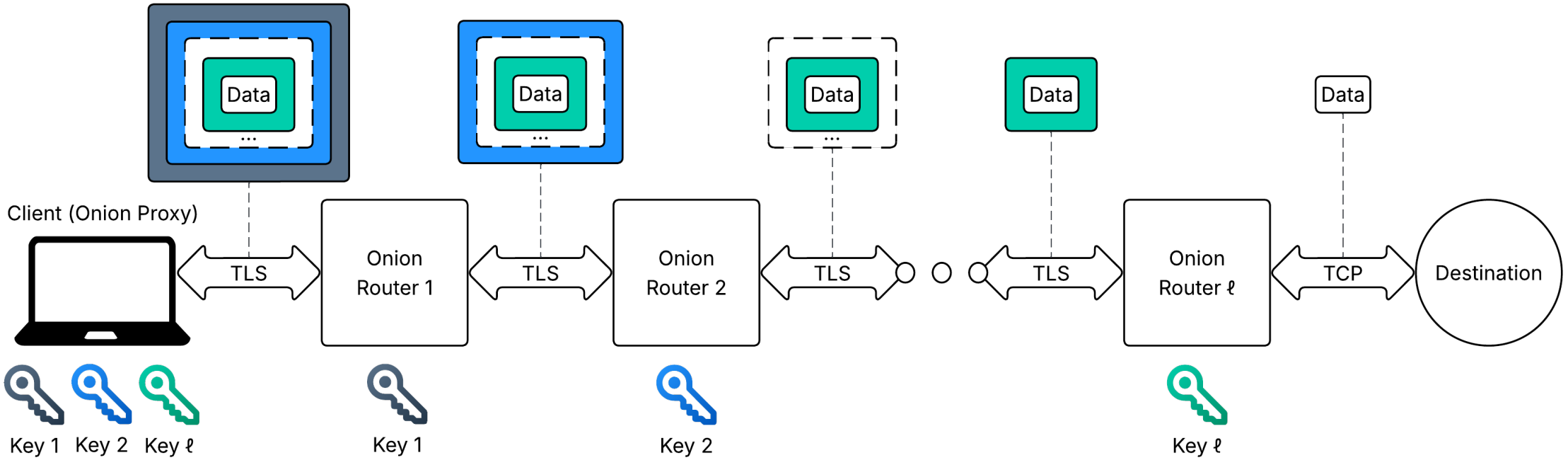
The relay protocol – onion encryption

- The sender (OP) shares a symmetric key with every OR on the circuit, and applies one layer of encryption for each



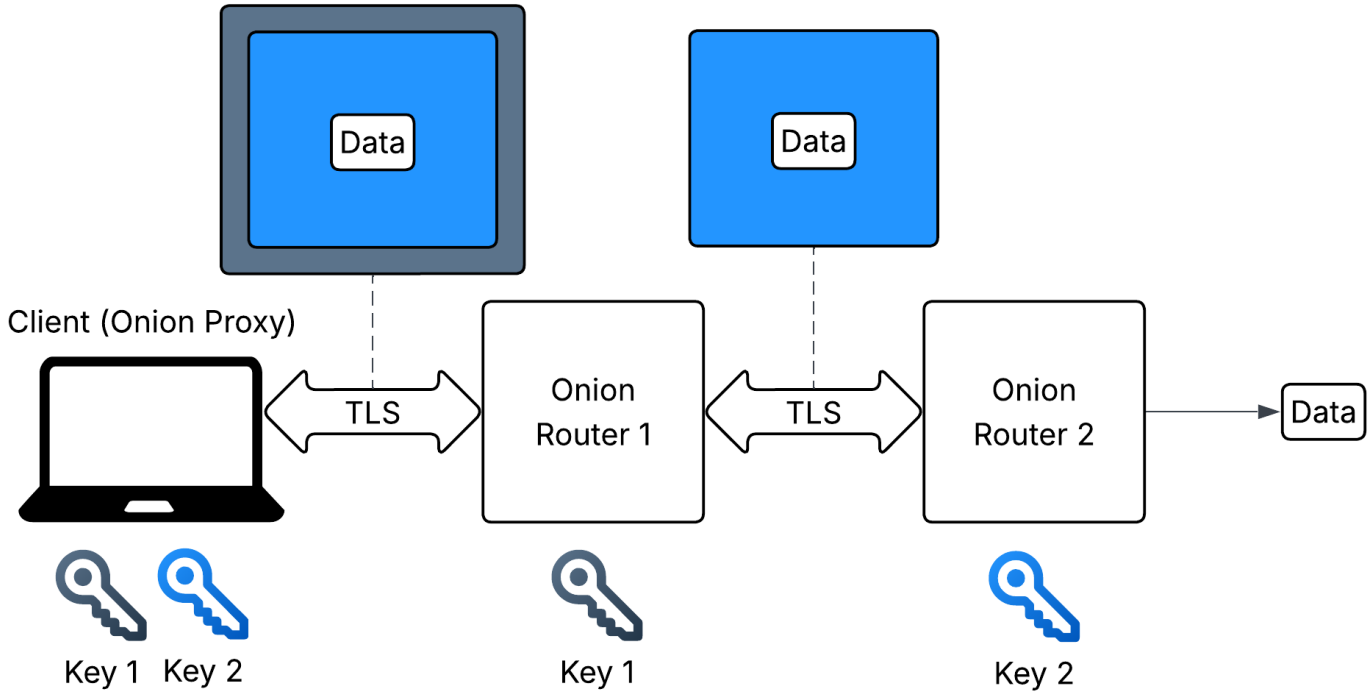
The relay protocol – onion encryption

- Each OR strips off one layer and either:
  - forwards the cell to the next node in the circuit, or



The relay protocol – onion encryption

- Each OR strips off one layer and either:
  - acts on the cell itself if it is the intended recipient (”leaky pipes”)

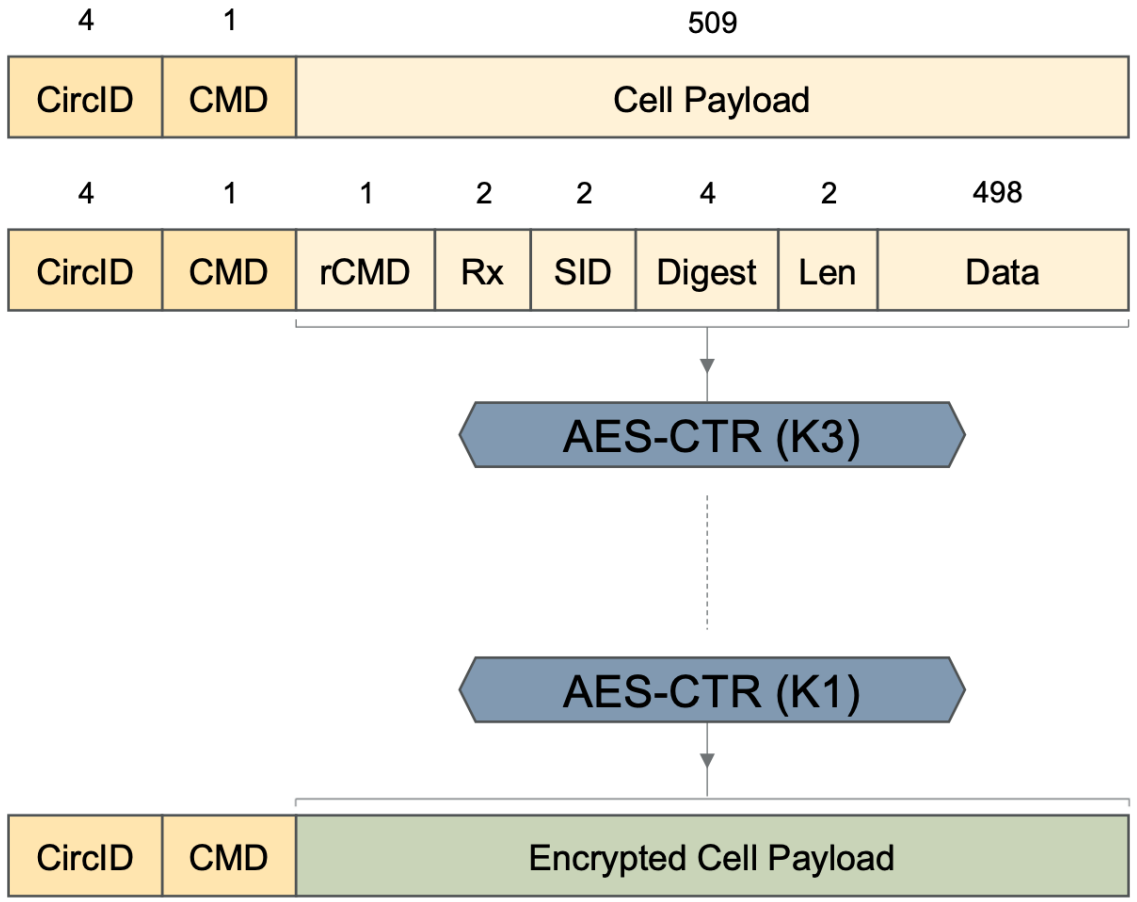


Roadmap

1. Motivation
2. Tor refresher
3. **What we have now, and why it's bad**
4. Cryptographic preliminaries
5. Counter Galois Onion
6. Conclusions

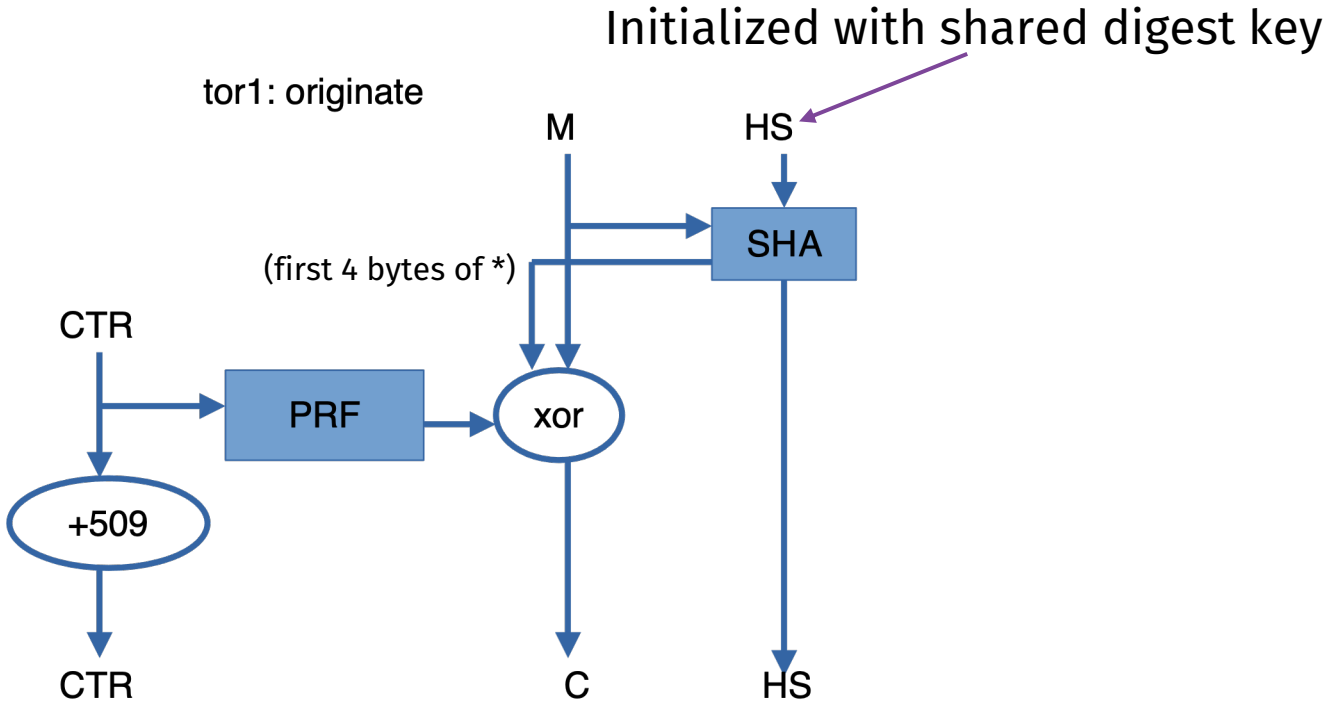
tor1

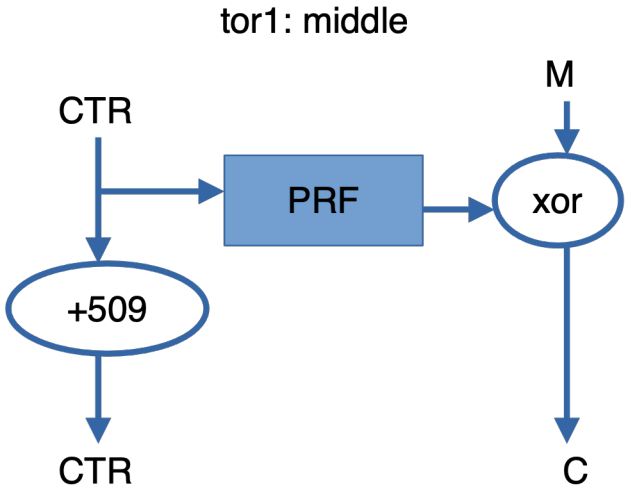
- Designed in 2002, never updated since
- Cryptographic digest
  - SHA-1
  - 4 bytes truncation
  - state maintained across cells
- “Recognized” (Rx) field of all zeros



[1]

tor1





- The first four bytes of hash are placed in “Digest” field
- PRF instantiated with AES-128

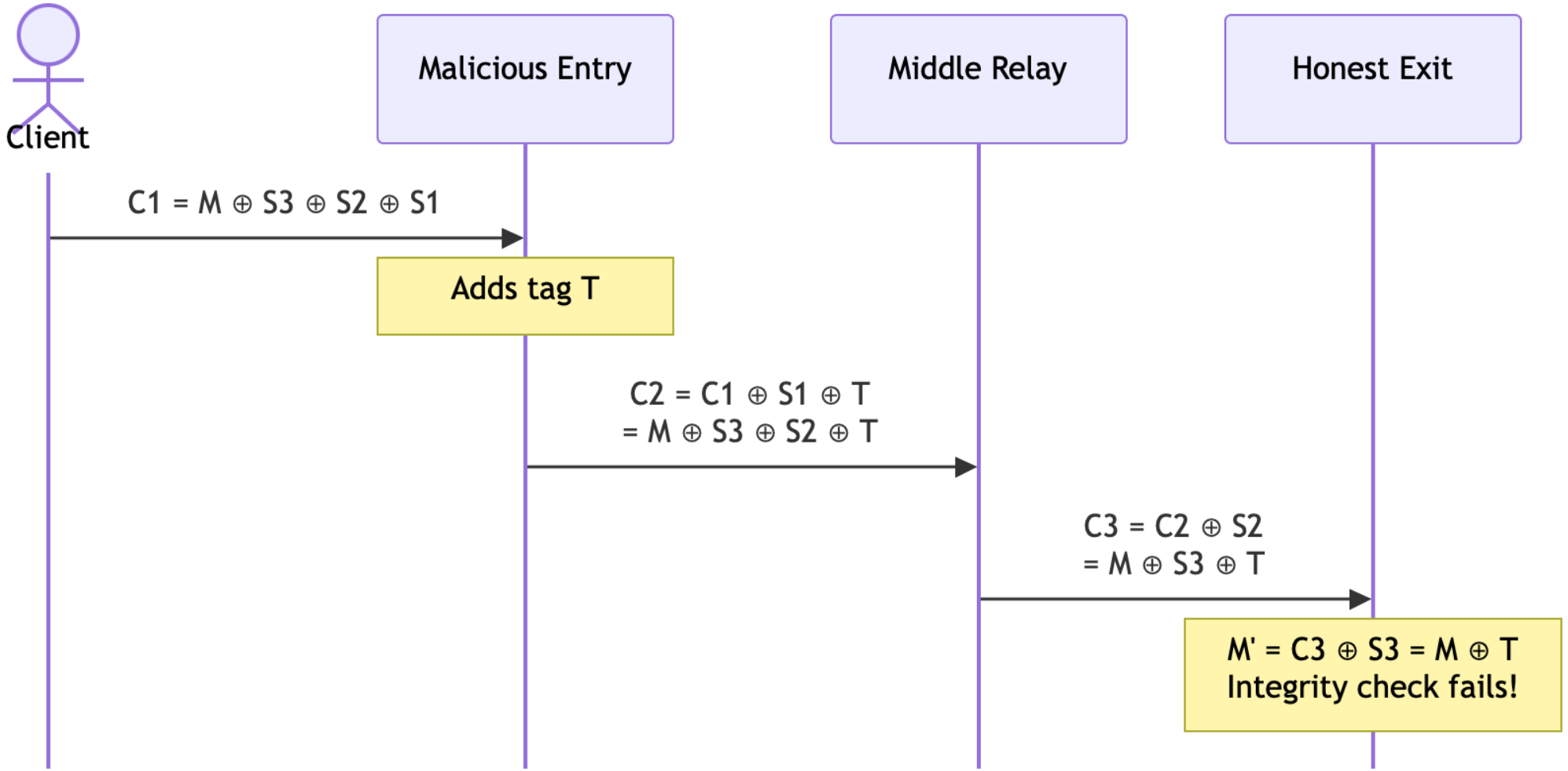
* = SHA-1(Shared_Digest_Key || Cell_1 || Cell_2 || Cell_3 ...)

Problems with tor1

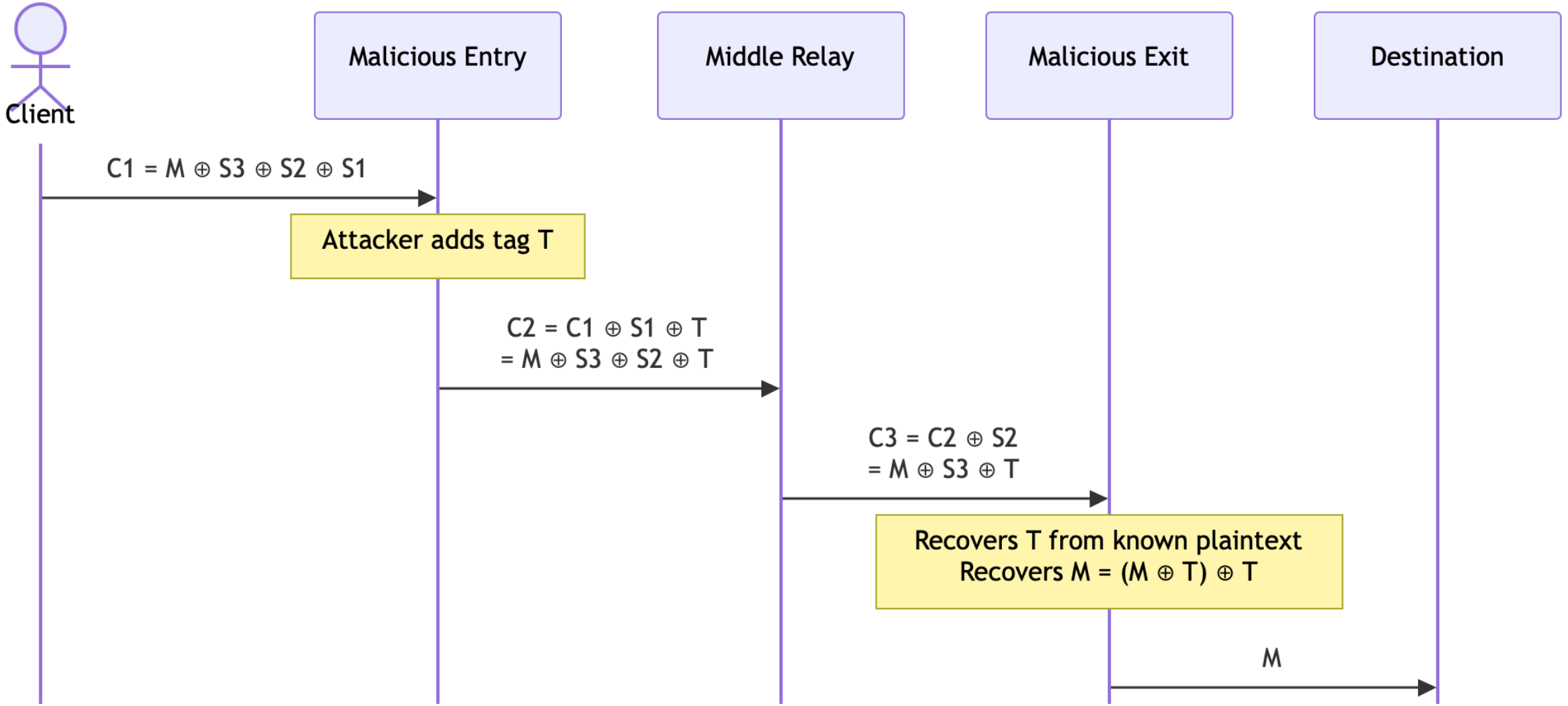
- MAC-then-encrypt paradigm
- Malleability
- No forward secrecy (during circuit lifetime)
- 4-byte authenticator
- **Tagging attacks**

“This is the most important attack we're solving with CGO. Even without the other problems, it would be worth fixing on its own.”

Tagging attacks



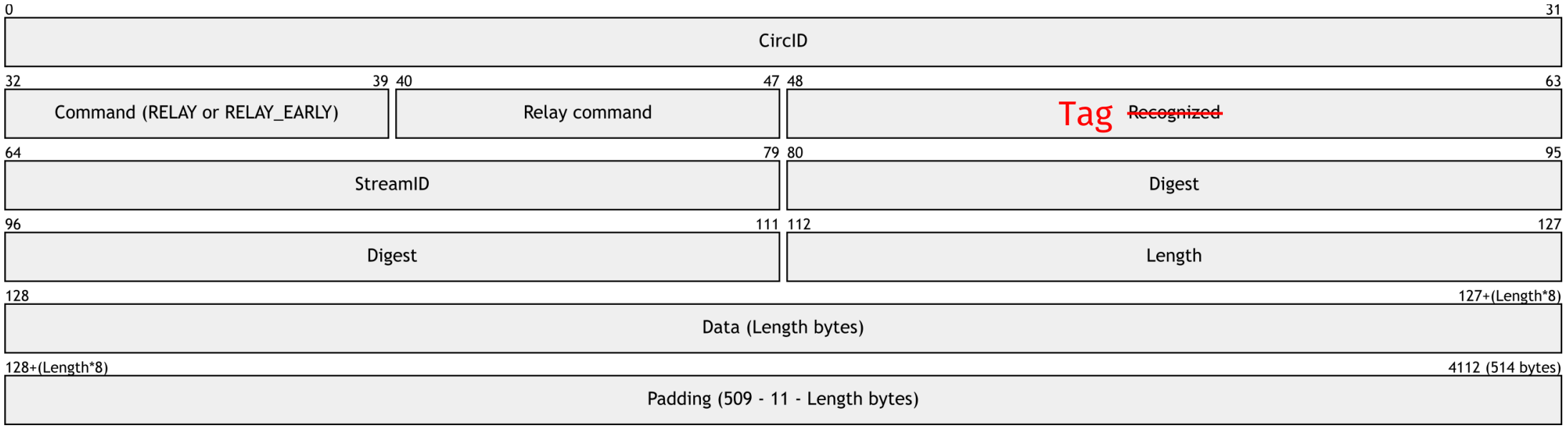
Tagging attacks



Tagging attacks

Exchange information out-of-band to de-anonymize client:

- Entry node knows relation (`Client IP, Tag T`)
- Exit node knows relation (`Destination IP, Tag T`)



Relative severity of tagging attacks

“Back when we did the Tor design, this didn't seem like a big deal”

- If an adversary controls entry and exit nodes it can correlate traffic by monitoring or manipulating the **timing** of traffic
- As Tor aims to be a *low-latency* system, it cannot prevent this
- So, tagging attacks can superficially look useless

Relative severity of tagging attacks

Tagging attacks “amplify” the resources of the attacker:

- **malicious entry → honest exit:** the decryption of the tagged cell will (likely) fail, and the circuit break
- **honest entry → malicious exit:** if the attacker receives an untagged cell, it can destroy the circuit

[tor-dev] Analysis of the Relative Severity of Tagging Attacks:

Assume an attacker has a 99.994% accurate **timing** filter. Due to a very low base rate, 1501 "false positive" circuits for every controlled circuit.

A cryptographic tagging attack compromises 1501× as many circuits with the same attack capacity!

Requirements (non-exhaustive list)

- Resistance to tagging attacks
- Support for circuits of arbitrary length
- Support for leaky pipes
- Constant ciphertext size (cell length is fixed)
- Forward secrecy

Roadmap

1. Motivation
2. Tor refresher
3. What we have now, and why it's bad
4. **Cryptographic preliminaries**
   - **CGO's building block**
   - Syntax
5. Counter Galois Onion
6. Conclusions

Preliminaries – building block

CGO's underlying primitive is an

updatable tweakable split-domain cipher

Preliminaries – building block

CGO's underlying primitive is an

updatable tweakable split-domain cipher

A *tweakable cipher* is an algorithm

$$\widetilde{\mathrm{E}}\widetilde{\mathrm{E}}\colon \mathcal{K}\times\mathcal{H}\times\mathcal{D}\to\mathcal{D}$$

such that for any *key* $K\in\mathcal{K}$, and *tweak* $H\in\mathcal{H}$, $\widetilde{\mathrm{E}}\widetilde{\mathrm{E}}(K,H,\cdot)$ identifies a permutation over $\mathcal{D}$.

Security notion: *tweakable* (S)PRP. Same as (S)PRP, but adversary tries to distinguish $\widetilde{\mathrm{E}}\widetilde{\mathrm{E}}_K(\cdot,\cdot)$ from a "tweakable random permutation" $\widetilde{\Pi}(\cdot,\cdot)$.

Preliminaries – building block

CGO's underlying primitive is an

updatable tweakable split-domain cipher

A *tweakable split-domain cipher* is an algorithm

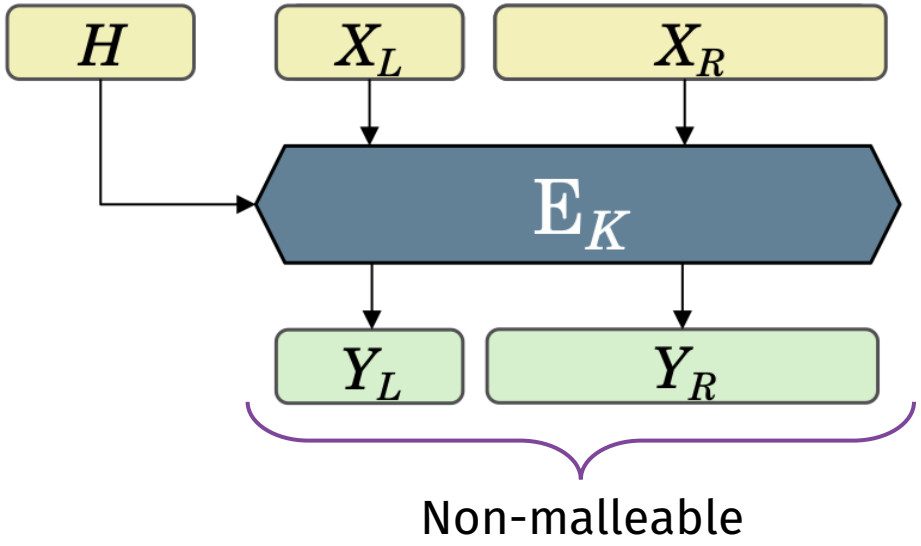
$$\widetilde{\mathrm{EE}}\colon \mathcal{K}\times\mathcal{H}\times\mathcal{D}_{\mathrm{L}}\times\mathcal{D}_{\mathrm{R}} \to \mathcal{D}_{\mathrm{L}}\times\mathcal{D}_{\mathrm{R}}$$

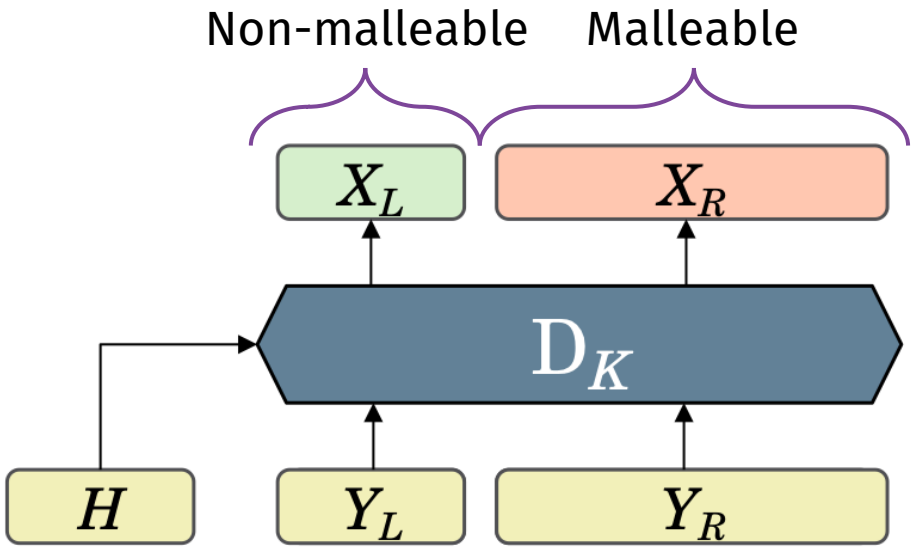
such that for any *key* $K \in \mathcal{K}$, and *tweak* $H \in \mathcal{H}$, $\widetilde{\mathrm{EE}}(K, H, \cdot, \cdot)$ identifies a permutation over $\mathcal{D} = \mathcal{D}_{\mathrm{L}}\times\mathcal{D}_{\mathrm{R}}$.

Security notion: *rugged* PRP. Sits between PRP and SPRP, in that the adversary is given *partial* access to decryption.

Intuition on RPRP

Loosely speaking, the enciphering algorithm is fully non-malleable, whereas the deciphering algorithm only offers a restricted form of non-malleability





Preliminaries – building block

CGO's underlying primitive is an

updatable tweakable split-domain cipher

An *updatable tweakable split-domain cipher* augments

$$\widetilde{\mathrm{EE}}\colon \mathcal{K}\times\mathcal{H}\times\mathcal{D}_{\mathrm{L}}\times\mathcal{D}_{\mathrm{R}} \to \mathcal{D}_{\mathrm{L}}\times\mathcal{D}_{\mathrm{R}}$$

with an update algorithm

$$\widetilde{\mathrm{EU}}\colon \mathcal{K}\times\mathcal{D}_{\mathrm{L}} \to \mathcal{K}\times\mathcal{D}_{\mathrm{L}}$$

which takes a *key* $K \in \mathcal{K}$, and left element $\mathrm{X}_{\mathrm{L}} \in \mathcal{D}_{\mathrm{L}}$, and returns new values for each.

Security notion: URRND / CRND_{ℓ}

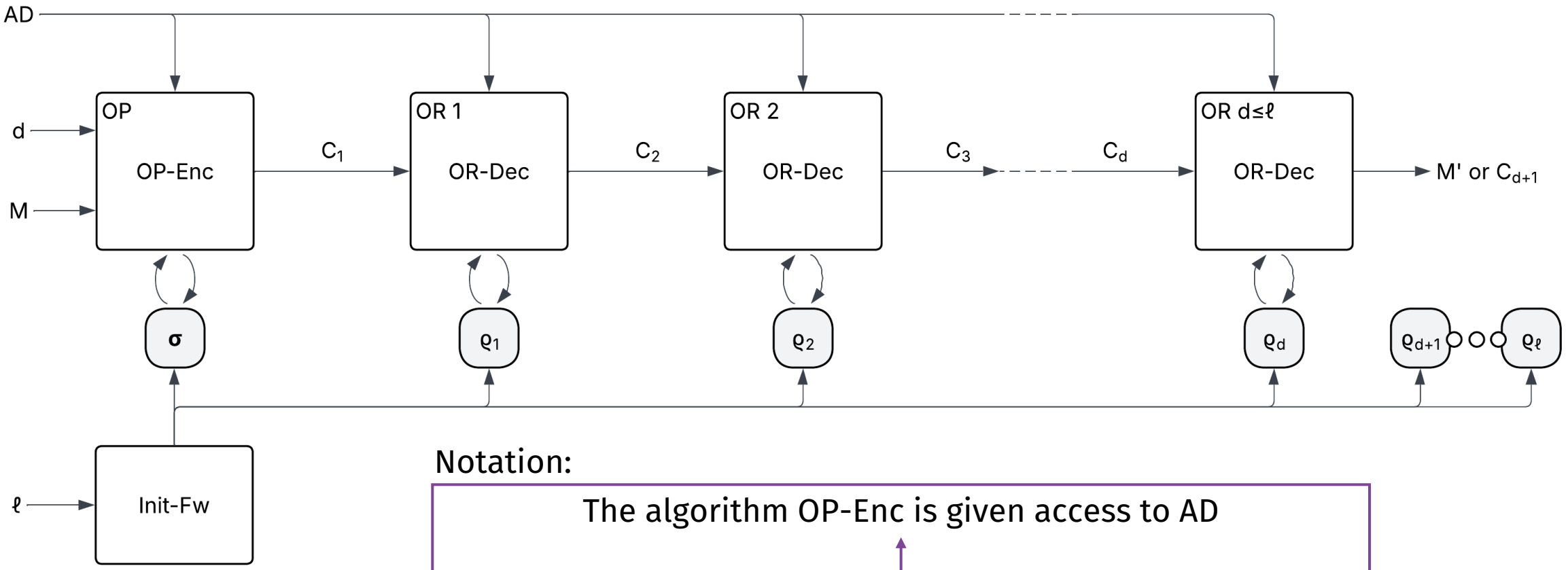
Roadmap

1. Motivation
2. Tor refresher
3. What we have now, and why it's bad
4. **Cryptographic preliminaries**
   - CGO's building block
   - **Syntax**
5. Counter Galois Onion
6. Conclusions

Onion Encryption

- Syntactically what we require from CGO is “Onion Encryption”
- Assume a circuit of length ℓ
  - OP “index 0”
  - OR indices $1, \ldots, \ell$
- Let $\mathcal{M} = \{0,1\}^m, \mathcal{C} = \{0,1\}^c$, we assume $m < c$ (disjoint sets)
  - in Tor $c = 8 \cdot 509$, and $m = 8 \cdot (509 - 16)$
- Model associate data $\mathcal{AD} \subseteq \{0,1\}^*$
- Assume the circuit setup phase (key exchange) is over
  - all parties’ states initialized simultaneously
  - Init-XX algorithms

Onion Encryption – forward direction



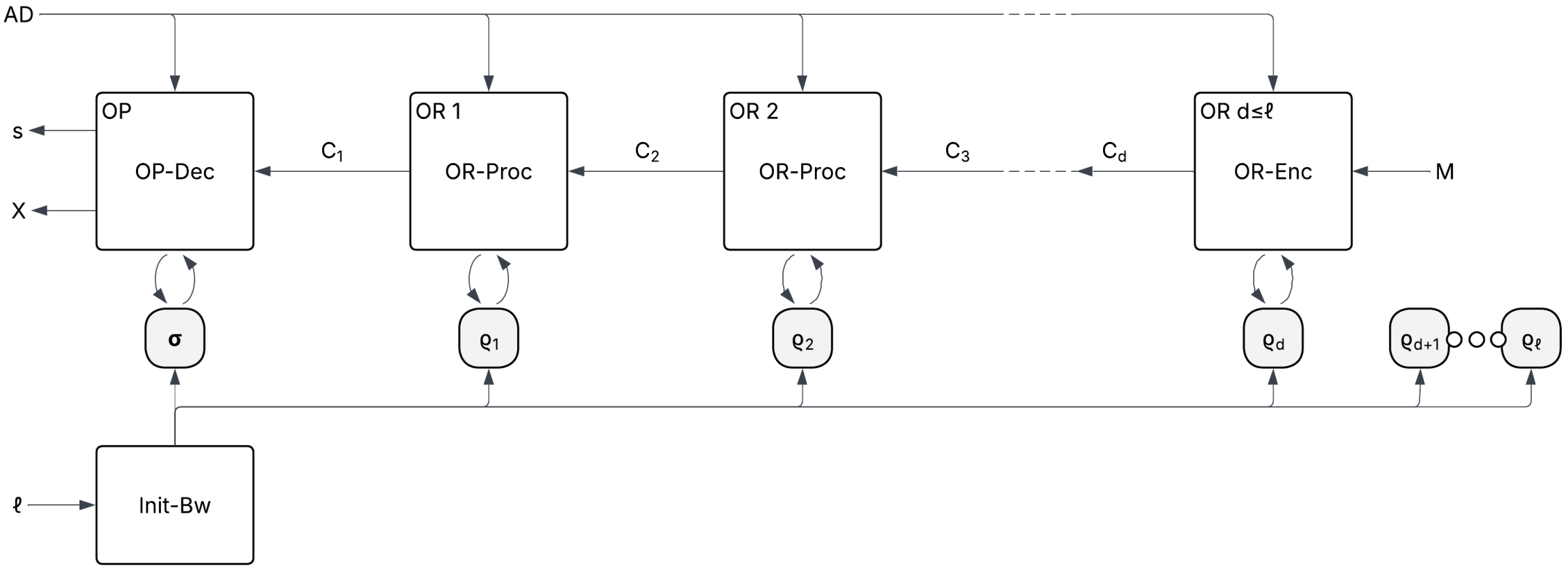
Notation:

The algorithm OP-Enc is given access to AD

$$\text{OP-Enc}^{AD}_{\langle \sigma \rangle}(d, M)$$

As a side-effect the algorithm OP-Enc may update its state σ

Onion Encryption – backwards direction



Roadmap

1. Motivation
2. Tor refresher
3. What we have now, and why it’s bad
4. Cryptographic preliminaries
5. **Counter Galois Onion**
6. Conclusions

CGO - initialization

State for the i-th hop:

maintained by OP

maintained by OR i

Init-Bw(ℓ)

for $i = 1$ **to** ℓ

$K \leftarrow\$ \{0,1\}^k$

$N \leftarrow\$ \{0,1\}^n$

$T' \leftarrow 0^n$

$\sigma_i \leftarrow (K, N, T')$

$\varrho_i \leftarrow (K, N, T')$

$\boldsymbol{\sigma} \leftarrow (\sigma_1, \ldots, \sigma_\ell)$

return $(\boldsymbol{\sigma}, \varrho_1, \ldots, \varrho_\ell)$

“Decryption state”

“Encryption/processing state”

Init-Fw(ℓ)

for $i = 1$ **to** ℓ

$K \leftarrow\$ \{0,1\}^k$

$N \leftarrow\$ \{0,1\}^n$

$T' \leftarrow 0^n$

$\sigma_i \leftarrow (K, N, T')$

$\varrho_i \leftarrow (K, N, T')$

$\boldsymbol{\sigma} \leftarrow (\sigma_1, \ldots, \sigma_\ell)$

return $(\boldsymbol{\sigma}, \varrho_1, \ldots, \varrho_\ell)$

“Encryption state”

“Decryption state”

CGO - initialization

Shared symmetric key

Init-Bw(ℓ) / Init-Fw(ℓ)

for $i = 1$ **to** ℓ

$K \leftarrow\$ \{0,1\}^k$

$N \leftarrow\$ \{0,1\}^n$

$T' \leftarrow 0^n$

$\sigma_i \leftarrow (K, N, T')$

$\varrho_i \leftarrow (K, N, T')$

$\boldsymbol{\sigma} \leftarrow (\sigma_1, \ldots, \sigma_\ell)$

return $(\boldsymbol{\sigma}, \varrho_1, \ldots, \varrho_\ell)$

CGO - initialization

Nonce, only updated when OR i is the source / destination

Init-Bw(ℓ) / Init-Fw(ℓ)

for $i = 1$ **to** ℓ

$K \leftarrow\$ \{0,1\}^k$

$N \leftarrow\$ \{0,1\}^n$

$T' \leftarrow 0^n$

$\sigma_i \leftarrow (K, N, T')$

$\varrho_i \leftarrow (K, N, T')$

$\boldsymbol{\sigma} \leftarrow (\sigma_1, \ldots, \sigma_\ell)$

return $(\boldsymbol{\sigma}, \varrho_1, \ldots, \varrho_\ell)$

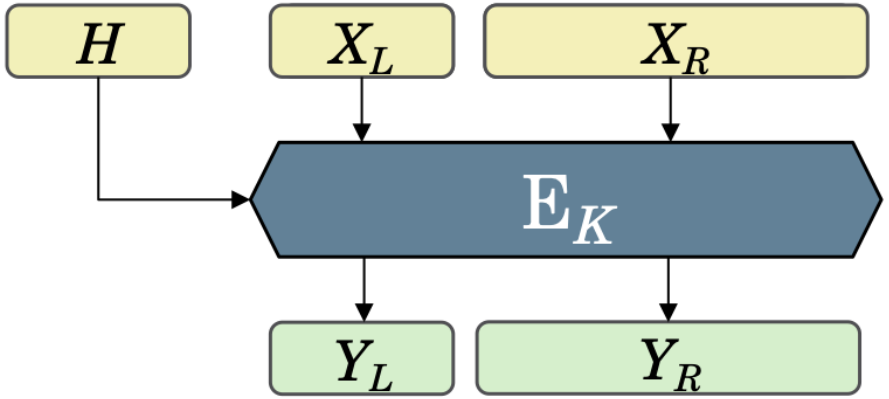
Chaining string, updated for each OR j, with $1 \leq j \leq i$

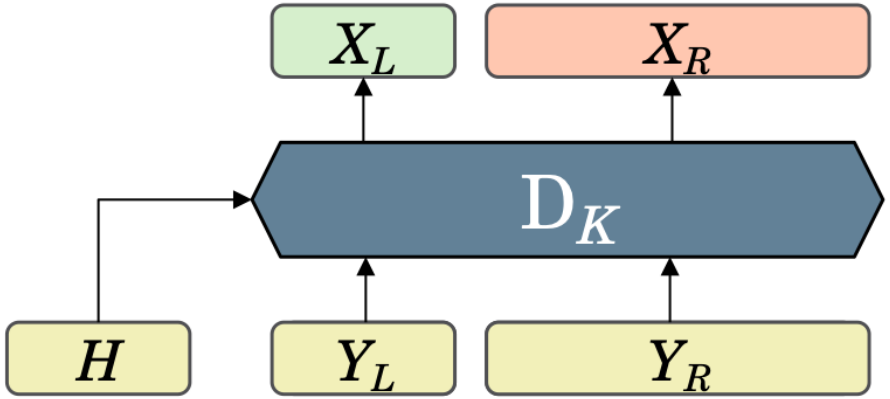
Some notes on syntax

CGO is built upon an updatable tweakable split-domain cipher :

- Encipher → $\mathrm{EE}_{\mathrm{K}}^{\mathrm{H}}(\mathrm{X_L}, \mathrm{X_R}) = (\mathrm{Y_L}, \mathrm{Y_R})$
- Decipher → $\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}}(\mathrm{Y_L}, \mathrm{Y_R}) = (\mathrm{X_L}, \mathrm{X_R})$
- Update → $\mathrm{EU}_{\mathrm{K}}(\mathrm{X_L}) = (\mathrm{K'}, \mathrm{X'_L})$

Correctness:
$\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}}(\mathrm{EE}_{\mathrm{K}}^{\mathrm{H}}(\mathrm{X_L}, \mathrm{X_R})) = (\mathrm{X_L}, \mathrm{X_R})$



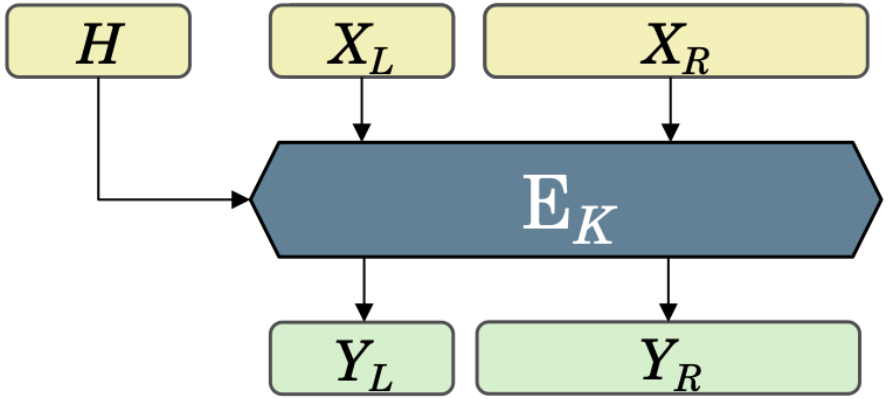


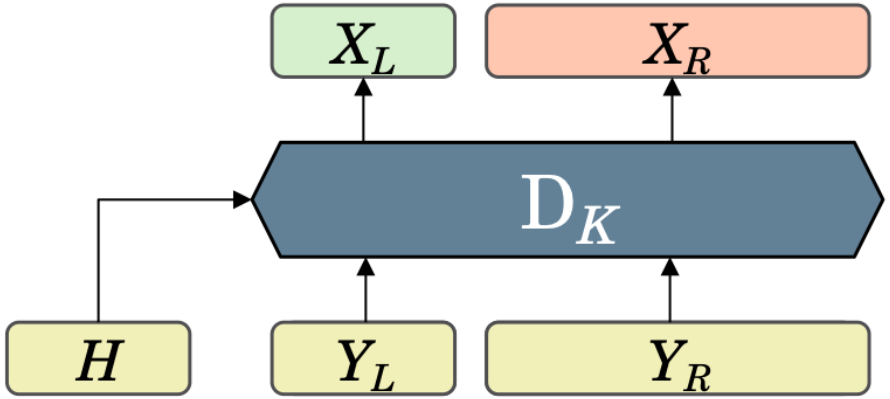
Some notes on syntax

As $\mathrm{EE}_{\mathrm{K}}^{\mathrm{H}}$ and $\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}}$ are permutations subject to $\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}} = (\mathrm{EE}_{\mathrm{K}}^{\mathrm{H}})^{-1}$:

$$\mathrm{EE}_{\mathrm{K}}^{\mathrm{H}} = (\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}})^{-1} \text{ and } \mathrm{EE}_{\mathrm{K}}^{\mathrm{H}}\left(\mathrm{ED}_{\mathrm{K}}^{\mathrm{H}}(\mathrm{Y_L}, \mathrm{Y_R})\right) = (\mathrm{Y_L}, \mathrm{Y_R})$$

i.e., we can use the encipher algorithm to decrypt!



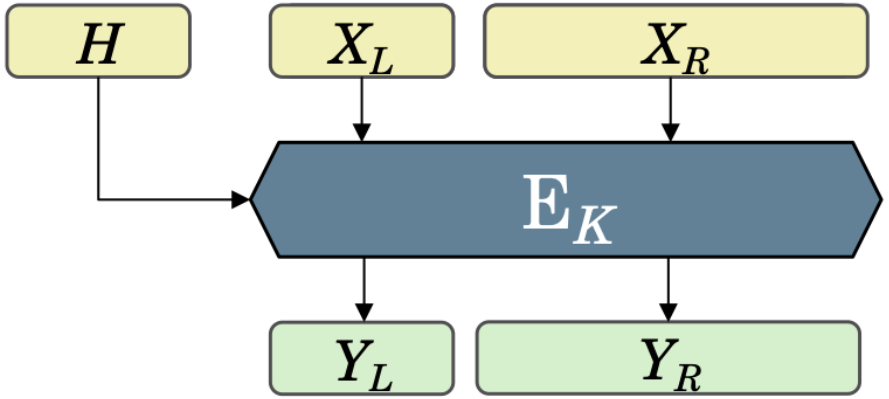


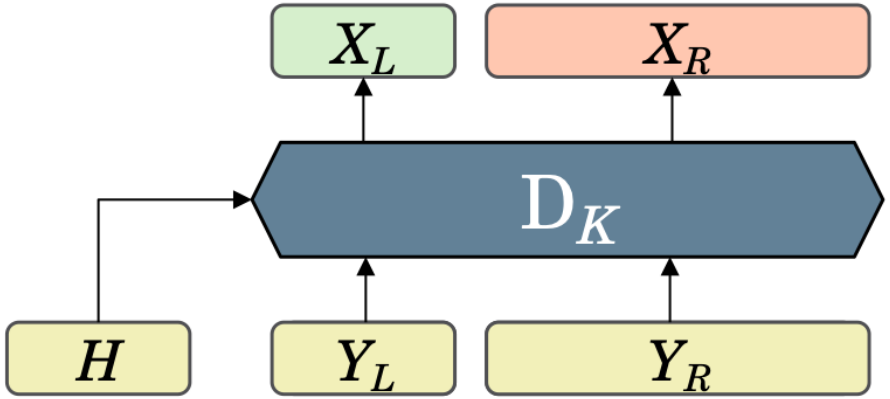
“Non-malleability only where it’s needed”

In CGO, the in cipher is only required to be a **rugged PRP**.

Informally: Enciphering is **pseudorandom**, whereas Deciphering only provides **partial non-malleability**.

Only the processing at the ORs needs to be non-malleable, whereas that at the OP does not.





CGO – forward direction

$\mathsf{OP\text{-}Enc}^{AD}_{\langle\sigma\rangle}(d, M)$

if $(d > \ell)$ **return** $\perp$

for $i = d$ **to** 1

$(T, \overline{C}) \leftarrow \mathsf{ED}^{H}_{K}(T, \overline{C})$

return $T \parallel \overline{C}$

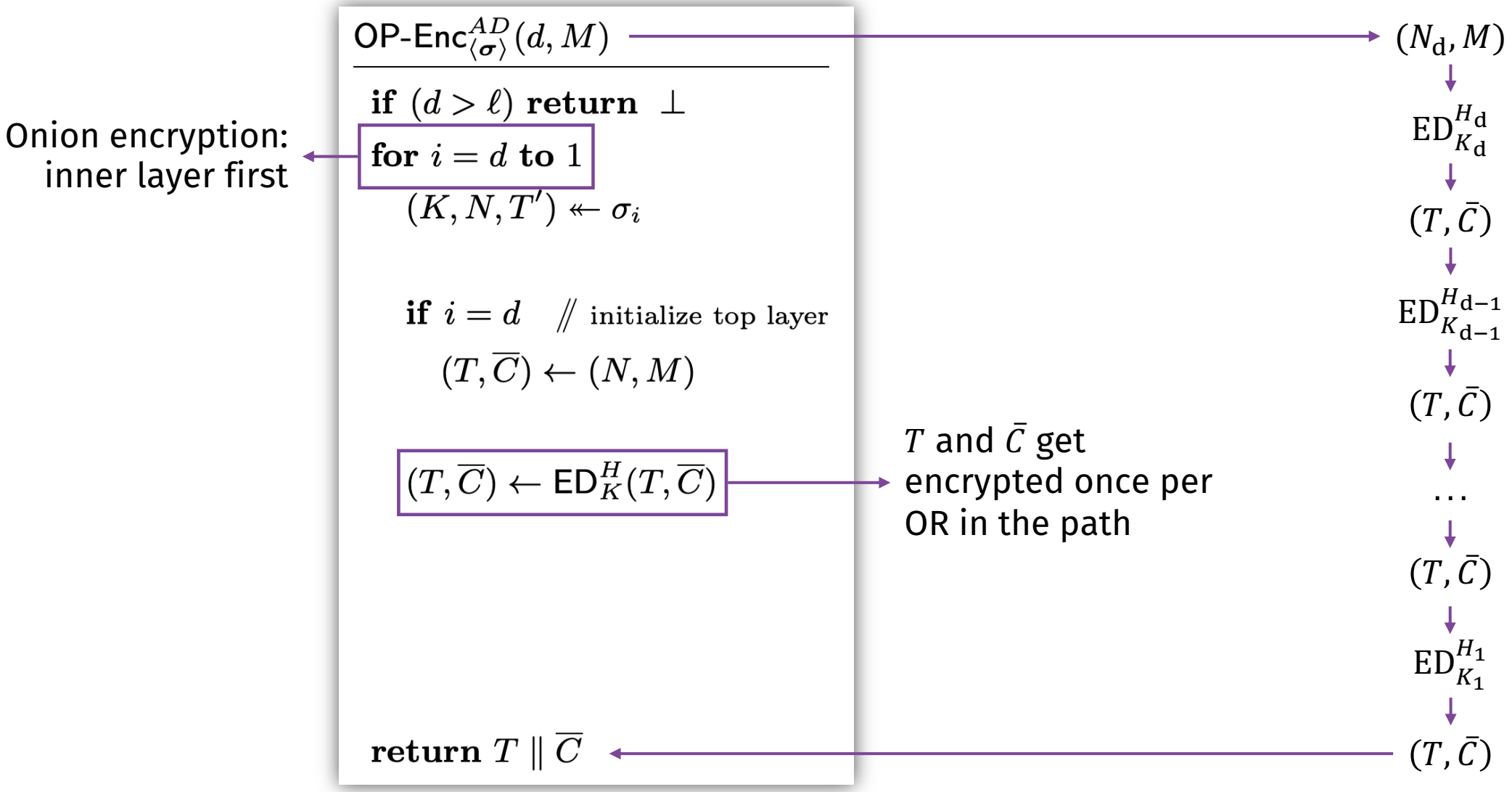
$\mathsf{OR\text{-}Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(T, \overline{C}) \leftarrow \mathsf{EE}^{H}_{K}(T, \overline{C})$

Only OP decrypts

ORs only ever use the cipher to encrypt

CGO – forward direction



CGO – forward direction

At OR i $\neq$ d:

$\mathsf{OR\text{-}Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$

$(T, \overline{C}) \leftarrow \mathsf{EE}^{H}_{K}(T, \overline{C})$

if $T = N$

else

// forwarding ciphertext

$X \leftarrow T \parallel \overline{C}$

$\varrho \leftarrow (K, N, T)$

return X

$(T, \bar{C})$

$\mathrm{EE}^{H_1}_{K_1}$

$(T, \bar{C})$

$\mathrm{EE}^{H_2}_{K_2}$

$(T, \bar{C})$

...

$(T, \bar{C})$

$\mathrm{EE}^{H_\mathrm{d}}_{K_\mathrm{d}}$

(N_d, M)

CGO – forward direction

At OR i = d:

$\text{OR-Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$

$(T, \overline{C}) \leftarrow \text{EE}^{H}_{K}(T, \overline{C})$

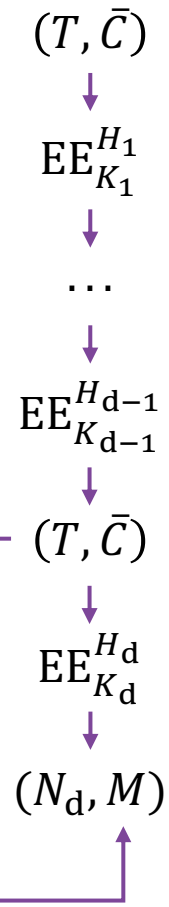
if $T = N$

// ciphertext recognised

$X \leftarrow \overline{C}$

$\varrho \leftarrow (K, N, T)$

return X



CGO – forward direction

$\mathsf{OR\text{-}Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$

$(T, \overline{C}) \leftarrow \mathsf{EE}^{H}_{K}(T, \overline{C})$

if $T = N$

// ciphertext recognised

$X \leftarrow \overline{C}$

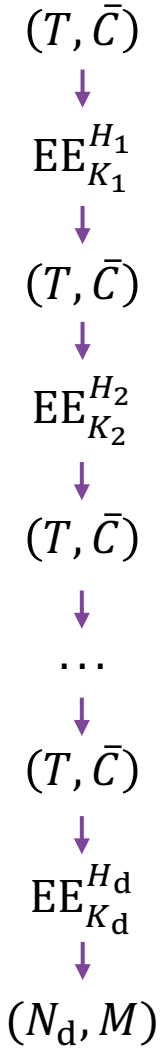
else

// forwarding ciphertext

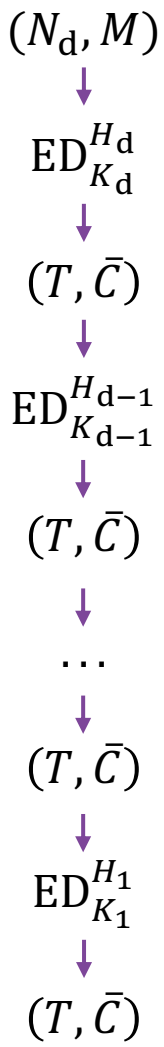
$X \leftarrow T \parallel \overline{C}$

$\varrho \leftarrow (K, N, T)$

return X



CGO – forward direction



$\mathsf{OP\text{-}Enc}_{\langle\sigma\rangle}^{AD}(d, M)$

if $(d > \ell)$ **return** $\perp$
for $i = d$ **to** 1
  $(K, N, T') \twoheadleftarrow \sigma_i$

  if $i = d$ // initialize top layer
    $(T, \overline{C}) \leftarrow (N, M)$

  $(T, \overline{C}) \leftarrow \mathsf{ED}_K^H(T, \overline{C})$

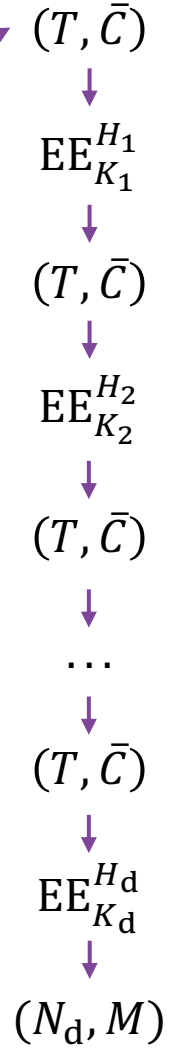
return $T \parallel \overline{C}$

$\mathsf{OR\text{-}Dec}_{\langle\varrho\rangle}^{AD}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$

$(T, \overline{C}) \leftarrow \mathsf{EE}_K^H(T, \overline{C})$
if $T = N$
  // ciphertext recognised

  $X \leftarrow \overline{C}$
else
  // forwarding ciphertext
  $X \leftarrow T \parallel \overline{C}$
$\varrho \leftarrow (K, N, T)$
return X



CGO – forward direction

$\mathsf{OP\text{-}Enc}^{AD}_{\langle\boldsymbol{\sigma}\rangle}(d, M)$

if $(d > \ell)$ **return** $\perp$
for $i = d$ **to** 1
  $(K, N, T') \twoheadleftarrow \sigma_i$

  if $i = d$ // initialize top layer
    $(T, \overline{C}) \leftarrow (N, M)$

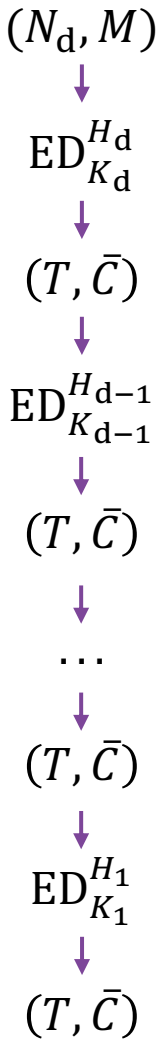
  $(T, \overline{C}) \leftarrow \mathsf{ED}^{H}_{K}(T, \overline{C})$
  if $i = d$
    // update destination's key
    $(K, N) \leftarrow \mathsf{EU}_{K}(N)$

return $T \parallel \overline{C}$

$\mathsf{OR\text{-}Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$
$H \leftarrow T' \parallel AD$
$(T, \overline{C}) \leftarrow \mathsf{EE}^{H}_{K}(T, \overline{C})$
if $T = N$
  // ciphertext recognised
  $(K, N) \leftarrow \mathsf{EU}_{K}(N)$
  $X \leftarrow \overline{C}$
else
  // forwarding ciphertext
  $X \leftarrow T \parallel \overline{C}$
$\varrho \leftarrow (K, N, T)$
return X

CGO – forward direction

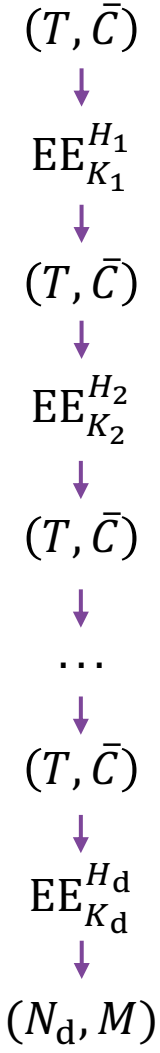


OP-Enc$^{AD}_{\langle\sigma\rangle}(d, M)$

if $(d > \ell)$ **return** $\perp$
for $i = d$ **to** 1
  $(K, N, T') \twoheadleftarrow \sigma_i$
  $H \leftarrow T' \parallel AD$
  if $i = d$ // initialize top layer
    $(T, \overline{C}) \leftarrow (N, M)$
  $(T, \overline{C}) \leftarrow \mathrm{ED}_K^H(T, \overline{C})$
return $T \parallel \overline{C}$

OR-Dec$^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$
$H \leftarrow T' \parallel AD$
$(T, \overline{C}) \leftarrow \mathrm{EE}_K^H(T, \overline{C})$
if $T = N$
  // ciphertext recognised
  $X \leftarrow \overline{C}$
else
  // forwarding ciphertext
  $X \leftarrow T \parallel \overline{C}$
$\varrho \leftarrow (K, N, T)$
return X



CGO – forward direction

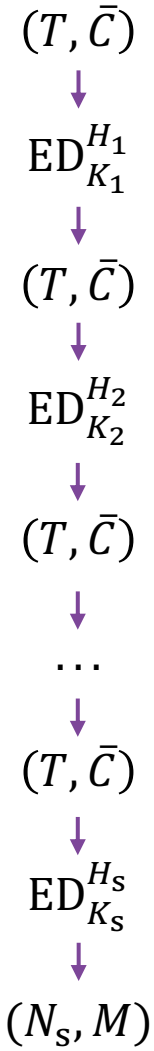
$\mathsf{OP\text{-}Enc}^{AD}_{\langle\boldsymbol{\sigma}\rangle}(d, M)$

if $(d > \ell)$ **return** $\perp$
for $i = d$ **to** 1
  $(K, N, T') \twoheadleftarrow \sigma_i$
  $H \leftarrow T' \parallel AD$
  if $i = d$ // initialize top layer
    $(T, \overline{C}) \leftarrow (N, M)$
  $T' \leftarrow T$
  $(T, \overline{C}) \leftarrow \mathsf{ED}^H_K(T, \overline{C})$
  if $i = d$
    // update destination's key
    $(K, N) \leftarrow \mathsf{EU}_K(N)$
  $\sigma_i \leftarrow (K, N, T')$
return $T \parallel \overline{C}$

$\mathsf{OR\text{-}Dec}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$
$H \leftarrow T' \parallel AD$
$(T, \overline{C}) \leftarrow \mathsf{EE}^H_K(T, \overline{C})$
if $T = N$
  // ciphertext recognised
  $(K, N) \leftarrow \mathsf{EU}_K(N)$
  $X \leftarrow \overline{C}$
else
  // forwarding ciphertext
  $X \leftarrow T \parallel \overline{C}$
$\varrho \leftarrow (K, N, T)$
return X

CGO – backwards direction



$\mathsf{OP\text{-}Dec}^{AD}_{\langle\sigma\rangle}(T \parallel \overline{C})$

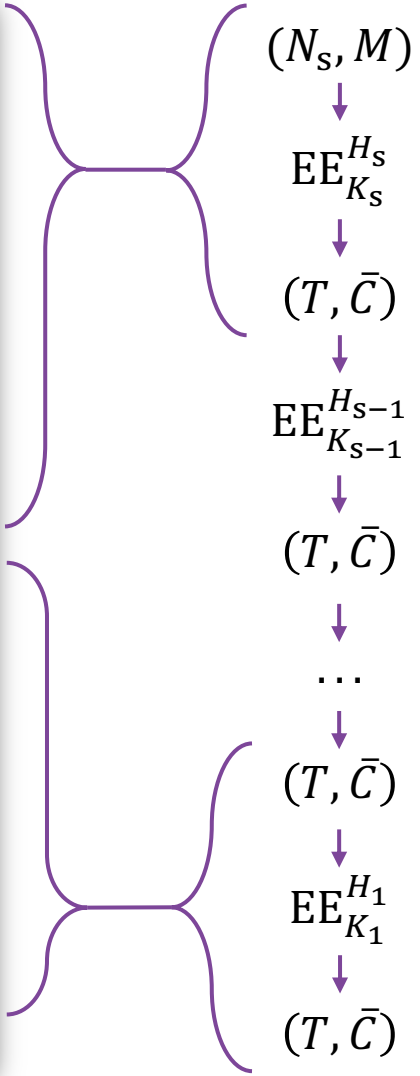
$i \leftarrow 1,\ s \leftarrow 0$
while $i \leq \ell \ \wedge \ s = 0$
  $(K, N, T') \twoheadleftarrow \sigma_i$
  $H \leftarrow T' \parallel AD$
  $(T', \overline{C}) \leftarrow \mathsf{ED}_K^H(T, \overline{C})$
  if $T' = N$
    // ciphertext recognised
    $(K, N) \leftarrow \mathsf{EU}_K(T)$
    $s \leftarrow i; M \leftarrow \overline{C}$
  $\sigma_i \leftarrow (K, N, T)$
  $i \leftarrow i + 1,\ T \leftarrow T'$
if $s = 0$
  $M \leftarrow \perp$
return (M, s)

$\mathsf{OR\text{-}Enc}^{AD}_{\langle\varrho\rangle}(M)$

$(K, N, T') \twoheadleftarrow \varrho$
$H \leftarrow T' \parallel AD$
$(T, \overline{C}) \leftarrow \mathsf{EE}_K^H(N, M)$
$(K, N) \leftarrow \mathsf{EU}_K(T)$
$\varrho \leftarrow (K, N, T)$
return $T \parallel \overline{C}$

$\mathsf{OR\text{-}Proc}^{AD}_{\langle\varrho\rangle}(T \parallel \overline{C})$

$(K, N, T') \twoheadleftarrow \varrho$
$H \leftarrow T' \parallel AD$
$(T, \overline{C}) \leftarrow \mathsf{EE}_K^H(T \parallel \overline{C})$
$\varrho \leftarrow (K, N, T)$
return $T \parallel \overline{C}$

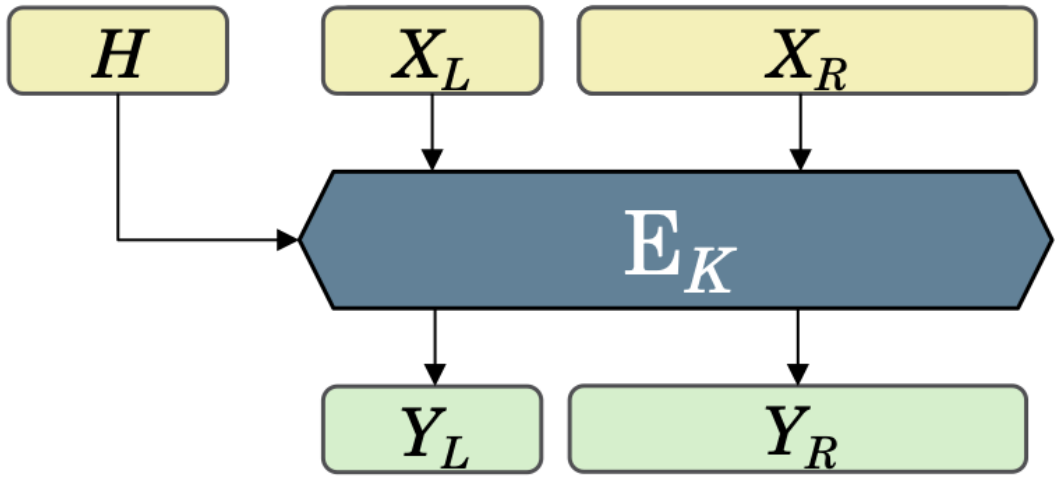


Requirements (non-exhaustive list)

- Resistance to tagging attacks?
- Support for circuits of arbitrary length?
- Support for leaky pipes?
- Constant ciphertext size?
- Forward secrecy?

Requirements (non-exhaustive list)

- Resistance to tagging attacks?



1. Malicious entry node flips a bit in the ciphertext
2. Honest middle node receives modified ciphertext $T'||\bar{C}'$
3. EE is a PRP, when the middle node enciphers $T'||\bar{C}'$ the output will not be predictable
4. At the exit node, the ciphertext is randomized and tampering cannot be undone
5. States get desynchronized and circuit tore down

Requirements (non-exhaustive list)

- Resistance to tagging attacks? ✓
- Support for circuits of arbitrary length? ✓
  - algorithms defined for any ℓ
- Support for leaky pipes?
- Constant ciphertext size?
- Forward secrecy?

Requirements (non-exhaustive list)

- Resistance to tagging attacks? ✓
- Support for circuits of arbitrary length? ✓
- Support for leaky pipes? ✓
  - any node can recognize its nonce and output a message early
- Constant ciphertext size?
- Forward secrecy?

Requirements (non-exhaustive list)

- Resistance to tagging attacks? ✓
- Support for circuits of arbitrary length? ✓
- Support for leaky pipes? ✓
- Constant ciphertext size? ✓
- Forward secrecy?

Requirements (non-exhaustive list)

- Resistance to tagging attacks? ✓
- Support for circuits of arbitrary length? ✓
- Support for leaky pipes? ✓
- Constant ciphertext size?
- Forward secrecy?
  - keys are updated after being used

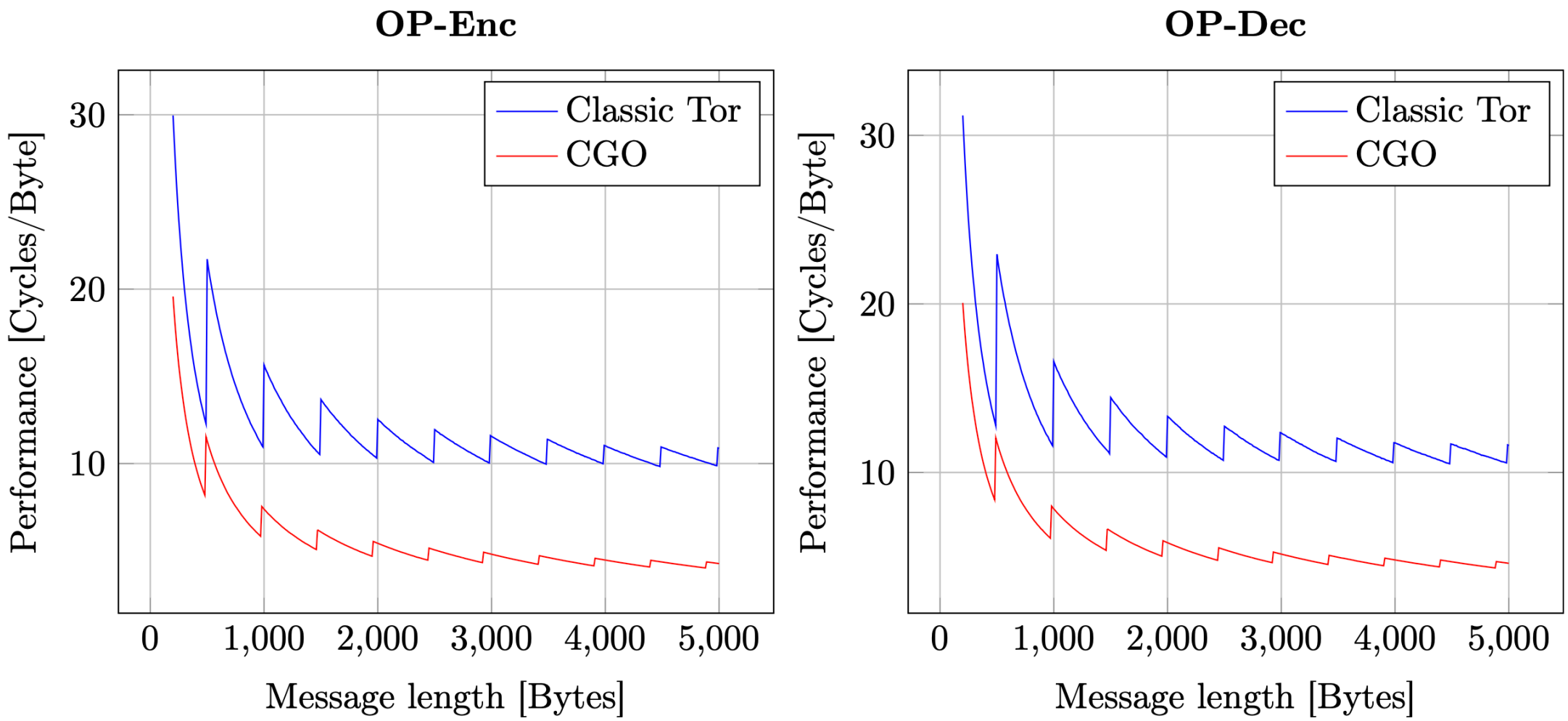
$(K, N) \leftarrow \mathsf{EU}_K(N)$

URRND / CRND_ℓ $\Longrightarrow$ this is pseudo random

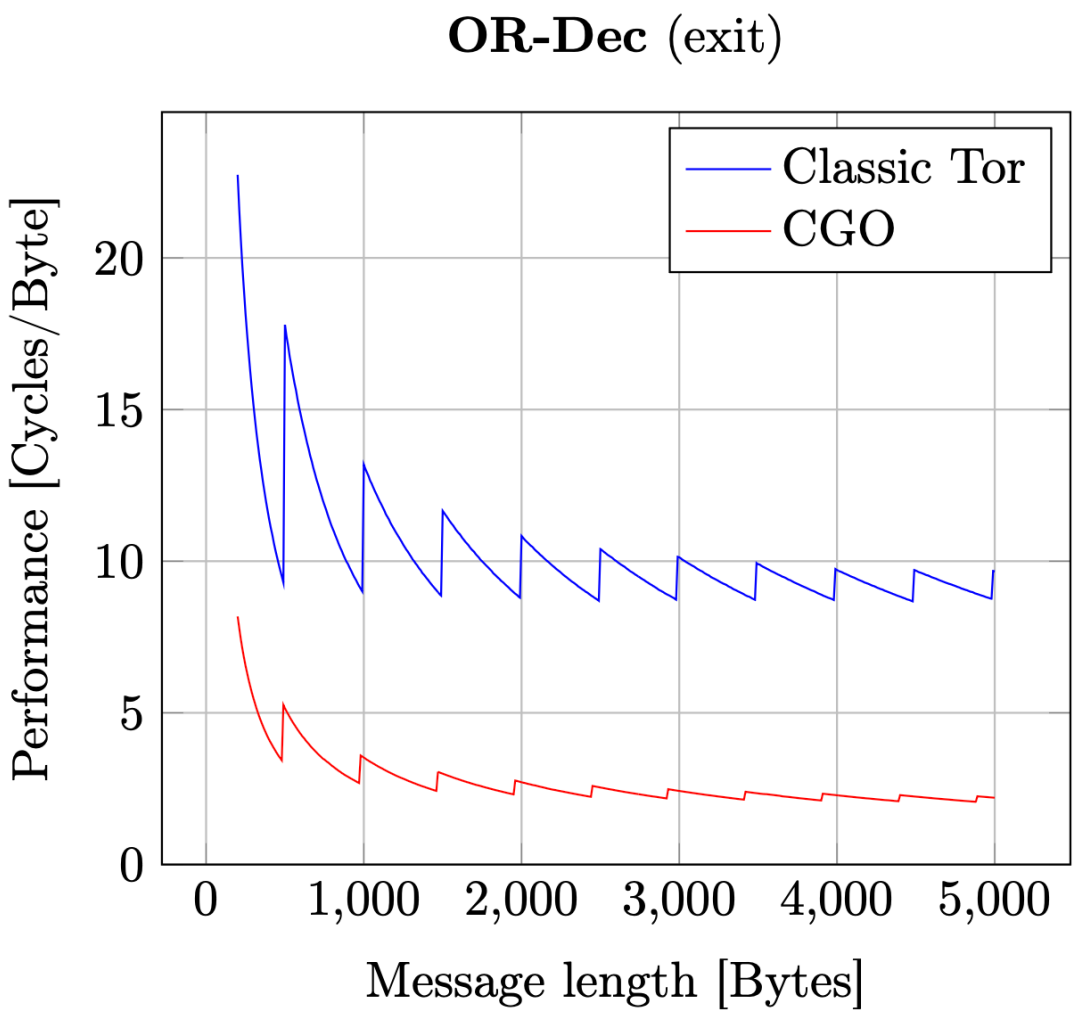
Requirements (non-exhaustive list)

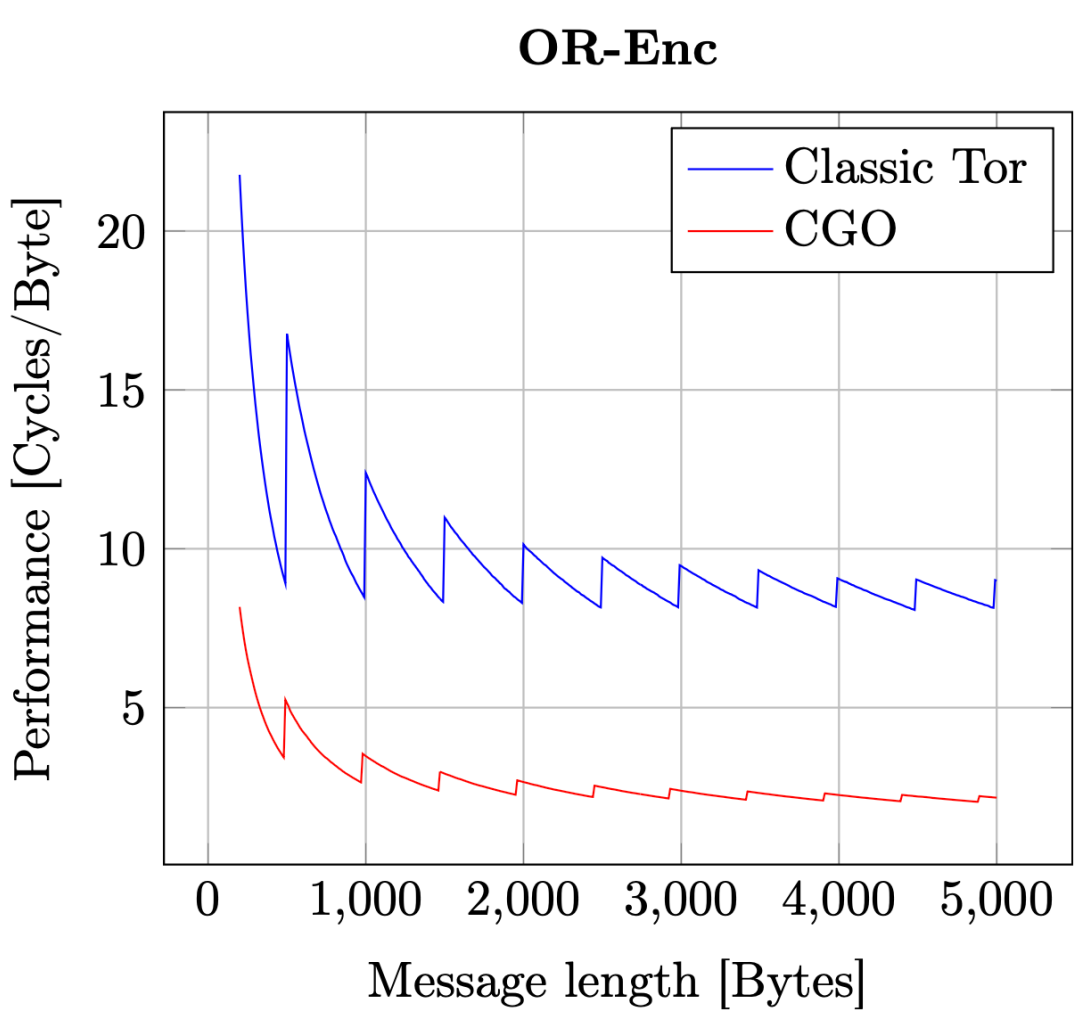
- Resistance to tagging attacks? ✓
- Support for circuits of arbitrary length? ✓
- Support for leaky pipes? ✓
- Constant ciphertext size?
- Forward secrecy?
- Missing anything?

Performance

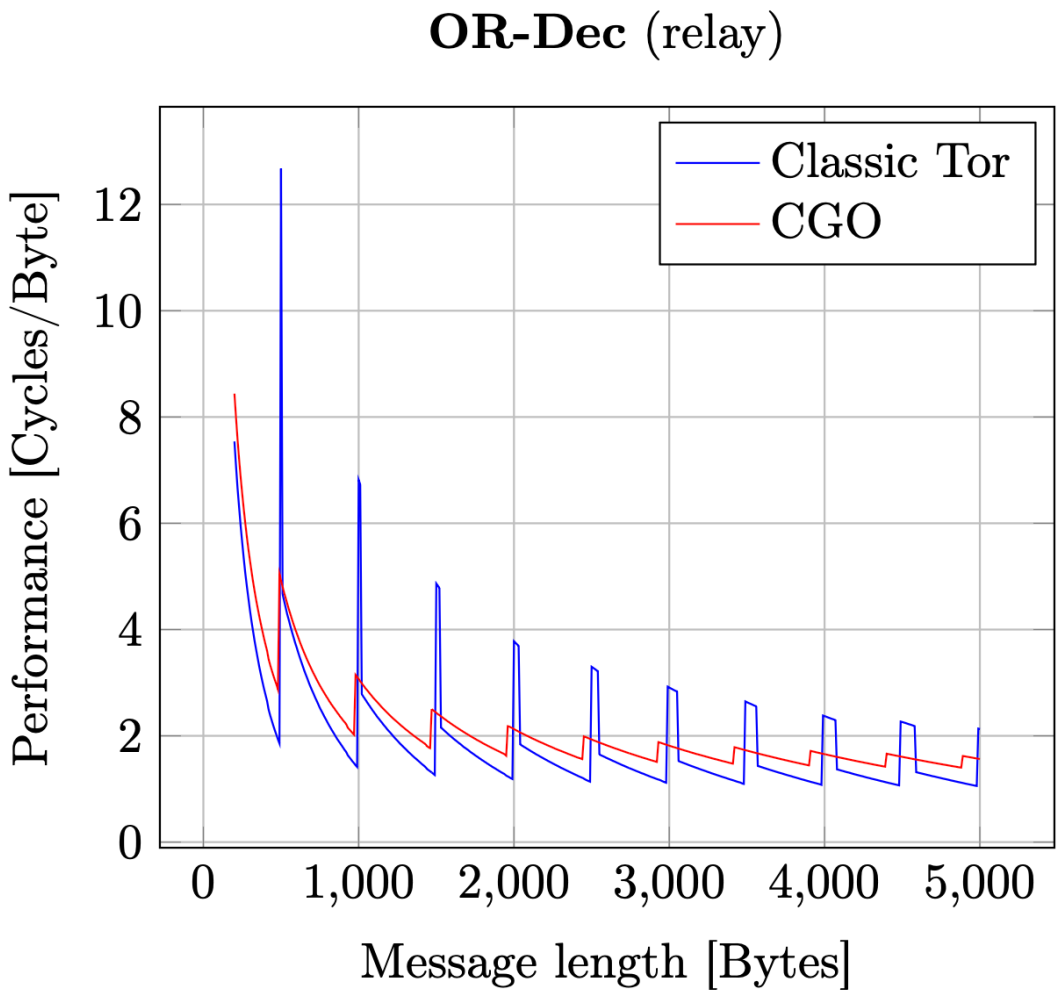


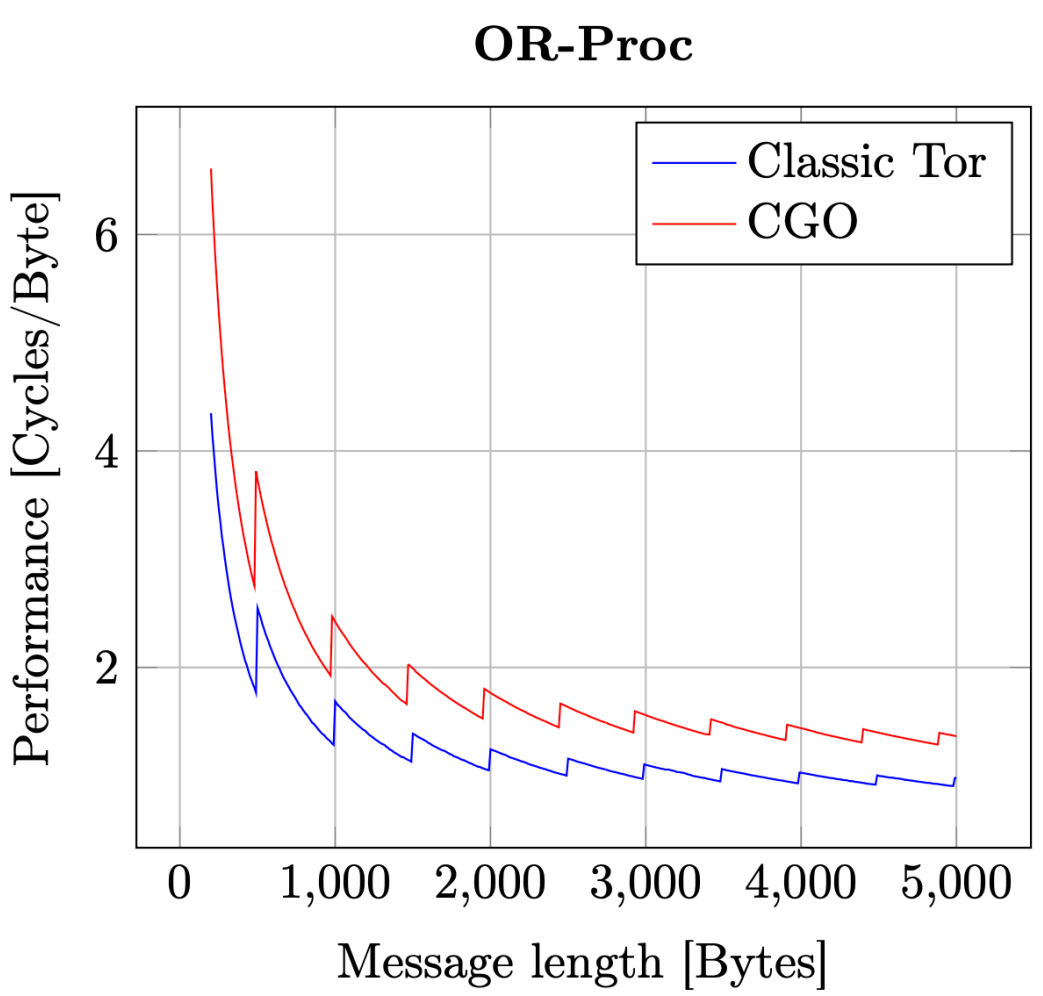
Performance





Performance





Roadmap

1. Motivation
2. Tor refresher
3. What we have now, and why it’s bad
4. Cryptographic preliminaries
5. Counter Galois Onion
6. **Conclusions**

Insights

Good news:

- Implementation is underway both for current (C) and future (project Arti, the Rust Tor implementation) codebases
- Supports heterogeneous deployment
- Under a certain model, provably secure

Something to think about:

- “All models are wrong, but some are useful”
- It is a quite targeted fix, in no way a panacea
- “most of us at Tor aren't cryptographers ourselves”[1]
- Performance is best case scenario
- Still missing CGO negotiation for Onion Services

Thank you for your attention

Questions?

Reference

[1] J. P. Degabriele, “Counter galois onion (CGO) a new onion encryption scheme for tor,” 2026. https://iacr.org/submit/files/slides/2026/rwc/rwc2026/87/87_slides.pdf

[2] J. P. Degabriele, A. Melloni, Jean-Pierre Münch, and M. Stam, “Counter galois onion (CGO) for tor: Fast non-malleable onion encryption,” 2025. https://eprint.iacr.org/2025/583

[3] J. P. Degabriele, A. Melloni, and M. Stam, “Counter Galois Onion: A new proposal for forward-secure relay cryptography,” The Tor Project, 308, 2019. Available: https://spec.torproject.org/proposals/308-counter-galois-onion.html

[4] J. P. Degabriele, A. Melloni, and M. Stam, “Secure onion encryption and the case of counter galois onion,” 2025. https://eprint.iacr.org/2025/2017

[5] J. P. Degabriele and M. Stam, “Untagging tor: A formal treatment of onion encryption,” 2018. https://eprint.iacr.org/2018/162

[6] J. P. Degabriele and Vukašin Karadžić, “Overloading the nonce: Rugged PRPs, nonce-set AEAD, and order-resilient channels,” 2022. https://eprint.iacr.org/2022/817

Reference

[7] M. Liskov, R. L. Rivest, and D. Wagner, “Tweakable block ciphers,” M. Yung, Ed., Springer Berlin Heidelberg, 2002, pp. 31–46.

[8] N. Mathewson, “Two improved relay encryption protocols for Tor cells,” The Tor Project, 202, Jun. 2012. Available: https://spec.torproject.org/proposals/202-improved-relay-crypto.html

[9] N. Mathewson, “Counter Galois Onion: Improved encryption for Tor circuit traffic,” The Tor Project, Nov. 2025. https://blog.torproject.org/introducing-cgo/ (accessed May 2026).

[10] The, “The Tor Project,” 2026. https://www.torproject.org

[11] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao, “One cell is enough to break Tor’s anonymity,” 2009, pp. 578–589. Available: https://blackhat.com/presentations/bh-dc-09/Fu/BlackHat-DC-09-Fu-Break-Tors-Anonymity.pdf

[12] R. Dingledine, N. Mathewson, S. J. Murdoch, and P. Syverson, “Tor: The second-generation onion router (2014 DRAFT v1),” The Tor Project, 2014. Available: https://murdoch.is/papers/tor14design.pdf

Reference

[13] The23rd Raccoon, “How I learned to stop ph34ring NSA and love the base rate fallacy,” 2008. https://archives.seul.org/or/dev/Sep-2008/msg00016.html

[14] The23rd Raccoon, “[tor-dev] analysis of the relative severity of tagging attacks,” 2012. https://archives.seul.org/or/dev/Mar-2012/msg00019.html

[15] R. Pries, W. Yu, X. Fu, and W. Zhao, “A new replay attack against anonymous communication networks,” IEEE, 2008. doi: https://doi.org/10.1109/ICC.2008.305.

[16] The Tor Project, Tor specifications. 2026. Available: https://spec.torproject.org/

Game $\mathbf{PRP}(\mathcal{D}, \mathsf{E})$

1 $b \leftarrow\$ \{0,1\}$
2 $\mathsf{K} \leftarrow\$ \mathcal{K}$
3 $\Pi \leftarrow\$ \mathsf{Perms}[\mathcal{X}]$
4 $b' \leftarrow\$ \mathcal{D}^{\mathrm{FN}}()$
5 **return** $b' = b$

Game $\mathbf{SPRP}(\mathcal{D}, \mathsf{E})$

1 $b \leftarrow\$ \{0,1\}$
2 $\mathsf{K} \leftarrow\$ \mathcal{K}$
3 $\Pi \leftarrow\$ \mathsf{Perms}[\mathcal{X}]$
4 $b' \leftarrow\$ \mathcal{D}^{\mathrm{FN},\mathrm{FN}^{-1}}()$
5 **return** $b' = b$

Oracle $\mathrm{FN}(x)$

1 **if** $b = 0$ **then**
2 $\quad y \leftarrow \mathsf{E}_{\mathsf{K}}(x)$
3 **elseif** $b = 1$ **then**
4 $\quad y \leftarrow \Pi(x)$
5 **return** y

Oracle $\mathrm{FN}^{-1}(y)$

1 **if** $b = 0$ **then**
2 $\quad x \leftarrow \mathsf{E}_{\mathsf{K}}^{-1}(y)$
3 **elseif** $b = 1$ **then**
4 $\quad x \leftarrow \Pi^{-1}(y)$
5 **return** x

Game $\mathbf{TPRP}(\mathcal{D}, \mathsf{E})$

$b \leftarrow\$ \{0,1\}$
$\mathsf{K} \leftarrow\$ \mathcal{K}$
$b' \leftarrow\$ \mathcal{D}^{\mathrm{FN}}()$
return $b' = b$

Game $\mathbf{TSPRP}(\mathcal{D}, \mathsf{E})$

$b \leftarrow\$ \{0,1\}$
$\mathsf{K} \leftarrow\$ \mathcal{K}$
$b' \leftarrow\$ \mathcal{D}^{\mathrm{FN},\mathrm{FN}^{-1}}()$
return $b' = b$

Oracle $\mathrm{FN}(t, x)$

if Π_t is undefined **then**
  $\Pi_t \leftarrow\$ \mathsf{Perms}[\mathcal{X}]$
if $b = 0$ **then**
  $y \leftarrow \mathsf{E}_{\mathsf{K}}(t, x)$
elseif $b = 1$ **then**
  $y \leftarrow \Pi_t(x)$
return y

Oracle $\mathrm{FN}^{-1}(t, y)$

if Π_t is undefined **then**
  $\Pi_t \leftarrow\$ \mathsf{Perms}[\mathcal{X}]$
if $b = 0$ **then**
  $x \leftarrow \mathsf{E}_{\mathsf{K}}^{-1}(t, y)$
elseif $b = 1$ **then**
  $x \leftarrow \Pi_t^{-1}(y)$
return x

Game $\mathbf{RPRP}^{\mathcal{A},v}_{\widetilde{\mathsf{EE}}}$

1 $K \leftarrow\$\ \mathcal{K}$
2 $b \leftarrow\$\ \{0,1\}$
3 $\mathcal{F},\mathcal{R},\mathcal{U} \leftarrow \emptyset,\emptyset,\emptyset$
4 $\widetilde{\Pi} \leftarrow\$\ \mathsf{IC}(\mathcal{T}, \mathcal{X}_L \times \mathcal{X}_R)$
5 $b' \leftarrow \mathcal{A}^{\mathrm{EN},\mathrm{DE},\mathrm{GU}}$
6 **return** $b = b'$

Oracle $\textsc{En}(T, X_L, X_R)$

1 **if** $b = 0$ **then**
2 $\quad (Y_L, Y_R) \leftarrow \widetilde{\Pi}(T, X_L, X_R)$
3 **else**
4 $\quad (Y_L, Y_R) \leftarrow \widetilde{\mathsf{EE}}_K(T, X_L, X_R)$
5 $\mathcal{F} \xleftarrow{\cup} \{Y_L\}; \mathcal{U} \xleftarrow{\cup} \{(T, Y_L, Y_R)\}$
6 **return** (Y_L, Y_R)

Y_L must never have ben queried either to Encryption oracle or Decryption oracle

Oracle $\textsc{De}(T, Y_L, Y_R)$

1 **if** $Y_L \in \mathcal{F} \cup \mathcal{R}$ **then**
2 $\quad$ **return** $\perp$
3 **if** $b = 0$ **then**
4 $\quad (X_L, X_R) \leftarrow \widetilde{\Pi}^{-1}(T, Y_L, Y_R)$
5 **else**
6 $\quad (X_L, X_R) \leftarrow \widetilde{\mathsf{EE}}_K^{-1}(T, Y_L, Y_R)$
7 $\mathcal{R} \xleftarrow{\cup} \{Y_L\}; \mathcal{U} \xleftarrow{\cup} \{(T, Y_L, Y_R)\}$
8 **return** (X_L, X_R)

Set of guesses

Oracle $\textsc{Gu}(T, Y_L, Y_R, \boldsymbol{V})$

1 **if** $((T, Y_L, Y_R) \in \mathcal{U}) \vee (|\boldsymbol{V}| > v)$ **then**
2 $\quad$ **return** $\perp$
3 **if** $b = 0$ **then**
4 $\quad$ **return false**
5 **else**
6 $\quad (X_L, X_R) \leftarrow \widetilde{\mathsf{EE}}_K^{-1}(T, Y_L, Y_R)$
7 $\quad$ **return** $X_L \in \boldsymbol{V}$

Security parameter

No trivial wins

Same as RPRP but substitutes PRF with a lazily evaluated two-sided function

Experiment $\mathbf{Exp}^{\mathrm{RRND}-b}_{\mathsf{EE}}(\mathbb{A})$

1 $K \leftarrow\$ \{0,1\}^k$
2 $\hat{b} \leftarrow \mathbb{A}^{\mathrm{En,De,Gu}}$
3 **return** $\hat{b}$

Oracle $\mathrm{En}(H, X_L, X_R)$

1 **if** $b = 1$ **then**
2 $\quad (Y_L, Y_R) \leftarrow \mathsf{EE}^H_K(X_L, X_R)$
3 **else**
4 $\quad (Y_L, Y_R) \leftarrow\$ \mathcal{D}_L \times \mathcal{D}_R$
5 $\mathcal{F} \xleftarrow{\cup} Y_L, \mathcal{U} \xleftarrow{\cup} (H, Y_L, Y_R)$
6 **return** (Y_L, Y_R)

Oracle $\mathrm{Gu}(H, Y_L, Y_R, X'_L)$

1 **if** $(H, Y_L, Y_R) \in \mathcal{U}$ **then**
2 $\quad$ **return** $\perp$
3 **if** $b = 1$ **then**
4 $\quad (X_L, X_R) \leftarrow \mathsf{ED}^H_K(Y_L, Y_R)$
5 $\quad$ **return** $X_L = X'_L$
6 **else**
7 $\quad$ **return false**

Experiment $\mathbf{Exp}^{\mathrm{URRND}-b}_{\mathsf{EE,EU}}(\mathbb{A})$

1 $K \leftarrow\$ \{0,1\}^k$
2 $\hat{b} \leftarrow \mathbb{A}^{\mathrm{En,De,Gu,Up}}$
3 **return** $\hat{b}$

Oracle $\mathrm{De}(H, Y_L, Y_R)$

1 **if** $Y_L \in \mathcal{F}$ **then**
2 $\quad$ **return** $\perp$
3 **if** $b = 1$ **then**
4 $\quad (X_L, X_R) \leftarrow \mathsf{ED}^H_K(Y_L, Y_R)$
5 **else**
6 $\quad (X_L, X_R) \leftarrow\$ \mathcal{D}_L \times \mathcal{D}_R$
7 $\mathcal{F} \xleftarrow{\cup} Y_L, \mathcal{U} \xleftarrow{\cup} (H, Y_L, Y_R)$
8 **return** (X_L, X_R)

Oracle $\mathrm{Up}(T)$

1 **if** $b = 1$ **then**
2 $\quad (\overline{K}, \overline{T}) \leftarrow \mathsf{EU}_K(T)$
3 **else**
4 $\quad (\overline{K}, \overline{T}) \leftarrow\$ \{0,1\}^k \times \mathcal{D}_L$
5 **return** $(\overline{K}, \overline{T})$

In URRND, we also have Update oracle

Only 1 query!

The adversary can interact with ℓ independent instances of the updatable split-domain tweakable cipher.

Experiment $\mathbf{Exp}^{\mathrm{CRND}_\ell - b}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})$

```
for h = 1 to ℓ
    K_{h[0]} ←$ {0,1}^k
    fresh_h ← true
    i_h ← 0
b̂ ← A^{En,De,Gu,Up,Co}
return b̂
```

h = instance handle
j = key version

Oracle $\mathrm{Up}(h, T)$

```
if h ∈ Z then
    return ⊥
if b = 1 then
    (K_{h[i_h+1]}, T) ← EU_{K_{h[i_h]}}(T)
else
    K_{h[i_h+1]}, T ←$ {0,1}^k × D_L
i_h ← i_h + 1, fresh_h ← true
return T
```

Only the updated T is returned

Oracle $\mathrm{Co}(h)$

```
if (h ∈ Z) ∨ (fresh_h = false) then
    return ⊥
Z ∪← h
return K_{h[i_h]}
```

Adversary can corrupt cipher instances to obtain the latest key version

Oracle $\mathrm{En}(h, j, H, X_L, X_R)$

```
if (j > i_h) ∨ (j = i_h ∧ h ∈ Z) then
    return ⊥
if j = i_h then
    fresh_h ← false
if b = 1 then
    (Y_L, Y_R) ← EE^H_{K_{h[j]}}(X_L, X_R)
else
    (Y_L, Y_R) ←$ D_L × D_R
F^j_h ∪← Y_L, U^j_h ∪← (H, Y_L, Y_R)
return (Y_L, Y_R)
```

Oracle $\mathrm{De}(h, j, H, Y_L, Y_R)$

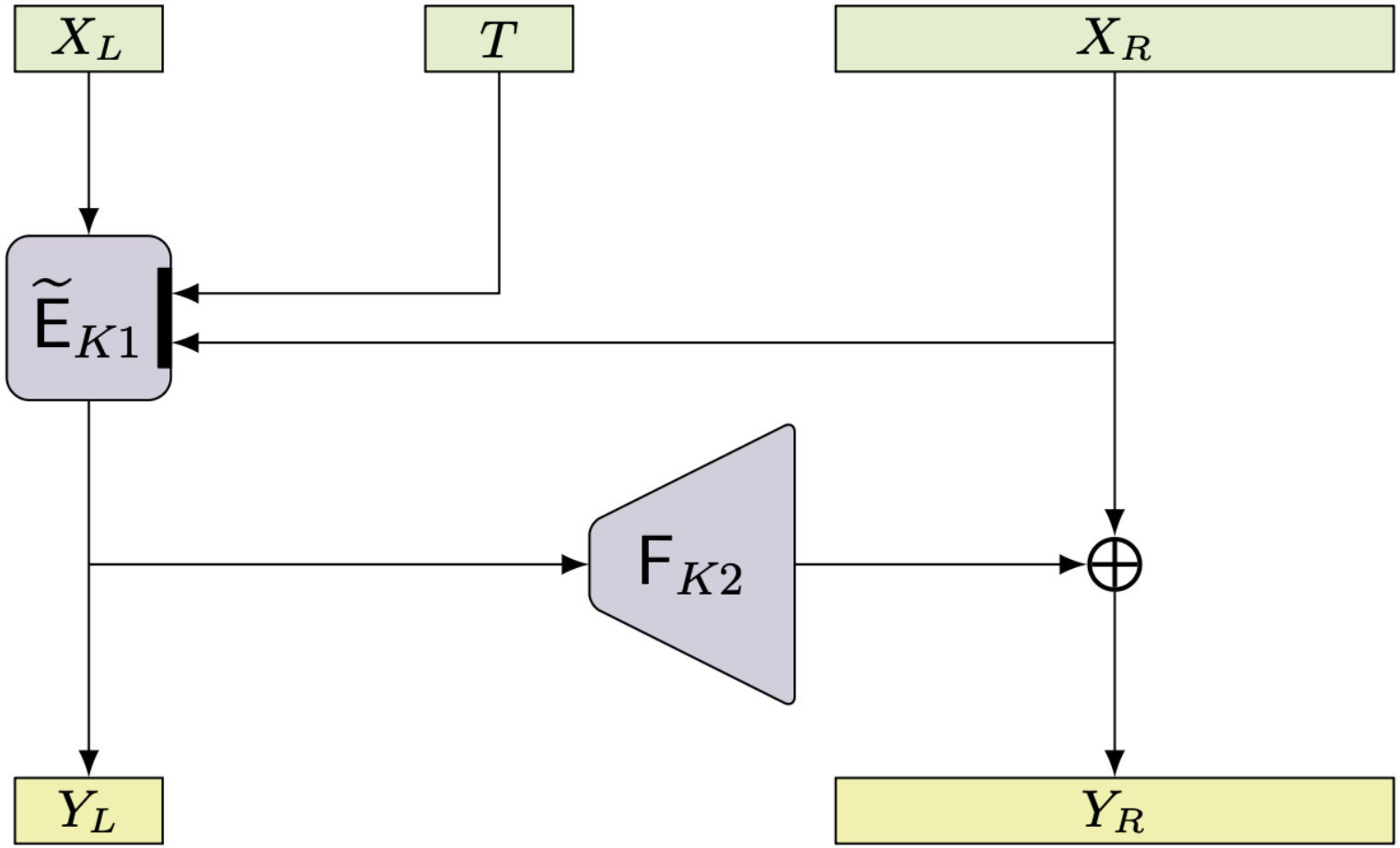
```
if (j > i_h) ∨ (j = i_h ∧ h ∈ Z) ∨ (Y_L ∈ F^j_h) then
    return ⊥
if j = i_h then
    fresh_h ← false
if b = 1 then
    (X_L, X_R) ← ED^H_{K_{h[j]}}(Y_L, Y_R)
else
    (X_L, X_R) ←$ D_L × D_R
F^j_h ∪← Y_L, U^j_h ∪← (H, Y_L, Y_R)
return (X_L, X_R)
```

Oracle $\mathrm{Gu}(h, j, H, Y_L, Y_R, X'_L)$

```
if (j > i_h) ∨ (j = i_h ∧ h ∈ Z) ∨ (H, Y_L, Y_R) ∈ U^j_h then
    return ⊥
if j = i_h then
    fresh_h ← false
if b = 1 then
    (X_L, X_R) ← ED^H_{K_{h[j]}}(Y_L, Y_R)
    return X_L = X'_L
else
    return false
```

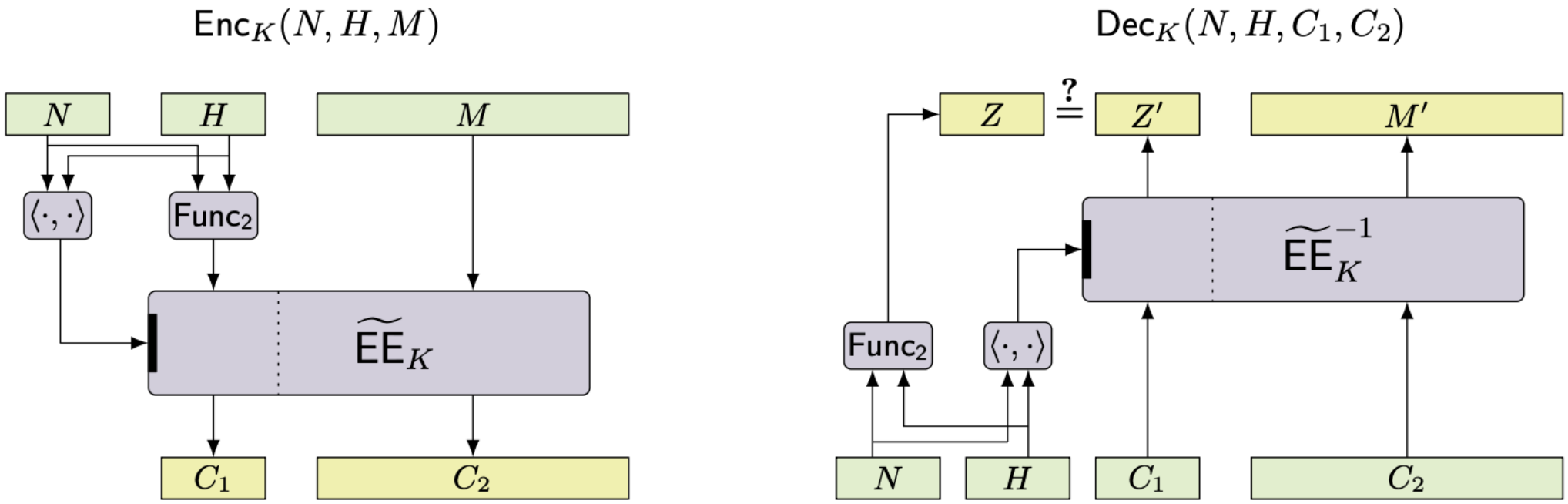
Corrupted keys in Z cannot be updated, nor used anymore

UIV



- E is a tweakable blockcipher with the domain XL = {0,1}^n and with tweak space T×X_R
- F variable-output-length pseudorandom function with domain XL and range XR

UIV as a Encode then Encipher



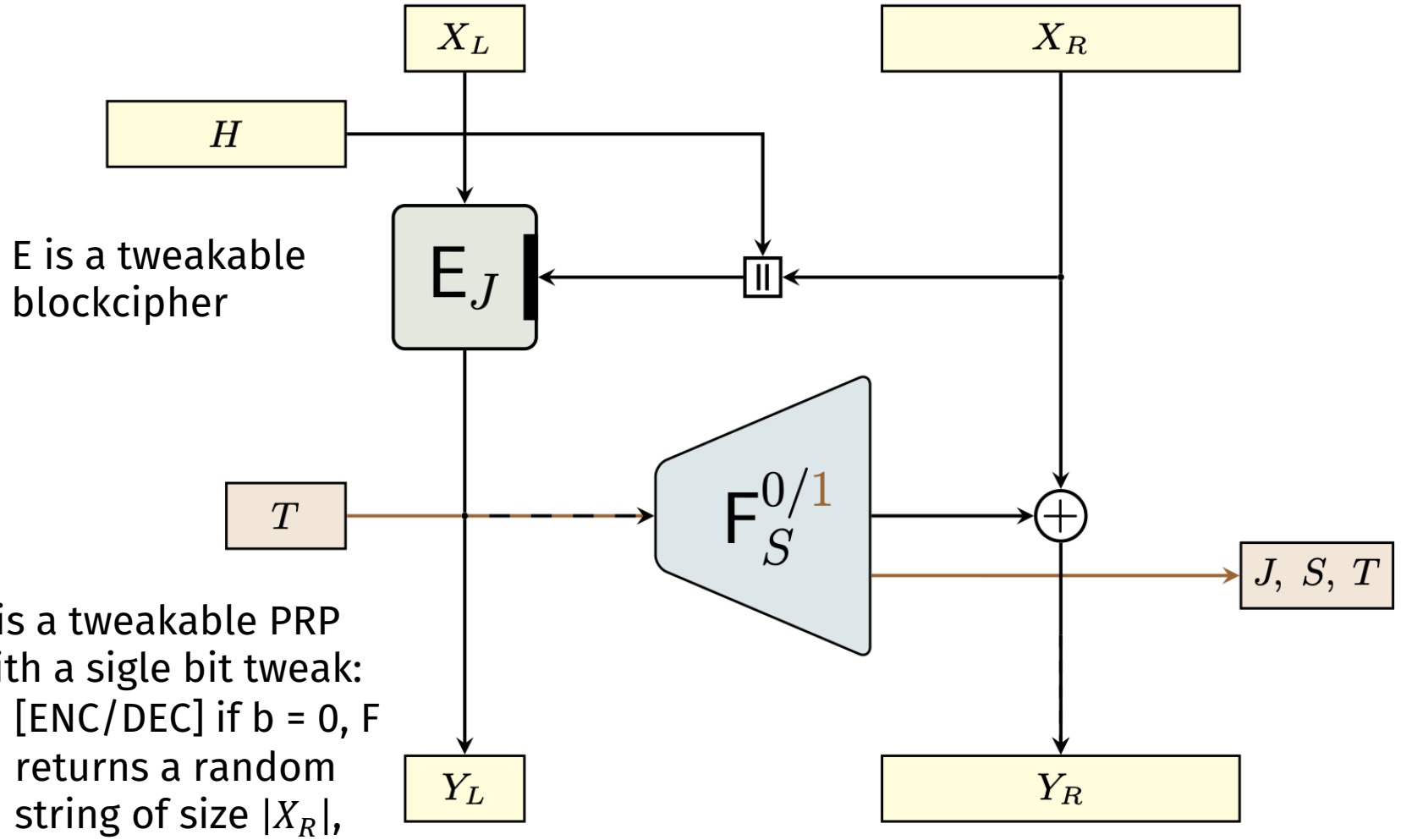
Theorem 2. *Let* EtE *be the nonce-based AEAD scheme defined in Fig. 6 realized from a tweakable cipher over the domain* $\{0,1\}^n \times \{0,1\}^{\geq m}$. *Then for any adversary* $\mathcal{A}$ *making* q_e *encryption queries and* q_v *verification queries, there exists an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathrm{mrae}}_{\mathtt{EtE}}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{\mathrm{rprp}}_{\widetilde{\mathsf{EE}}}(\mathcal{B}, 1) + \frac{q_e^2}{2^{n+m+1}},$$

where $\mathcal{B}$ *makes* q_e *encipher queries,* q_v *guess queries, and its runtime is similar to that of* $\mathcal{A}$.

Misuse-resistant: security degrades gracefully

UIV+



E is a tweakable blockcipher

F is a tweakable PRP with a sigle bit tweak:

- [ENC/DEC] if b = 0, F returns a random string of size $|X_R|$,
- [UPDATE] if b = 1, it returns a new state

$\mathsf{EU}_K(T)$

$(J, S) \twoheadleftarrow K$

$(\overline{K}, \overline{T}) \leftarrow \mathsf{F}_S^1(T)$

return $(\overline{K}, \overline{T})$

$\mathsf{EE}_K^H(X_L, X_R)$

$(J, S) \twoheadleftarrow K$

$Y_L \leftarrow \mathsf{E}_J^{H \| X_R}(X_L)$

$Y_R \leftarrow \mathsf{F}_S^0(Y_L) \oplus X_R$

return (Y_L, Y_R)

$\mathsf{ED}_K^H(Y_L, Y_R)$

$(J, S) \twoheadleftarrow K$

$X_R \leftarrow \mathsf{F}_S^0(Y_L) \oplus Y_R$

$X_L \leftarrow \mathsf{D}_J^{H \| X_R}(Y_L)$

return (X_L, X_R)

Security of UIV+

- Assume we have a RPRP
- We can show that RRND is equivalent up to a birthday bound

Lemma 1 (Lemma 6. in [21]). *Let $\mathcal{A}$ be an adversary that does not forward the result of one oracle to another. Then it holds*

$$\left|\Pr\left[\widetilde{\Pi} \leftarrow\$ \mathsf{IC}(\mathcal{T},\mathcal{X}) : \mathcal{A}^{\widetilde{\Pi}(\cdot,\cdot),\widetilde{\Pi}^{-1}(\cdot,\cdot)}\right] - \Pr\left[\widetilde{\mathsf{RR}} \leftarrow\$ \pm\mathrm{Func}(\mathcal{T},\mathcal{X}) : \mathcal{A}^{\widetilde{\mathsf{RR}}(+,\cdot,\cdot),\widetilde{\mathsf{RR}}(-,\cdot,\cdot)}\right]\right| \leq \frac{q(q-1)}{2^{b+1}},$$

where q is the total number of $\mathcal{A}$'s queries to both oracles and b is the length of the shortest element in $\mathcal{X}$.

- Extend RRND to include a single update: URRND
- Extend to *continuous* key updates over ℓ keys: CRRND_ℓ

Security of UIV+

- For the UIV construction, the following claim related to its RRND security can be proven

$$\mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{UIV}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{stprp}}_{\mathsf{E}}(\mathbb{B}^{\mathbb{A}}) + \mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathbb{C}^{\mathbb{A}}) + \frac{v \cdot q_{\mathrm{Gu}}}{2^{n-1}} + \frac{q_1(q_1-1)}{2^{n+1}} + \frac{q_{\mathrm{En}}(q_{\mathrm{En}}-1)}{2^{n+1}},$$

- We can show that combining security for the update and RRND security grants URRND security and, that URRND implies CRND_ℓ

$$\mathrm{RRND} \xrightarrow[\text{Lemma 1}]{+\mathrm{UPDT}} \mathrm{URRND} \xrightarrow[\text{Lemma 3}]{\cdot q_{\mathrm{Up}}} \mathrm{CRND} \xrightarrow[\text{Lemma 4}]{\cdot \ell} \mathrm{CRND}_\ell$$

$$\mathrm{RRND} \xrightarrow{+\frac{q(q-1)}{2^{n+m+1}}} \mathrm{RPRP}$$

Security of UIV+

- We obtain the following chain of inequalities:

Lemma 3 and 4

$$\mathsf{Adv}^{\mathrm{CRND}_\ell}_{\mathsf{UIV}+}(\mathbb{A}) \le \ell \cdot q_{\mathrm{Up}} \cdot \mathsf{Adv}^{\mathrm{URRND}}_{\mathsf{UIV}+}(\mathbb{B}^{\mathbb{A}})$$

Lemma 1

$$\le \ell \cdot q_{\mathrm{Up}} \left(\mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{UIV}}(\mathbb{B}^{\mathbb{A}}) + \mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{UIV}+}(\mathbb{C}^{\mathbb{A}}) \right)$$

Security of UIV [6] and update (Lemma 5)

$$\le \ell \cdot q_{\mathrm{Up}} \left(\mathsf{Adv}^{\mathrm{stprp}}_{\mathsf{E}}(\mathbb{B}^{\mathbb{A}}) + 2 \cdot \mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathbb{C}^{\mathbb{A}}) + \frac{q_{\mathrm{Gu}}}{2^{n-1}} + \frac{q_1(q_1-1)}{2^{n+1}} + \frac{q_{\mathrm{En}}(q_{\mathrm{En}}-1)}{2^{n+1}} \right)$$

where technically B and C change from line to line

and $q_1 = q_{\mathrm{En}} + q_{\mathrm{De}} + q_{\mathrm{Gu}}$

Proof of Lemma 1

- Split the bit in URRND game into two different bits for Up and En/De/Gu oracles
- Play around with definitions and define some reductions that use/ignore the update oracle

Experiment $\mathsf{Exp}^{\text{URRND-}b_{\text{Up}},b_{\text{RRND}}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})$

$K \leftarrow\!\!\$\ \{0,1\}^k$

$\hat{b} \leftarrow \mathbb{A}^{\text{En,De,Gu,Up}}$

return $\hat{b}$

$\text{En}(H, X_L, X_R)$

if $b_{\text{RRND}} = 1$

  $(Y_L, Y_R) \leftarrow \mathsf{EE}^H_K(X_L, X_R)$

else

  $(Y_L, Y_R) \leftarrow\!\!\$\ \mathcal{D}_L \times \mathcal{D}_R$

$\mathcal{F} \xleftarrow{\cup} Y_L,\ \mathcal{U} \xleftarrow{\cup} (H, Y_L, Y_R)$

return (Y_L, Y_R)

$\text{Gu}(H, Y_L, Y_R, X'_L)$

if $(H, Y_L, Y_R) \in \mathcal{U}$

  return $\lightning$

if $b_{\text{RRND}} = 1$

  $(X_L, X_R) \leftarrow \mathsf{ED}^H_K(Y_L, Y_R)$

  return $X_L = X'_L$

else

  return false

$\text{De}(H, Y_L, Y_R)$

if $Y_L \in \mathcal{F}$

  return $\lightning$

if $b_{\text{RRND}} = 1$

  $(X_L, X_R) \leftarrow \mathsf{ED}^H_K(Y_L, Y_R)$

else

  $(X_L, X_R) \leftarrow\!\!\$\ \mathcal{D}_L \times \mathcal{D}_R$

$\mathcal{F} \xleftarrow{\cup} Y_L,\ \mathcal{U} \xleftarrow{\cup} H, Y_L, Y_R)$

return (X_L, X_R)

$\text{Up}(T)$

if $b_{\text{Up}} = 1$

  $(\overline{K}, \overline{T}) \leftarrow \mathsf{EU}_K(T)$

else

  $(\overline{K}, \overline{T}) \leftarrow\!\!\$\ \{0,1\}^k \times \mathcal{D}_L$

return $(\overline{K}, \overline{T})$

Proof of Lemma 1

Definition 3 (URRND, UPDT, RRND **advantages**). *For any split-domain tweakable cipher* EE *over* $\mathcal{D}_L \times \mathcal{D}_R$ *and any associated update function* EU*, the corresponding* URRND, UPDT, *and* RRND *advantages of an adversary* $\mathbb{A}$ *are given by:*

$$\mathsf{Adv}^{\mathrm{URRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}0,0}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}1,1}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right].$$

$$\mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}0,1}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}1,1}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right].$$

$$\mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}0,0}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{URRND\text{-}0,1}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right].$$

where $\mathsf{Exp}^{\mathrm{URRND}\text{-}b_{\mathrm{Up}},b_{\mathrm{RRND}}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})$ *is defined in Fig. 11.*

Proof of Lemma 1

Lemma 1 (RRND $\wedge$ UPDT **security** $\implies$ URRND **security**). *Let* $(\mathsf{EE}, \mathsf{EU})$ *be an updatable tweakable split-domain cipher and* $\mathbb{A}$ *an* URRND *adversary* $\mathbb{A}$ *against* $(\mathsf{EE}, \mathsf{EU})$. *Then*

$$\mathsf{Adv}^{\mathrm{URRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) + \mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{EE}}(\mathbb{A}^{-}) \,.$$

Proof. The result comes straightforward from the definition:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{URRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) &= \Pr\left[\mathsf{Exp}^{\text{URRND-}0,0}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{URRND-}1,1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] \\
&= \Pr\left[\mathsf{Exp}^{\text{URRND-}0,0}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{URRND-}0,1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] \\
&\qquad + \Pr\left[\mathsf{Exp}^{\text{URRND-}0,1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{URRND-}1,1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) = 1\right] \\
&= \mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) + \mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) \\
&\leq \mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) + \mathsf{Adv}^{\mathrm{RRND}}_{\mathsf{EE}}(\mathbb{A}^{-}) \,.
\end{aligned}$$

etc.

Proof of Lemma 3

Lemma 3 (URRND **security** $\implies$ CRND **security**). *Let* $(\mathsf{EE}, \mathsf{EU})$ *be an updatable split-domain tweakable cipher. Then there exists a simple fully black box reduction* $\mathbb{B}$ *such that, for all* CRND *adversaries* $\mathbb{A}$ *against* $(\mathsf{EE}, \mathsf{EU})$ *performing* q_{Up} *queries to the oracle* Up,

$$\mathsf{Adv}^{\mathrm{CRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) \leq q_{\mathrm{Up}} \cdot \mathsf{Adv}^{\mathrm{URRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{B}^{\mathbb{A}}).$$

- Define a simplified CRND game which is cleaned up $\mathrm{CRND}_{\ell=1}$
- URRND can be viewed as a single key experiment in which the adversary can obtain what would become the next key and continue to query the oracles En,De, and Gu on the original key
- Perform hybrid argument using game Exp^{k}, where we answer to the first k Up queries with random elements, then real EU

Experiment $\mathsf{Exp}^{\mathrm{k}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})$

$K_0 \leftarrow\!\!\$\ \{0,1\}^k$
$fresh \leftarrow \mathsf{true}$
$i \leftarrow 0$
$\hat{b} \leftarrow \mathbb{A}^{\mathrm{En},\mathrm{De},\mathrm{Gu},\mathrm{Up},\mathrm{Co}}$
return $\hat{b}$

$\mathrm{Up}(T)$

if $\mathcal{Z} \neq \emptyset \vee i = q_{\mathrm{Up}}$ **then return** $\lightning$
if $i \geq k$ **then** $(K_{i+1}, T) \leftarrow \mathsf{EU}_{K_i}(T)$
else $(K_{i+1}, T) \leftarrow\!\!\$\ \{0,1\}^k \times \mathcal{D}_L$
$i \leftarrow i + 1$
$fresh \leftarrow \mathsf{true}$
return T

$\mathrm{Gu}(j, H, Y_L, Y_R, X'_L)$

if $j > i$ **then return** $\lightning$
if $\mathcal{Z} \neq \emptyset \wedge j = i$ **then return** $\lightning$
$(X_L, X_R) \leftarrow \mathsf{ED}^H_{K_j}(Y_L, Y_R)$
$fresh \leftarrow \mathsf{false}$
return $X_L = X'_L$

$\mathrm{En}(j, H, X_L, X_R)$

if $j > i$ **then return** $\lightning$
if $\mathcal{Z} \neq \emptyset \wedge j = i$ **then return** $\lightning$
$(Y_L, Y_R) \leftarrow \mathsf{EE}^H_{K_j}(X_L, X_R)$
$\mathcal{F}^j \xleftarrow{\cup} Y_L$
$fresh \leftarrow \mathsf{false}$
return (Y_L, Y_R)

$\mathrm{De}(j, H, Y_L, Y_R)$

if $j > i$ **then return** $\lightning$
if $\mathcal{Z} \neq \emptyset \wedge j = i$ **then return** $\lightning$
if $Y_L \in \mathcal{F}^j$ **then return** $\lightning$
$(X_L, X_R) \leftarrow \mathsf{ED}^H_{K_j}(Y_L, Y_R)$
$\mathcal{F}^j \xleftarrow{\cup} Y_L$
$fresh \leftarrow \mathsf{false}$
return (X_L, X_R)

Co

if $\mathcal{Z} \neq \emptyset$ **then return** $\lightning$
if $fresh = \mathsf{false}$ **then return** $\lightning$
$\mathcal{Z} \leftarrow i$
return K_i

Proof of Lemma 3

- Perform hybrid argument using game Exp^{k}, where we answer to the first k Up queries with random elements, then real EU
- Then CRND with b=0 corresponds to $\mathrm{Exp}^{\mathrm{q_{Up}}}$ (adversary does at most $\mathrm{q_{Up}}$ queries, and all are random)
- CRND with b=1 corresponds to Exp^{0} (all real queries)
- a single step from $k-1$ to k in $\mathsf{Exp}^{\mathrm{k}}_{\mathsf{EE},\mathsf{EU}}$ corresponds to $\mathsf{Exp}^{\mathrm{URRND}\text{-}b_{\mathrm{Up}},b_{\mathrm{RRND}}}_{\mathsf{EE}}$

$$
\begin{aligned}
\mathsf{Adv}^{\mathrm{CRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) &= \Pr\left[\mathsf{Exp}^{\mathrm{CRND}\text{-}1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{CRND}\text{-}0}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] \\
&= \Pr\left[\mathsf{Exp}^{0}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] - \Pr\left[\mathsf{Exp}^{q_{\mathrm{Up}}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] = \sum_{k=1}^{q_{\mathrm{Up}}} \left(\Pr\left[\mathsf{Exp}^{\mathrm{k}-1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{k}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right]\right) \\
&\leq q_{\mathrm{Up}} \cdot \max_{k=1,\ldots,q_{\mathrm{Up}}} \left(\Pr\left[\mathsf{Exp}^{\mathrm{k}-1}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{k}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A})=1\right]\right)
\end{aligned}
$$

Proof of Lemma 4

Lemma 4 (CRND **security** $\implies$ CRND_ℓ **security**). *Let* $(\mathsf{EE}, \mathsf{EU})$ *be an updatable tweakable split-domain cipher. Then there exists a simple fully black box reductions* $\mathbb{B}$ *such that, for all* CRND_ℓ *adversaries* $\mathbb{A}$ *against* $(\mathsf{EE}, \mathsf{EU})$,

$$\mathsf{Adv}^{\mathrm{CRND}_\ell}_{\mathsf{EE},\mathsf{EU}}(\mathbb{A}) \leq \ell \cdot \mathsf{Adv}^{\mathrm{CRND}}_{\mathsf{EE},\mathsf{EU}}(\mathbb{B}^{\mathbb{A}}).$$

- Introduce sequence number h (represents an onion router in circuit)
- “By a standard hybrid argument”

Proof of Lemma 5

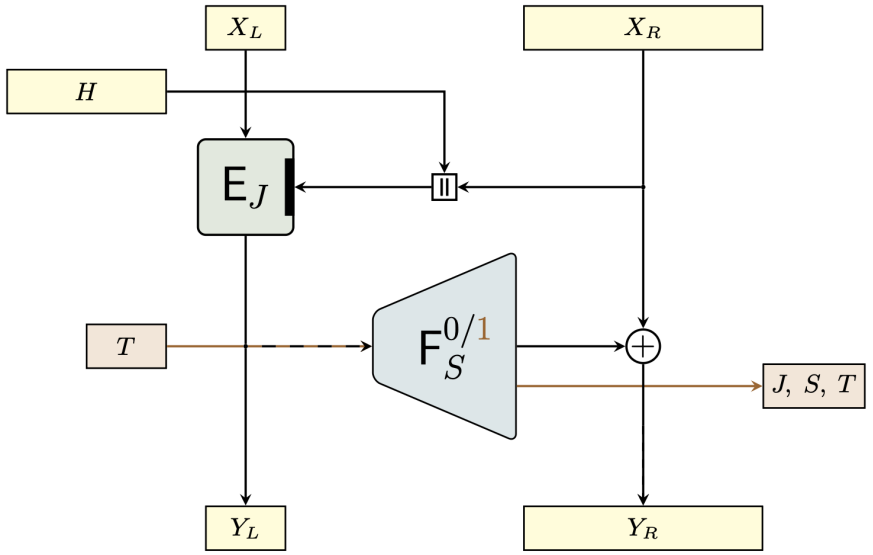
Lemma 5 (PRF Security of F $\implies$ UPDT security). *Consider* UIV+ *as updatable tweakable split-domain cipher. Then there exists a simple fully black box reductions* $\mathbb{B}$ *such that, for all* UPDT *adversaries* $\mathbb{A}$ *against* UIV+,

$$\mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{UIV+}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathbb{B}^{\mathbb{A}}).$$

MULTI-ORACLE PRF GAME!

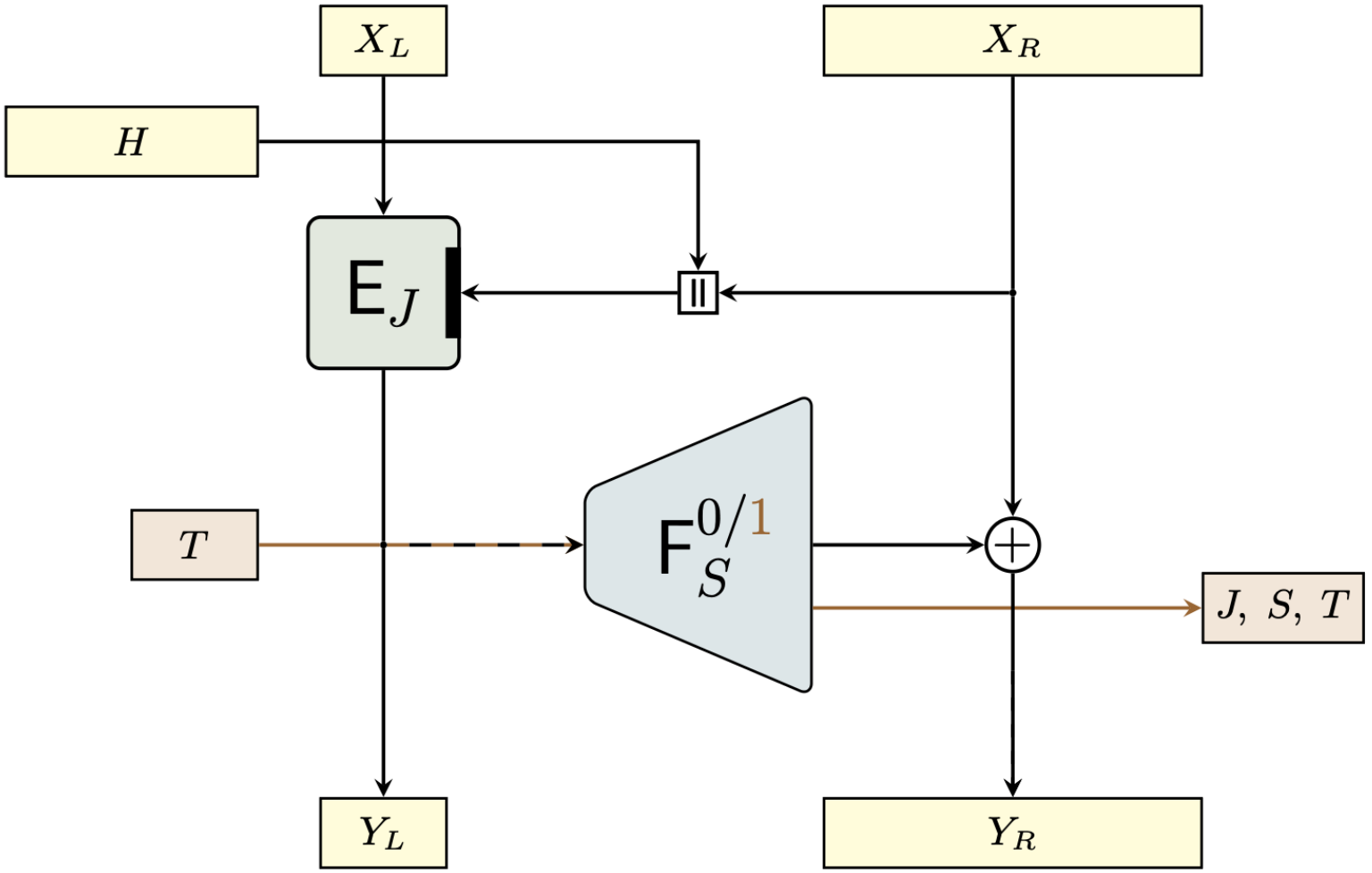
Proof (sketch). Recall that, by definition,

$$\mathsf{Adv}^{\mathrm{UPDT}}_{\mathsf{EE,EU}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{URRND-0,1}}_{\mathsf{EE,EU}}(\mathbb{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{URRND-1,1}}_{\mathsf{EE,EU}}(\mathbb{A}) = 1\right],$$



- K = (J,S)
- B is given **real** oracle and challenge oracle
  - with restriction of not being able to query both on the same inputs (no trivial wins)
- Reduction B can sample J, and use challenge oracle for queries to F with tweak 0
- For queries to F with tweak 1, use real oracle
- As domains don't overlap the simulation is sound

GCM-UIV+



$\mathsf{EU}_K(T)$

$(J, S) \twoheadleftarrow K$

$(\overline{K}, \overline{T}) \leftarrow \mathsf{F}_S^1(T)$

return $(\overline{K}, \overline{T})$

$\mathsf{EE}_K^H(X_L, X_R)$

$(J, S) \twoheadleftarrow K$

$Y_L \leftarrow \mathsf{E}_J^{H\|X_R}(X_L)$

$Y_R \leftarrow \mathsf{F}_S^0(Y_L) \oplus X_R$

return (Y_L, Y_R)

$\mathsf{ED}_K^H(Y_L, Y_R)$

$(J, S) \twoheadleftarrow K$

$X_R \leftarrow \mathsf{F}_S^0(Y_L) \oplus Y_R$

$X_L \leftarrow \mathsf{D}_J^{H\|X_R}(Y_L)$

return (X_L, X_R)

GCM-UIV+

- In CGO, we instantiate UIV+ using GCM components to take advantage of common hardware instructions

$\mathsf{F}^b_S(T)$ | J | = 256 = | S | = 32byte

$(L, B) \leftarrow S,\ Z = \varepsilon$

if $b = 0$

for $i = 0$ **to** 30

$Z \leftarrow Z \,\|\, \mathsf{AES}_L(\lfloor \mathsf{POLYVAL}(B, T) \rfloor_{122} \,\|\, i)$

else

for $i = 31$ **to** 35

$Z \leftarrow Z \,\|\, \mathsf{AES}_L(\lfloor \mathsf{POLYVAL}(B, T) \rfloor_{122} \,\|\, i)$

return Z

31 x 16 bytes = 496

Hash-then-PRF

5 x 16 bytes = 256 + 256 + 128

J S T

last 3 bytes of keystream are discarded

Ciphertext is 509 bytes:

- X_L is 16 bytes
- X_R is 509 – 16 = 493 bytes

$\mathsf{E}^{H \| X_R}_J(X_L)$

$(\overline{L}, \overline{B}) \leftarrow J$

$Z \leftarrow \mathsf{POLYVAL}(\overline{B}, H \,\|\, X_R)$

$Y_L \leftarrow Z \oplus \mathsf{AES}_{\overline{L}}(Z \oplus X_L)$

return Y_L

$\mathsf{D}^{H \| X_R}_J(Y_L)$

$(\overline{L}, \overline{B}) \leftarrow J$

$Z \leftarrow \mathsf{POLYVAL}(\overline{B}, H \,\|\, X_R)$

$X_L \leftarrow Z \oplus \mathsf{AES}^{-1}_{\overline{L}}(Z \oplus Y_L)$

return X_L

AES-128 + POLYVAL (universal hash function)

with 128 bit key, block size and output

6-bit counter is enough for domain separation between b = 0 / 1

Security of GCM-UIV+

Proposition 1 (Security of E (LRW2 cf. Theorem 2 [33])). *Let* E *be the tweakable blockcipher construction described in Fig. 9 composed from 128-bit* AES *and an* 2^{-123}*-AXU hash function. Then there exists a simple, query-preserving, fully black box reduction* $\mathbb{B}$ *such that for any* STPRP *adversary* $\mathbb{A}$ *making* q *queries,*

$$\mathsf{Adv}^{\mathrm{STPRP}}_{\mathsf{E}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{SPRP}}_{\mathsf{AES}}(\mathbb{B}^{\mathbb{A}}) + \frac{3q^2}{2^{124}}\,.$$

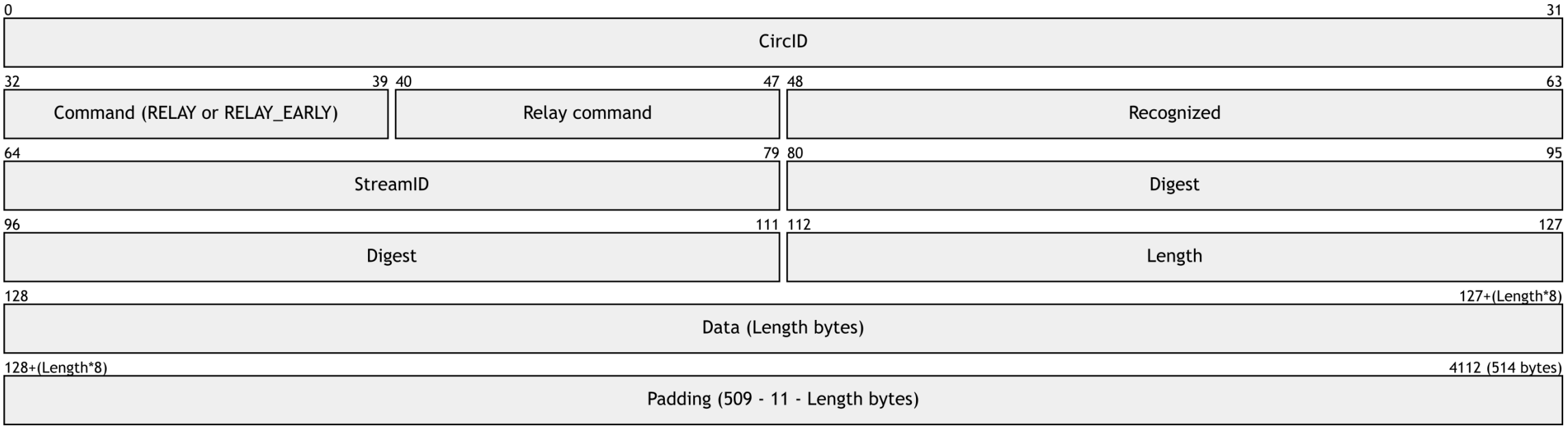
Proposition 2 (Security of F (Hash-then-PRF cf. Lemma 5.1 [31])). *Let* F *be the tweakable pseudorandom function construction described in Fig. 9 composed from counter-mode* AES *and an* ϵ*-AU hash function mapping* 128*-bit strings to* 122*-bit strings. Then there exists a simple, query-preserving, fully black box reduction* $\mathbb{B}$ *such that for any* PRF *adversary* $\mathbb{A}$ *making* q *queries,*

$$\mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{AES\text{-}CTR}}(\mathbb{B}^{\mathbb{A}}) + \frac{q^2}{2}\epsilon\,.$$

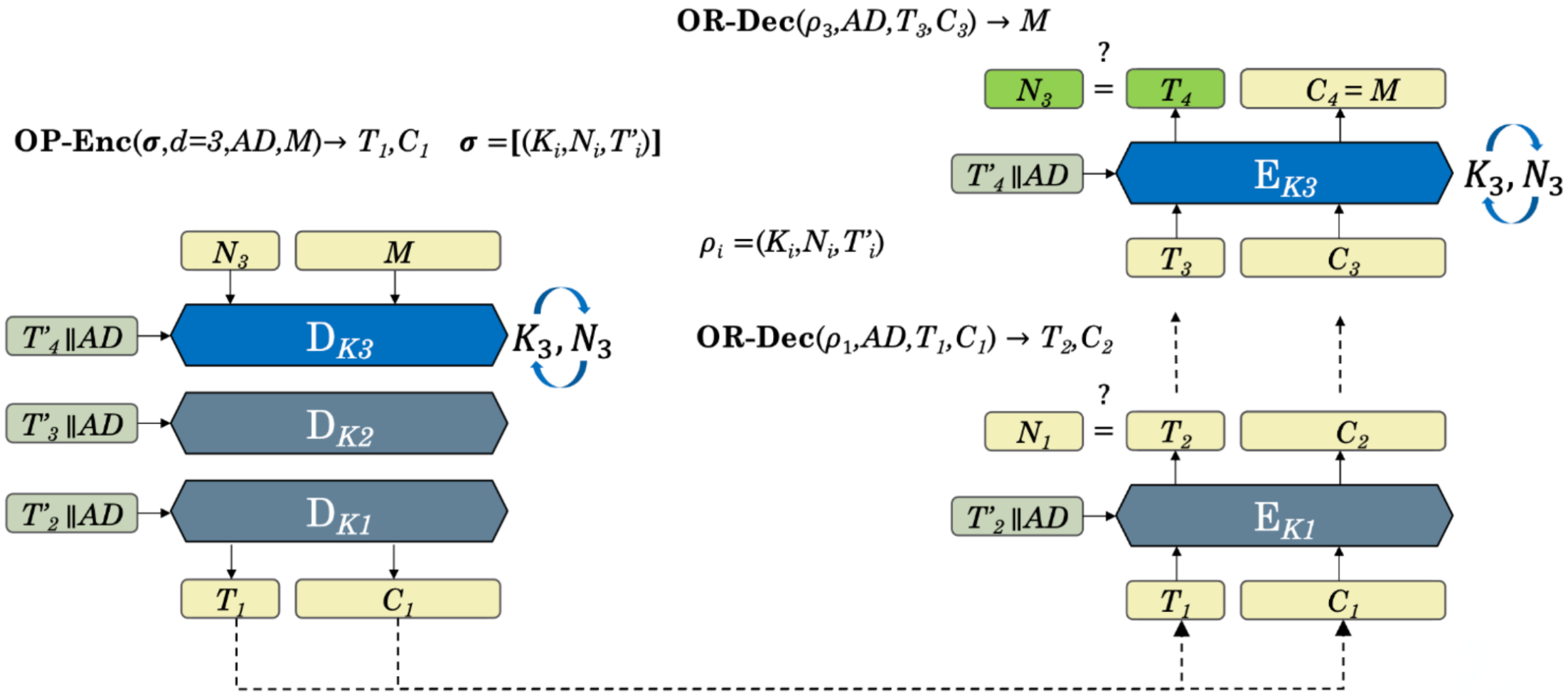
Proposition 3 (Security of AES-CTR). *Let* AES-CTR *be AES in counter mode limited to* 36 *blocks of output per invocation. Then there exists a simple, fully black box reduction* $\mathbb{B}$ *such that for any* PRF *adversary* $\mathbb{A}$ *against* AES-CTR *making* q *queries,*

$$\mathsf{Adv}^{\mathrm{PRF}}_{\mathsf{AES\text{-}CTR}}(\mathbb{A}) \leq \mathsf{Adv}^{\mathrm{PRP}}_{\mathsf{AES}}(\mathbb{B}^{\mathbb{A}}) + \frac{(36q)^2}{2^{128+1}}\,,$$

Tor relay protocol cell format



CGO – forward direction



Per hop authentication

What if every router in the circuit verifies the cell's integrity?

- cell size must remain fixed
- a tag must be appended in every layer of encryption
- tags steal from plaintext space
- results in waste of bandwidth

"[...] with onion services, we'd be devoting something like 15% of our bandwidth to authenticator fields."

Other constructions

Wide-block ciphers:

- ciphers (or modes of using ciphers) that encrypt an entire message as if it were a single block in a regular block cipher
- if it is a SPRP it can resist tagging attacks by reserving a portion of the plaintext for a known value e.g., 16 bytes of zeros

Problem: it is provable that they require at least two passes of hash per hop

- expensive!

Proposed Solutions

Cryptography Research Centre

- In **Prop 261(2016)** Matthewson proposed a new Onion Encryption scheme where each encryption layer consists of a **Wide-Input Tweakable Cipher.**
- Intuitively, **WITC** are **non-expanding**, **non-malleable**, and **integrity** can be added by appending **redundant bits** to the plaintext.
- **[ADL17]** Propose **GCM-RUP,** an AEAD scheme realized from GCM components that is **secure** under the **release of unverified plaintext**.
- In **[ADL17]** GCM-RUP is also proposed for **use in Tor**, claiming that a **WITC is overkill**, an OE scheme is described but **without any formal analysis**.

Onion Encryption from GCM-RUP

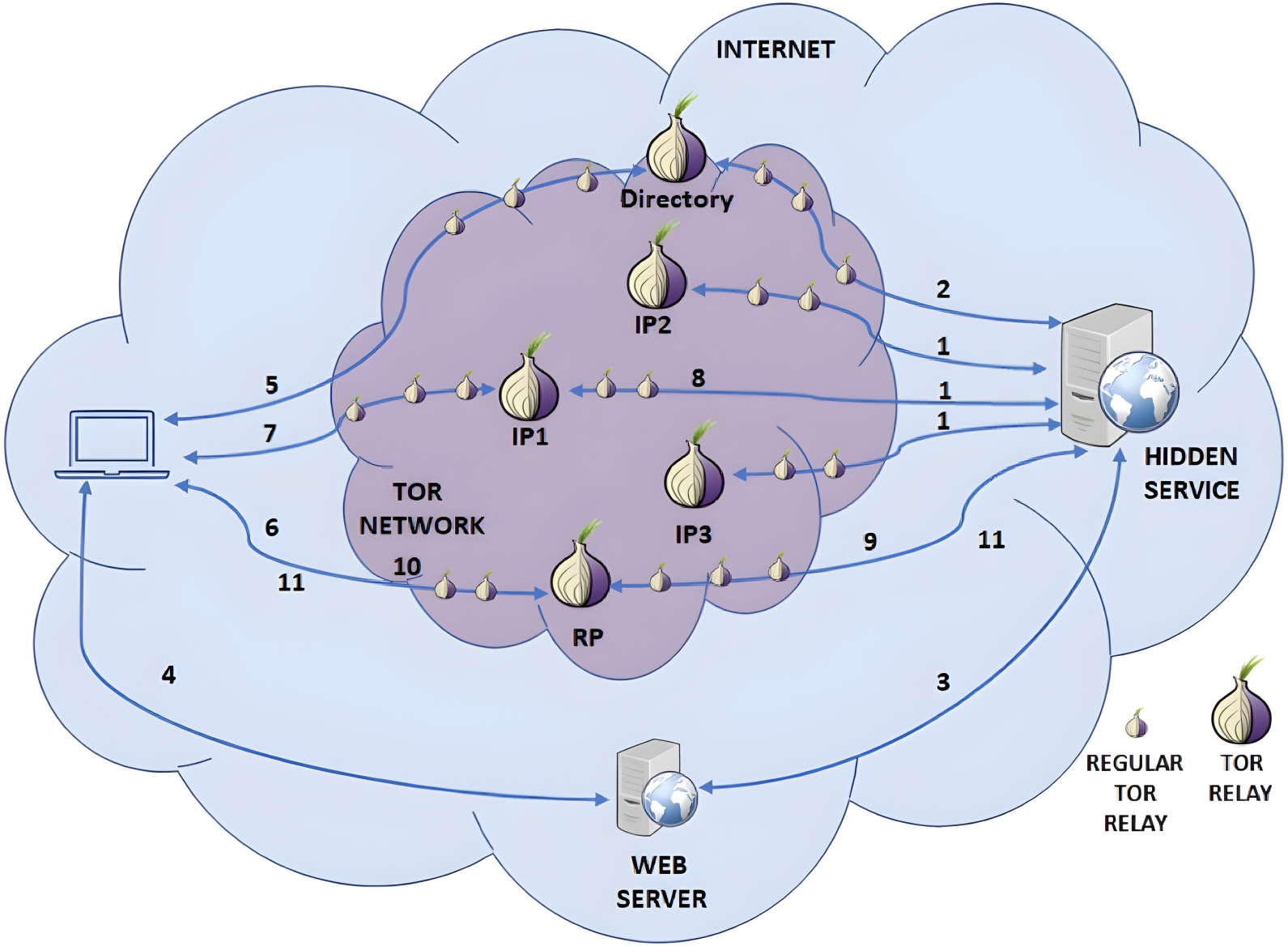
Cryptography Research Centre

- In [**ADL17**] the Onion Encryption scheme was **underspecified**, **no integrity**, **no leaky pipes**, **no ciphertext chaining**, and **not forward secure**.

- [**Prop 295-Jul 2018**] Updated version **added leaky pipes** and **integrity**, **but not IND-CPA secure**, still no **ciphertext chaining** and **forward security**.

- **[Prop 295-Mar 2019]** Heavily revised scheme, made stateful by **feeding back GHASH outputs** (invalidating proofs) and **extending GCM-RUP to full WITC (SPRP),** forfeiting the performance gains originally sought!

Onion services



source

Other attempts

	tor1 (Legacy Protocol)	**Proposal 261 (AEZ)**	**Proposal 295 (ADL)**	**CGO (UIV+ / Proposal 359)**
Primary Cipher Primitive	AES-128-CTR	AEZ Wide-block	PIV + GCM-RUP	AES-128 + POLYVAL
Mathematical Malleability	Highly Malleable	Non-Malleable	Flawed Randomness	Non-Malleable (via RPRP)
Integrity Check Mechanism	4-byte SHA-1 digest	Implicit (SPRP)	Appended MAC	16-byte structural tag
Forward Secrecy Profile	Per Circuit (Static over hours)	Per Circuit (Static)	Unclear/Vulnerable	Immediate (Per Cell Update)
Computational Overhead	Low (Fast)	Extremely High (Slow)	Moderate	Low (Hardware Accelerated)
Formal Anonymity Model	N/A (Pre-dates models)	Heuristic	Fails CircuitHiding	Provably CircuitHiding