# Onion-AE
## Foundations of Nested Encryption

**Phillip Rogaway** **Yusi Zhang**

University of California, Davis, USA

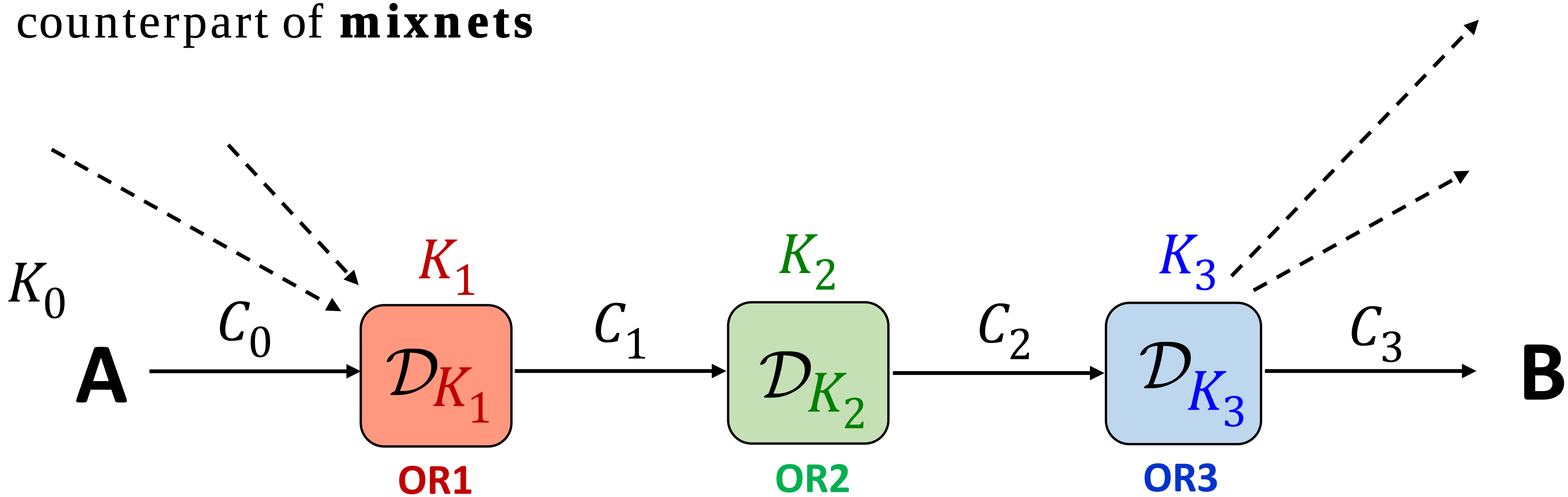
**Privacy Enhancing Technologies Symposium**
**(PETS 2018)**
Barcelona, Spain
Thursday – 26 July 2018

## **Nested encryption**
as used for **onion routing**

[Goldschlag, Reed, Syverson 1996a, 1996b]
[Syverson, Goldschlag, Reed 1997]
[Dingledine, Mathewson, Syverson 2004]

The symmetric, low-latency counterpart of **mixnets**

[Chaum 1981]




$$C_0 = \mathcal{E}_{K_1}(\mathcal{E}_{K_2}(\mathcal{E}_{K_3}(M)))$$

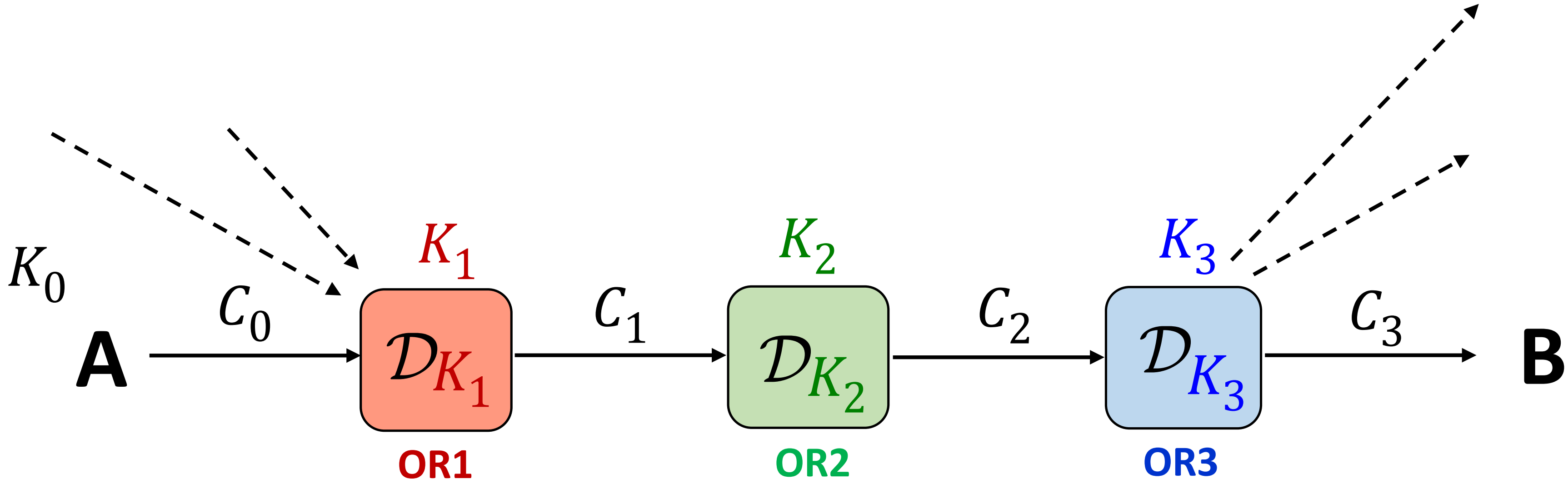
$$M = \mathbf{B} \,||\, M'$$

$$C_1 = \mathcal{E}_{K_2}(\mathcal{E}_{K_3}(M))$$

$$C_2 = \mathcal{E}_{K_3}(M)$$

$$C_3 = M$$

# What **problem** does nested encryption supposedly solve?




$$C_0 = \mathcal{E}_{K_1}(\mathcal{E}_{K_2}(\mathcal{E}_{K_3}(M)))$$

$$C_1 = \mathcal{E}_{K_2}(\mathcal{E}_{K_3}(M))$$

$$M = \mathbf{B} \,||\, M'$$

$$C_2 = \mathcal{E}_{K_3}(M)$$

$$C_3 = M$$

# What **problem** does nested encryption supposedly solve?

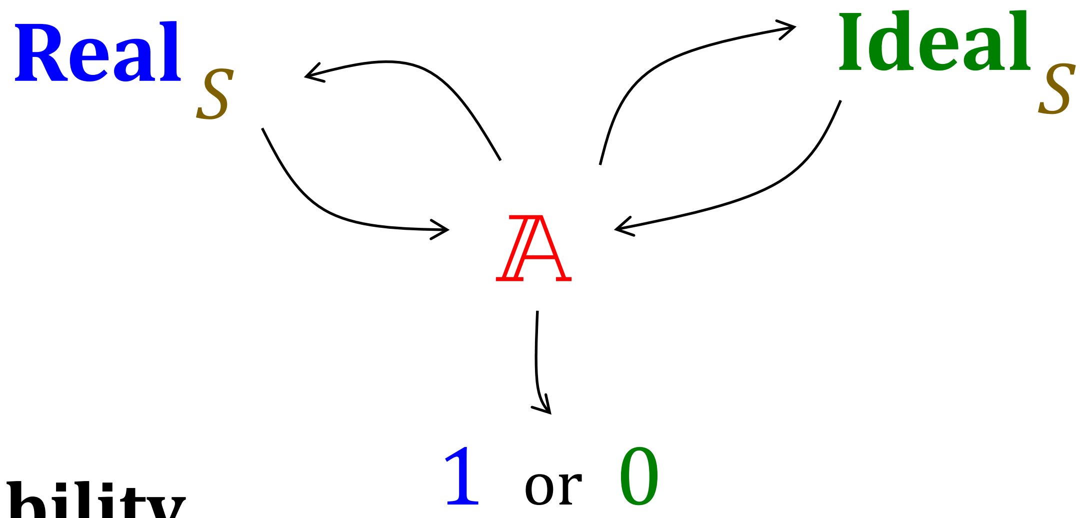
Concrete, self-contained, understandable.
Not building on UC [Canetti], [Camenisch, Lysyanskaya 2005]

A provable-security treatment of it

- Provide **syntax** and a **definition**
- Analyze **constructions**
    - Tor's relay protocol: doesn't satisfy our definition
    - LBE: does satisfy our definition

design 1 of proposal 202 of [Mathewson 2012]

If the underlying blockcipher is a tweakable wideblock PRP




**Indistinguishability**

**Advantage**

An adversary

$$\mathbf{Adv}_S(\mathbb{A}) = \mathbf{Pr}[\mathbb{A}^{\mathbf{Real}} \to 1] - \mathbf{Pr}[\mathbb{A}^{\mathbf{Ideal}} \to 1]$$

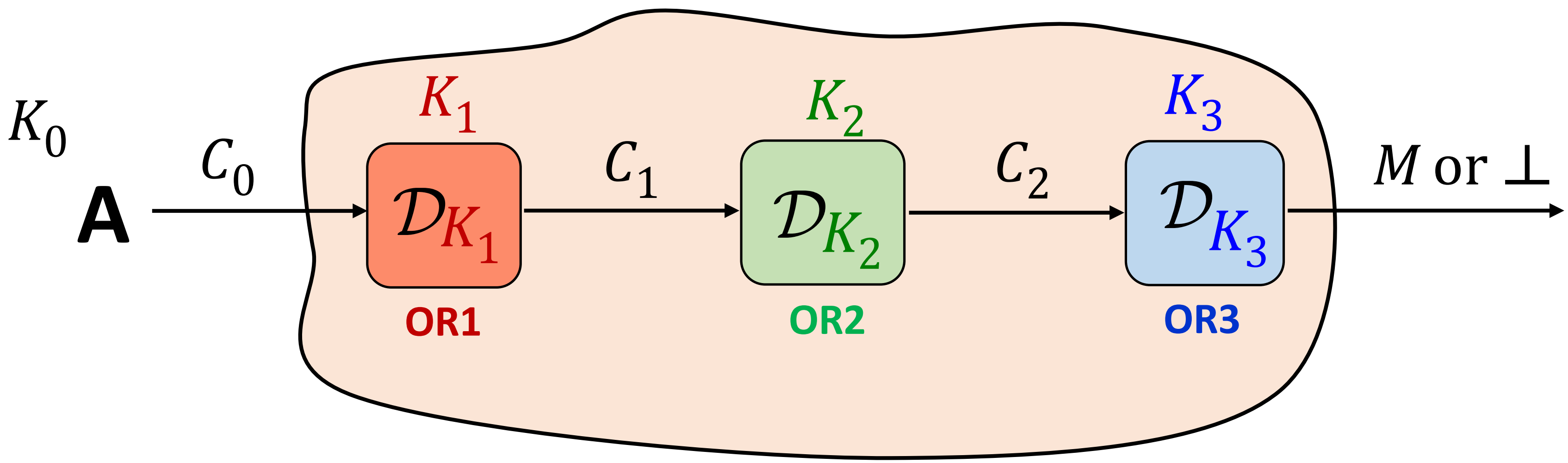
A protocol

**Seeing our problem as a type of**

# Authenticated Encryption (AE)

"Onion-AE"

Symmetric encryption that aims to achieve both **privacy** and **authenticity**



## Seeing our problem as a type of
# Authenticated Encryption (AE)

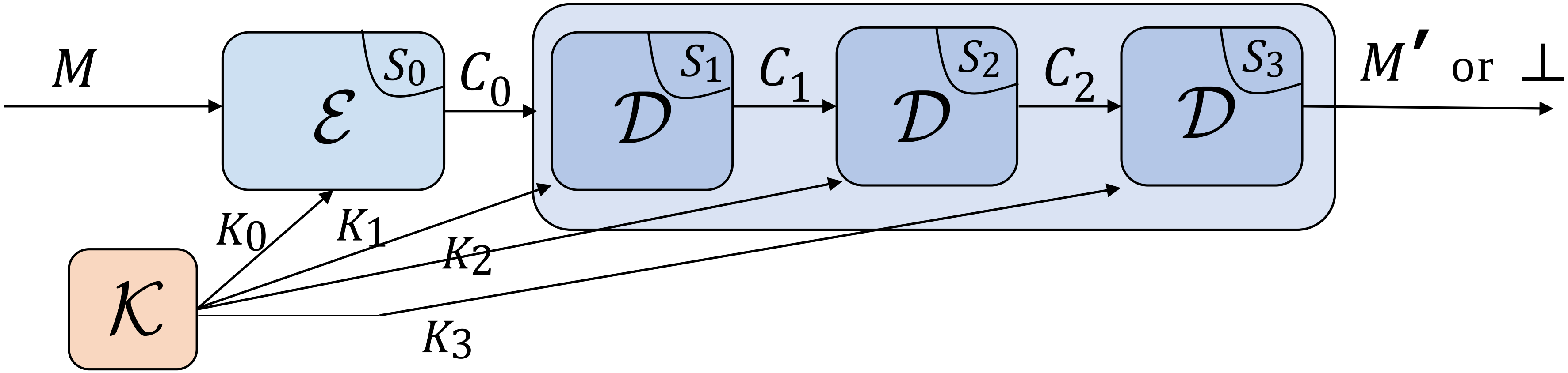
"Onion-AE"

Symmetric encryption that aims to achieve both **privacy** and **authenticity**

**Lots of flavors of AE already:**

- Probabilistic AE [Bellare, Rogaway 2000], [Katz, Yung 2000]
- Nonce-based AE [Rogaway, Bellare, Black, Krovetz 2001]
- Nonce-based AE with associated data (AEAD) [Rogaway 2002]
- Stateful AE [Bellare, Kohno, Namprempre 2004] ← Most closely related
- Misuse-Resistant AE [Rogaway, Shrimpton 2006]
- Release of Unverified Plaintext [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 2014]
- Robust AE [Hoang, Krovetz, Rogaway 2015]
- Online-AE [Hoang, Reyhanitabar, Rogaway, Vizár 2015]

# Onion-AE syntax



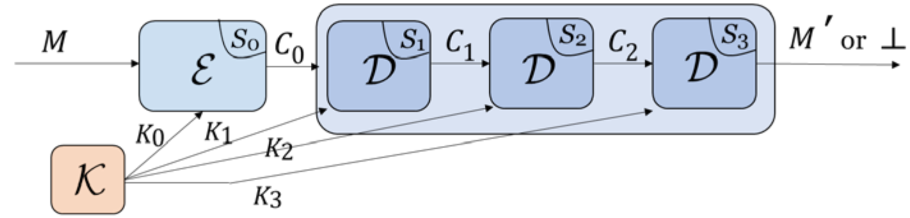

A 3-tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

$\mathcal{K}: \mathbb{N} \rightarrow \mathcal{K}^*$ maps *n* to *n*+1 strings

$\mathcal{E}: \mathcal{K} \times \mathcal{M} \times \mathcal{U} \rightarrow \mathcal{C} \times \mathcal{U}$

$\mathcal{D}: \mathcal{K} \times \mathcal{C} \times \mathcal{S} \rightarrow (\mathcal{M} \cup \mathcal{C} \cup \{\perp\}) \times \mathcal{S}$

# Correctness




$(\forall\ n)\ (K_0, K_1, \ldots, K_n) \twoheadleftarrow \mathcal{K}(n);\ (K_0, K_1, \ldots, K_n) \twoheadleftarrow \mathcal{K}(n)$

$(\forall\ t)\ (M_1, \ldots, M_t) \twoheadleftarrow \mathcal{M};\ \ S_0\ , S_1, \ldots, S_t \leftarrow \varepsilon$

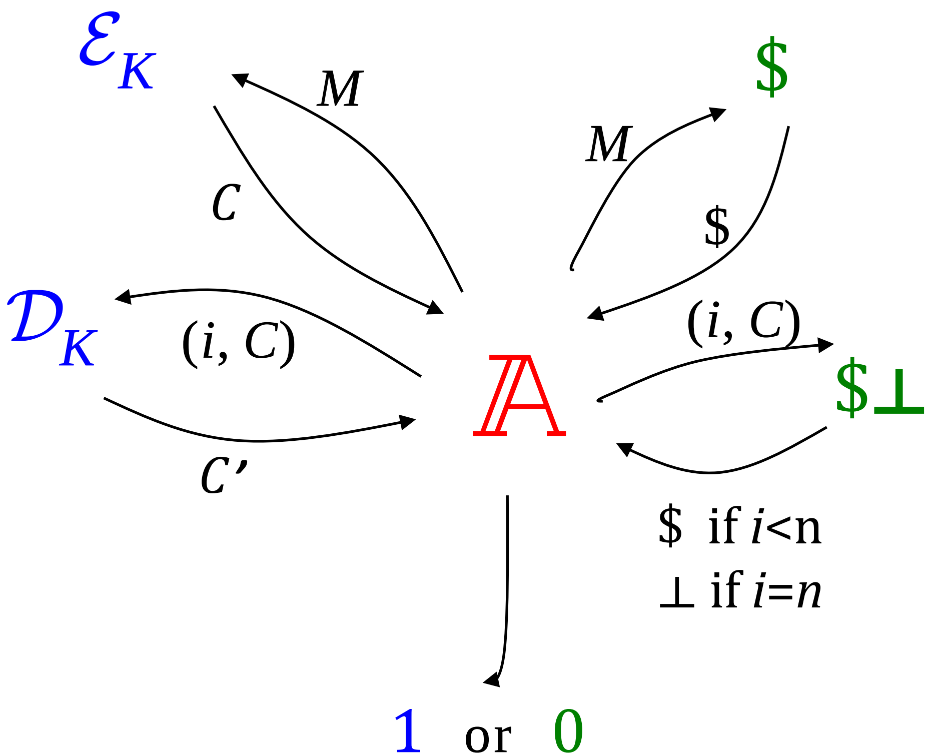
**for** $i \leftarrow 1$ **to** $t$ **do**

$(C_0, S_0) \leftarrow \mathcal{E}\ (K_i\ , M_i\ , S_0)$

**for** $j \leftarrow 1$ **to** $n$ **do** $(C_j\ , S_j) \leftarrow \mathcal{D}\ (K_j\ , C_{j-1}\ , S_j\ )$

**assert** $C_n = M_i$

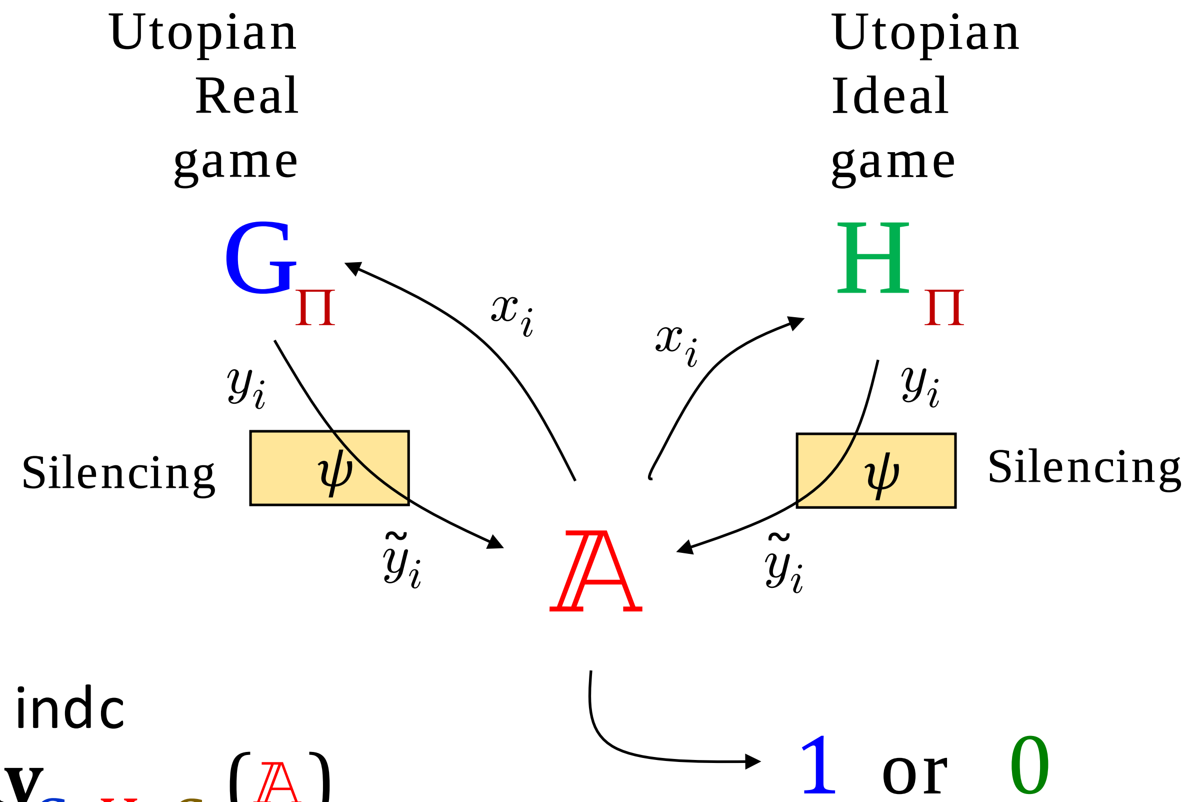
# Formalizing security




**Oracle silencing**:
behave like the **utopian** game shown **unless** the response you are about to give is **fixed** in every **correct** protocol.
In that case, answer ✧ .

Idea explored in CRYPTO 2018 paper.

# **IND|C** Indistinguishability up to correctness




$\mathbf{Adv}^{\text{indc}}_{G,\,H,\,C}(\mathbb{A})$

Silence an oracle response if, for the real game, given the transcript ***t*** so far, the answer is fully determined by $\Pi \in C$.

| | |
|---|---|
| $\underline{\textsc{Key}(n')}$ | $\underline{\textsc{Key}(n')}$ |
| **if** $n \neq \perp$ **then return** Err | **if** $n \neq \perp$ **then return** Err |
| $n \leftarrow n'$ | $n \leftarrow n'$ |
| $(k_0, \ldots, k_n) \twoheadleftarrow \mathcal{K}(n)$ | |
| $\underline{\textsc{Enc}(m)}$ | $\underline{\textsc{Enc}(m)}$ |
| **if** $n = \perp$ **then return** Err | **if** $n = \perp$ **then return** Err |
| $(c, u) \leftarrow \mathcal{E}(k_0, m, u)$ | $c \twoheadleftarrow \mathfrak{C}$ |
| **return** $c$ | **return** $c$ |
| $\underline{\textsc{Dec}(c, i)}$ | $\underline{\textsc{Dec}(c, i)}$ |
| **if** $n = \perp$ **then return** Err | **if** $n = \perp$ **then return** Err |
| $(d, s_i) \leftarrow \mathcal{D}(k_i, c, s_i)$ | **if** $i = n$ **then** $d \leftarrow \perp$ |
| **return** $d$ | **else** $d \twoheadleftarrow \mathfrak{C}$ |
| | **return** $d$ |

# Without oracle silencing

```
Game TRANSMIT^S_OE
  ϱ ← ε; n ← 0
  win ← false
  S^{Add,Enc,Pass}
  return win

Pass(i, j)
  if ¬(0 < j ≤ ℓ_i) ∨ Q^i_j = []
    return ↯
  c ← Q^i_j.dequeue()
  s ← p_i[j − 1]
  (v, w') ← map(i, j)
  w ← D(τ_v, s, c)
  if w ≠ w'
    win ← true
    return ⊥
  (τ̄_v[w], d, x) ← D̄(τ̄_v[w], s, c)
  if j < ℓ_i ∧ d = p_i[j + 1]
    Q^i_{j+1}.enqueue(x)
  elseif j = ℓ_i ∧ d = ⊘ ∧ x = m_i[ctr_i]
    ctr_i ← ctr_i + 1
  else win ← true
  return (d, x)

Enc(i, m)
  (v, w) ← map(i, 0)
  m_i.append(m)
  (σ_v[w], d, c) ← E(σ_v[w], m)
  if d ≠ p_i[1]
    win ← true
  else
    Q^i_1.enqueue(c)
  return (d, c)

Add(p)
  if |p| ≥ 1
    n ← n + 1
    p_n ← p; ℓ_n ← |p|
    ctr_n ← 1
    (ϱ, σ, t, t̄) ← G(ϱ, p)
    if |t| ≠ ℓ_n ∨ |t̄| ≠ ℓ_n
      win ← true
    σ_{p[0]}.append(σ)
    for j = 1 to ℓ_n
      v ← p[j]
      τ_v.append(t[j])
      τ̄_v.append(t̄[j])
  return n
```

**Concurrent work**
**[Degabriele, Stam 2018]**
***Untagging Tor: A Formal Treatment of Onion Encryption***

```
Game PINT^A_OE
  ϱ ← ε; n ← 0
  win ← false
  (C, st) ← A_1
  A_2^{Add,Enc,Proc}(st)
  return win

Add(p)
  if |p| ≥ 1
    n ← n + 1
    p_n ← p; ℓ_n ← |p|
    ctr_n ← 1; sync_n ← true
    (ϱ, σ, t, t̄) ← G(ϱ, p)
    σ_{p[0]}.append(σ)
    for j = 1 to ℓ_n
      v ← p[j]
      τ_v.append(t[j])
      τ̄_v.append(t̄[j])
  return (σ, t, t̄)|_C

Enc(i, m)
  (v, w) ← map(i, 0)
  if v ∈ C
    return ↯
  m_i.append(m)
  (σ_v[w], d, c) ← E(σ_v[w], m)
  return (d, c)

Proc(s, v, c)
  if v ∈ C
    return ↯
  w ← D(τ_v, s, c)
  if w = ⊥
    return ⊥
  (i, j) ← map^{-1}(v, w)
  (τ̄_v[w], d, x) ← D̄(τ̄_v[w], s, c)
  if d = ⊘ ∧ x ≠ ⊥
    if j = ℓ_i ∧ x = m_i[ctr_i]
      ctr_i ← ctr_i + 1
    else
      win ← true
  return (d, x)
```

# Without oracle silencing

```
Game C-HIDE^A_OE
  (W₀, W₁, C, st) ← A₁
  if ¬VALID(W₀, W₁, C)
    return false
  ∀i sync_i ← true
  ϱ ← ε; n ← 0; b ←$ {0, 1}
  INIT-CIRC(W_b)
  τ_C ← {(v, σ_v, τ_v, τ̄_v) | v ∈ C}
  b' ← A₂^{Enc,Net}(st, τ_C)
  return b = b'

NET(z)
  ∀i assc_i ← 0; x ← []
  for i' = 1 to |z|
    (s, v, c) ← z[i']
    w ← D(τ_v, s, c)
    if s ∉ C ∨ v ∈ C ∨ w = ⊥
      return ↯
  for i' = 1 to |z|
    (s, v, c) ← z[i']; c* ← c
    w ← D(τ_v, s, c)
    (τ̄_v[w], d, c) ← D̄(τ̄_v[w], s, c)
    (i, j) ← map(v, w)
    while d ∉ C ∧ d ≠ ⊘
      s ← v; v ← d
      w ← D(τ_v, s, c)
      (τ̄_v[w], d, c) ← D̄(τ̄_v[w], s, c)
    if d ∈ C
      x.append(v, d, c)
    if d ∈ C ∨ i ∈ I_nuw
      assc_i ← assc_i + 1
      if c* ≠ Q^i.dequeue()
        sync_i ← false
  if ⋁_{i∈I_en} (sync_i ∨ assc_i ≠ 1)
    return ↯
  return sort(x)

INIT-CIRC(W)
  for i = 1 to |W|
    n ← n + 1; p_n ← W[i]
    (ϱ, σ, t, t̄) ← G(ϱ, p_n)
    ℓ_n ← |p_n|
    sync_n ← true
    σ_{p_n[0]}.append(σ)
    for j = 1 to ℓ_n
      v ← p_n[j]
      τ_v.append(t[j])
      τ̄_v.append(t̄[j])
    if EN(p_n, C) ∧ p_n[0] ∉ C
      I_en ← I_en ∪ {i}
      if NOP(p_n, C)
        I_nuw ← I_nuw ∪ {i}
  foreach v
    Shuffle(σ_v, τ_v, τ̄_v)

ENC(i, m)
  (v, w) ← map(i, 0)
  if v ∈ C
    return ↯
  (σ_v[w], d, c) ← E(σ_v[w], m)
  while d ∉ C
    s ← v; v ← d
    w ← D(τ_v, s, c)
    (τ̄_v[w], d, c) ← D̄(τ̄_v[w], s, c)
  (v*, d*, c*) ← (v, d, c)
  while d ∈ C
    s ← v; v ← d
    w ← D(τ_v, s, c)
    (τ̄_v[w], d, c) ← D̄(τ̄_v[w], s, c)
  if d ≠ ⊘
    (i, j) ← map⁻¹(v, w)
    Q^i.enqueue(c)
  return (v*, d*, c*)
```

**Concurrent work**
**[Degabriele, Stam 2018]**
***Untagging Tor: A Formal Treatment of Onion Encryption***

```
Game LOR^A_OE
  ϱ ← ε; n ← 0
  win ← false
  b ←$ {0,1}
  (C, st) ← A₁
  b' ← A₂^{ADD,ENC,PROC}(st)
  return b = b'

ENC(i, m₀, m₁)
  (v, w) ← map(i, 0)
  if v ∈ C ∨ p_i[ℓ_i] ∈ C ∨ |m₀| ≠ |m₁|
    return ↯
  m_i.append(m_b)
  (σ_v[w], d, c) ← E(σ_v[w], m_b)
  return (d, c)

PROC(s, v, c)
  if v ∈ C
    return ↯
  w ← D(τ_v, s, c)
  if w = ⊥
    return ⊥
  (i, j) ← map⁻¹(v, w)
  (τ̄_v[w], d, x) ← D̄(τ̄_v[w], s, c)
  if j = ℓ_i ∧ d = ⊘
    if c = m_i[ctr_i] ∧ sync_i = true
      ctr_i ← ctr_i + 1
      return ↯
    else
      sync_i ← false
  return (d, x)
```

# Limitations on this treatment of onion-AE

**Relaxations sketched in the paper**

- Only attended to **outbound** messages
- No **corrupted** routers
- Fixed sequence of hops: **no "leaky pipe"**
- Authenticity checked only at **time of exit.**

**"Lazy authenticity"**

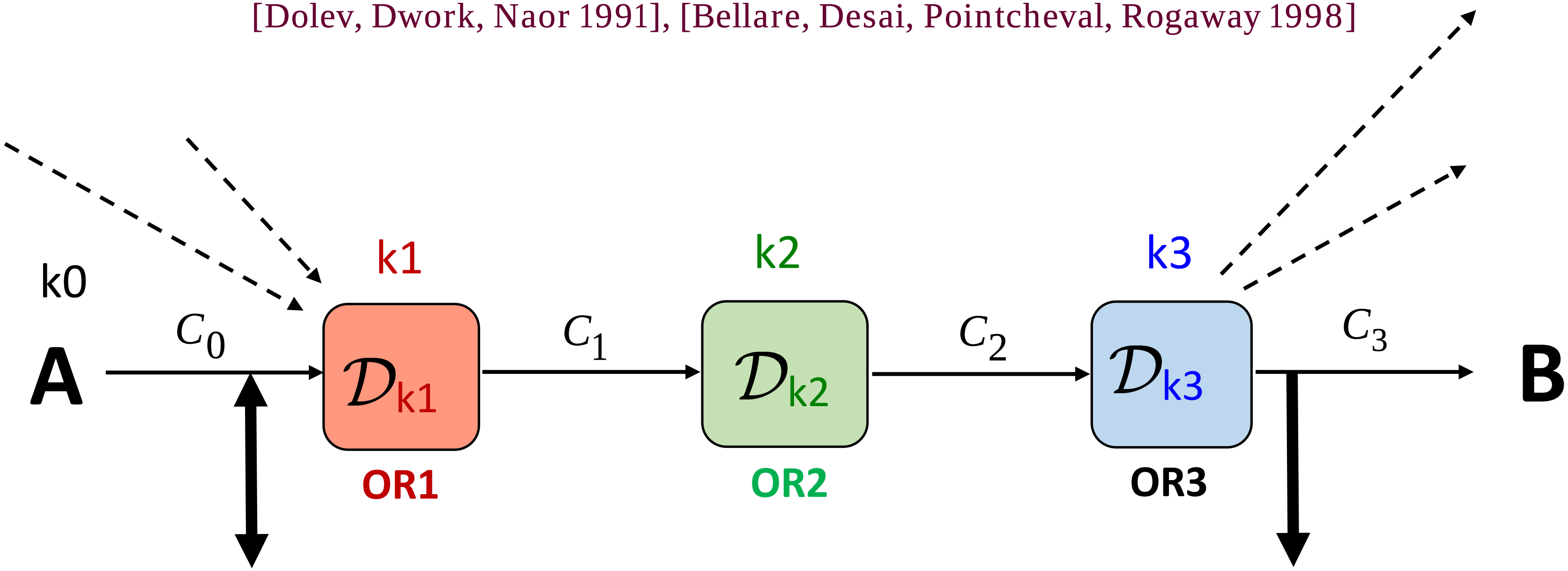
Alternative: "**Eager authenticity**" might be preferred.

# Tagging attacks

[Goldschlag, Reed, Syverson 1996]
[Dingledine, Mathewson, Syverson 2004]
[Fu, Ling 2009] [Racoon23 2012]

**Confirmation attacks** that a particular flow into an entry node leaves at some particular exit node, based on the **malleability** of the encryption

[Dolev, Dwork, Naor 1991], [Bellare, Desai, Pointcheval, Rogaway 1998]




$\mathbb{A}$ exploits malleability of encryption scheme to *tag* a ciphertext, e.g., xor'ing it with some constant $\Delta$

$\mathbb{A}$ detects the mauled ciphertext, confirming the originator of this flow.

Excluded because AE $\Rightarrow$ nonmalleability $\Rightarrow$ no tagging attacks

# LBE is onion-AE secure

≈ Mathewson's Proposal 202 (Design 1, **Large Block Encryption**), 2012.
Proposal 261 is 202 with AEZ

$$C_0 = \mathbb{E}_{K_1}^{\mathbf{c_1\text{-hist}}}\left(\mathbb{E}_{K_2}^{\mathbf{c_2\text{-hist}}}\left(\mathbb{E}_{K_3}^{\mathbf{c_3\text{-hist}}}\left(M \,||\, \mathbf{0}\right)\right)\right)$$

$\mathbb{E}$ a **wideblock TBC**, eg
**AEZ**, **EME2**, **Farfalle**, **HHFHFH**

**Theorem** [informal]: From an adversary $\mathbb{A}$ that attacks LBE[$\mathbb{E}$] we construct an adversary $\mathbb{B}$ that breaks $\mathbb{E}$ as a PRP with comparable resources and advantage.

# Final remarks

Two major definitional variants for onion-AE, **eager** and **lazy** authenticity. Both can be defined with oracle silencing. Which notion is desired?

[Proposal 295: Tomer Ashur, Orr Dunkelman, Atul Lyukx 2018]. Onion-AE secure??

Does any of this matter for Tor?  I don't know.
But it's best when we build our protocols out of primitives that achieve strong, formalized security definitions.