Onion Encryption Revisited: Relations Among Security Notions

Daichong Chao[1], Liehuang Zhu[2], Dawei Xu[2], Tong Wu[3], Chuan Zhang[2], and Fuchun Guo[4]

[1] School of Computer Science and Technology, Beijing Institute of Technology, China
chaodaichong@163.com
[2] School of Cyberspace Science and Technology, Beijing Institute of Technology, China
liehuangz@bit.edu.cn,xudw@ccu.edu.cn,chuanz@bit.edu.cn
[3] School of Computer and Communication Engineering, University of Science and Technology Beijing, China
tongw@ustb.edu.cn
[4] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia
fuchun@uow.edu.au

Abstract. This paper compares the relative strengths of prominent security notions for onion encryption within the Tor setting, specifically focusing on *CircuitHiding* (EUROCRYPT 2018, an anonymity flavor notion) and *OnionAE* (PETS 2018, a stateful authenticated encryption flavor notion). Although both are state-of-the-art, Tor-specific notions, they have exhibited different definitional choices, along with variations in complexity and usability. By employing an indirect approach, we compare them using a set of onion layer-centric notions: IND$-CPA, IPR/IPR$^+$, and INT-sfCTXT, to compare with the two, respectively. Since the same notion set that implies *OnionAE* does not imply *CircuitHiding*, and vice versa, this leads to the conclusion that *OnionAE* and *CircuitHiding* are mutually separable. Therefore, neither notion fully expresses satisfactory security on its own. Importantly, IND$-CPA, IPR$^+$ (a stronger variant of IPR), and INT-sfCTXT collectively and strictly imply *OnionAE* and *CircuitHiding*. Given their onion layer-centric and thus simpler nature, this provides a practical approach to simultaneously satisfying *CircuitHiding* and *OnionAE*. While the formal treatment of (general) public-key onion routing has been relatively well-studied, formal treatment tailored to Tor remains insufficient, and thus our work narrows this gap.

Keywords: Onion encryption · Tor · Onion routing · Anonymity · Notion relationship

1 Introduction

The importance of onion encryption (OE) within the widely used anonymity network Tor [15] is fundamental. The existing different approaches to cryptographic treatment pertaining to OE have impeded a comprehensive understanding of the security essence in this specific context. In this paper, we compare two prominent security notions relating to OE in the Tor setting, clarifying their relations and thereby promoting understanding of OE security.

1.1 Tor and OE

Tor [15] is widely considered the most influential anonymity network today, consisting of over 7,000 volunteer-operated relay nodes that provide anonymity services to over two million daily users [30]. To achieve this, Tor employs onion routing technology [32], establishing a multi-hop secure channel–*circuit* formed by multiple intermediate relays, thereby hiding the communication relationship between a Tor browser (sender) and its destination from a local observer. For onion routing, Tor employs OE, which involves multiple isomorphic layers of symmetric encryption in an iterative manner. In a circuit, the relay nodes involved are also called *onion routers* (OR); the sender is called *onion proxy* (OP); the first onion router is often called the *entry* node, and the last onion router is often called the *exit* node.

Regarding OE, its working details and current cryptographic choice merit more attention. An OP shares a symmetric key with each OR and performs multiple layers of encryption successively for a message. The output ciphertext–an *onion*, which is encapsulated into a fixed-size data unit–*cell*, is then passed along to the remaining ORs in the designated order, with each OR stripping off one layer of encryption. Finally, at the *exit* node, an integrity check is performed to verify the authentication of the incoming cell. The current OE scheme adheres to the MAC-then-encrypt paradigm. It involves the computation of a four-byte running SHA1 digest, followed by three isomorphic layers of 128-bit AES counter-mode encryption. The running state and the counter value for AES encryption collectively form the state of the current OE scheme, rendering it stateful. (For more detailed information on Tor and OE,

refer to [14].) It is worth mentioning that the current OE scheme is susceptible to *tagging attacks* [16,31], which leverage the malleability of AES counter mode to comprise a circuit's unlinkability (thus a user's anonymity). The Tor team has emphasized the defense against tagging attacks [25], and a series of OE proposals [23,24,4,12] have been put forward.

On August 15, 2023, the Tor team updated the status of proposal 308 [12], marking it as "superseded." [29] This indicates that the Tor team is cautious about choosing an alternative OE scheme, and they are eager to have a well-designed one. Therefore, researching the security foundation of OE is crucial.

1.2 *CircuitHiding* and *OnionAE*

CircuitHiding [13] and *OnionAE* [35] are two Tor-specific notions, both of them designed to mitigate crypto tagging attacks. The former adopts the anonymity flavor, while the latter follows the stateful authenticated encryption (AE) flavor. *CircuitHiding* and *OnionAE* embrace different philosophical perspectives and setting-oriented considerations, coupled with differences in complexity and usability.

In [13], Degabriele and Stam provide an anonymity-flavor notion called *CircuitHiding*. This notion characterizes the ability of an OE scheme to preserve anonymity against a static-corruption network adversary $\mathcal{A}_2$, who may corrupt multiple ORs (thus possessing their corresponding secret information) and introduce cryptographic deviations adaptively on these nodes. In the notion's game, two sufficiently general network topologies, $\mathcal{W}_0$ and $\mathcal{W}_1$, are specified by another adversary $\mathcal{A}_1$, based on the relevant validation rules specified by predicate VALID. Each network topology comprises multiple circuits (created by multiple users), with a portion of ORs being corrupted by the adversary $\mathcal{A}_2$. $\mathcal{A}_2$ is provided with two types of oracles: an encryption oracle denoted as ENC, used for outputting cells that it can manipulate (e.g., tagging, replaying, reordering, inserting) at a corrupt OR at the front of the circuit, and a network oracle denoted as NET, used for outputting cells that it can obtain at a corrupt OR at the back of the circuit. $\mathcal{A}_2$'s goal is to guess the actual interacting network topology, with the corresponding advantage expected to be negligible. Given the intricate nature of the network topology, *CircuitHiding* provides detailed descriptions of various aspects of the Tor setting, including topology (circuit) bookkeeping, stateful circuit building, and cell routing. Besides, *CircuitHiding* incorporates some crucial technical tricks to ensure meaningful cryptographic treatment. However, these factors lead to an increase in the complexity of the notion. To some extent, the complexity of this notion trades off its usability for one's intuition[5].

Definition 1 (*CircuitHiding* advantage, informal). *The CircuitHiding advantage of adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against an OE scheme* Π *is defined by*

$$\boldsymbol{Adv}_{\mathcal{A},\Pi}^{CircuitHiding} = Pr[\mathcal{A}_2^{\mathcal{W}_1} = 1] - Pr[\mathcal{A}_2^{\mathcal{W}_0} = 1] \tag{1}$$

We direct readers to Appendix A.1 for a detailed/formal description of the *CircuitHiding* experiment.

Another notion by Rogaway and Zhang [35], *OnionAE*, extends the conventional AE notion to the Tor setting, highlighting the AE requirements of a single circuit. At first glance, *OnionAE* draws inspiration from work [34] for nonce-based authenticated encryption, which employs a single game that encompasses both confidentiality and authentication. More specifically, *OnionAE* specifies an indistinguishability-based game with two possible worlds, G_0 and G_1. G_0 is the real world instantiated with a practical OE scheme, providing an encryption oracle (denoted as Enc) for the OP and a decryption oracle (denoted as Dec) for each OR[6]. G_1 is the ideal world that explicitly defines the expected properties of the encryption and decryption oracles of a circuit. Very importantly, the ideal world G_1 highlights that the intermediate decryption oracles provide independent randomness (compared to the encryption oracle) and the last decryption oracle outputs consistent authentication failure for the exit node (since the circuit goes out-of-sync). Anyway, compared to the network-level anonymity formalism in *CircuitHiding*, *OnionAE*'s single-circuit formalism appears to be more simple and easy to use[7]. However, the differences in their definitional choice have also blurred an understanding of whether there is a connection between the two.

[5] A classical and similar example in cryptography is semantic security for privacy, which has a strong intuitive appeal but weak usability.

[6] Reference [35] mentions that there is only one decryption oracle that takes an OR index as an input parameter. For the convenience of this paper, we refer to each OR as having a separate decryption oracle.

[7] In our opinion, *OnionAE* is still somewhat hard to work with, mainly because the authentication property for the exit node and the randomness property for the intermediate decryption oracles are coupled together via the out-of-sync (unsilenced) concept of a circuit. See details in Appendix A.2.

Definition 2 (*OnionAE* advantage, informal). *The OnionAE advantage of adversary $\mathcal{A}$ against an OE scheme Π is defined by*

$$\boldsymbol{Adv}_{\mathcal{A},\Pi}^{OnionAE} = Pr[\mathcal{A}^{\boldsymbol{G}_1} = 1] - Pr[\mathcal{A}^{\boldsymbol{G}_0} = 1] \tag{2}$$

We direct readers to Appendix A.2 for a detailed/formal description of the *OnionAE* experiment.

1.3 Our contributions

We provide a number of novel results to understand better *CircuitHiding* and *OnionAE*. Interestingly, we show that *OnionAE* and *CircuitHiding* are mutually separable. However, IND$-CPA (indistinguishability from random bits under chosen-plaintext attack), IPR$^+$ (enhanced independent pseudorandomness of the intermediate nodes), and INT-sfCTXT (stateful ciphertext integrity) collectively imply these two. Before delving further into these results, we first explain our methodology, which is directly related to them.

Methodology. To compare *OnionAE* and *CircuitHiding*, an obvious obstacle is that they involve different adversary types. The former deals with an adversary with corruption power (possessing secret information about corrupted nodes), while the latter deals with an ordinary adversary without corruption power. If *OnionAE* implies *CircuitHiding*, an explicit black-box reduction is infeasible (because *OnionAE* adversaries do not know how to simulate *CircuitHiding* adversaries). Therefore, we employ an *indirect* approach rather than directly providing corresponding reductions or counterexamples. Especially, we employ a set of onion layer-centric notions, IND$-CPA, IPR/IPR$^+$ (IPR: independent pseudorandomness of the intermediate nodes), and INT-sfCTXT, to compare with them, respectively. The deeper reason behind this approach is that the implementation of any security notion for OE ultimately depends on the security requirements for individual onion layers. By respectively identifying a suitable set of onion layer-centric notions implying *CircuitHiding* and *OnionAE*, we may also potentially determine the relationship between the two. This approach *not only* accomplishes the comparison *but also* precisely describes the relationship between them[8].

Among the above notions, IND$-CPA [33,7] is a randomness-flavor variant of IND-CPA. IPR/IPR$^+$ is a Tor-specific notion introduced in this paper, saying that for the outer $n-1$ (n: the number of OR nodes in a circuit) onion layers, decryptions of out-of-sync ciphertexts on the decryption side are indistinguishable from uniformly random strings, regardless of the inputs and outputs on the encryption side. Both IPR and IPR$^+$ aim at stymying tagging attacks, with the latter being a stronger variant of the former in terms of randomness. Additionally, INT-sfCTXT [6] is a conventional stateful AE notion for ciphertext integrity, roughly stating that when an adversary sends an illegal ciphertext to the decryption side, any subsequent ciphertext it sends will encounter authentication failure. IND$-CPA is applicable to all onion layers, IPR/IPR$^+$ is applicable to the outer $n-1$ onion layers, while INT-sfCTXT is only applicable to the innermost onion layer (consistent with Tor's end-to-end authentication). To prove that *OnionAE* and *CircuitHiding* are mutually separable, it is sufficient to show that the same notion set that implies *OnionAE* does not imply *CircuitHiding*, and vice versa.

Relations. The relations can first be presented from two directions.

In the direction towards the separation "*OnionAE* $\not\Rightarrow$ *CircuitHiding*", Theorem 1 demonstrates that "IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT" (read "$\wedge$" as "and") implies *OnionAE*; however, Theorem 4 shows that "IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT" (via a counterexample) does not imply *CircuitHiding*. Therefore, this separation holds.

In the direction towards the separation "*CircuitHiding* $\not\Rightarrow$ *OnionAE*", Theorem 2 establishes that *OnionAE* implies INT-sfCTXT, indicating that INT-sfCTXT is a necessary property of *OnionAE*; Theorem 3 indicates that IND$-CPA $\wedge$ IPR$^+$ implies *CircuitHiding*; however, Theorem 5 shows that "IND$-CPA $\wedge$ IPR$^+$" (via a counterexample not satisfying INT-sfCTXT) does not imply *OnionAE*. Therefore, this separation holds also.

Overall, based on Theorem 3, we infer that "IND$-CPA $\wedge$ IPR$^+$ $\wedge$ INT-sfCTXT" implies *CircuitHiding*; based on Theorem 1, we deduce that "IND$-CPA $\wedge$ IPR$^+$ $\wedge$ INT-sfCTXT" implies *OnionAE*, given that IPR$^+$ is a stronger variant of IPR in terms of randomness. IND$-CPA, IPR$^+$, and INT-sfCTXT collectively and strictly imply both *OnionAE* and *CircuitHiding*.

Implications for Design and Evaluation. Since neither *CircuitHiding* nor *OnionAE* fully expresses satisfactory security of OE on its own, and the new notion set "IND$-CPA $\wedge$ IPR$^+$ $\wedge$ INT-sfCTXT" is much simpler, a convenient approach in OE's design and evaluation is to use this notion set to provide the necessary security. For an experienced cryptographer, it might be possible to use this notion set to

[8] Therefore, from this perspective, our work differs significantly from most cryptographic definitional works.

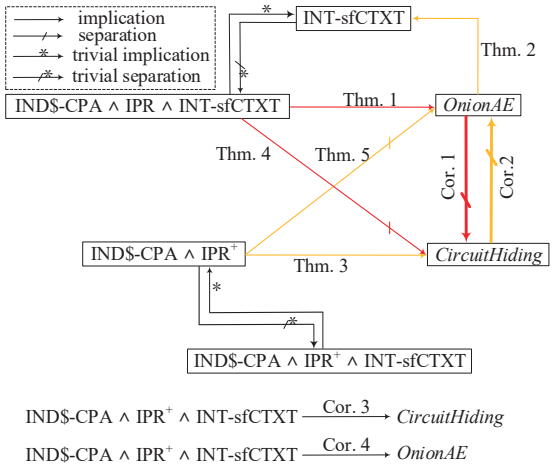

Fig. 1. Relationships among *OnionAE*, *CircuitHiding*, involving IND$-CPA, IPR/IPR$^+$, and INT-sfCTXT. Normal arrows are implications established; barred arrows are separations; * denotes a trivial result. The red indicates the relevant results for "*OnionAE* $\not\Rightarrow$ *CircuitHiding*," while the orange represents the relevant results for "*CircuitHiding* $\not\Rightarrow$ *OnionAE*". Remarkably, *OnionAE* and *CircuitHiding* are mutually separable. Thus, neither sufficiently expresses the security of OE.

quickly check the security of an OE scheme, avoiding costly fixes or security breaches post-deployment. Moreover, since IND$-CPA, IPR$^+$, and INT-sfCTXT are all onion layer-centric, we may consider the design of different onion layers independently, reducing the complexity during the design phase.

1.4 Related work

Camenisch and Lysyanskaya [8] define an ideal functionality within the Universal Composability (UC) framework for public-key onion routing and additionally introduce several game-based properties that collectively imply this ideal functionality. Consequently, they also provide a convenient proof strategy for securely realizing their ideal functionality. Leveraging this proof strategy, Danezis and Goldberg [11] proposed a popular cryptographic packet format, Sphinx, which has been used in multiple onion routing and mix networks [9,10,28,37]. However, Kuhn et al. [20] identified flaws in the proof strategy and introduced new game-based properties that imply the ideal functionality as originally defined by Camenisch and Lysyanskaya. Subsequently, both Ando and Lysyanskaya [1] and Kuhn et al. [21] extended the unified ideal functionality and the corresponding game-based properties to handle replies. Recently, Scherer et al. [38] adapted Kuhn et al.'s work [21] to the service model, where the receiver does not participate in the onion routing protocol (as in Tor), aiming to restore a suitable analytical framework for Sphinx to match its original design purpose.

One commonality among the works [8,20,1,21,38] is that they all propose (or use) an ideal functionality to elegantly describe security requirements and, more importantly, propose a set of easy-to-use game-based properties that imply the functionality. The notion set we introduced plays a similar role. However, due to subtle differences between public-key onion routing and the Tor setting, the results of works [8,20,1,21,38] may not be directly applicable to Tor. This is at least because: (1) Public-key onion routing focuses on a global adversary, while Tor considers a local adversary. (2) Public-key onion routing integrates routing and onion encryption into one ciphertext-oriented structure. Tor adopts a circuit-oriented architecture, delegating routing *largely* to an orthogonal circuit extend protocol. Due to the integration of routing, public-key onion routing ciphertexts additionally contain encrypted routing information and keys, significantly increasing ciphertext's size. In contrast, Tor's cells are fixed-sized and small to support low-latency communication. (3) Onion routers in public-key onion routing are stateless, while Tor operates in a stateful AE setting and needs to defend against replay, dropping, and reordering attacks at least. (4) The definition of malleability attacks in works [20,21,38] is stricter than that in Tor. Tor favors end-to-end authentication, taking authentication failures at the exit node due to midway manipulation as granted, whereas works [20,21] consider such scenarios insecure. Therefore, an independent treatment of Tor's security is desirable.

Backes et al. [5] give an ideal functionality called $\mathcal{F}_{\mathcal{OR}}$ within the UC framework, encompassing various life-cycle phases of Tor, such as circuit building, cell construction, and cell routing. As for OE, they identified an exact property known as predictably malleable onion secrecy (one of the conditions

for implying $\mathcal{F}_{\mathcal{OR}}$). This notion appears to resemble more of a characterization of the properties of the current OE scheme employed by Tor. It rightly captures the malleability of AES counter mode, thereby overlooking the need to defend against tagging attacks. Therefore, we omit it for comparison in this paper. In contrast, both Rogaway and Zhang [35] and Degabriele and Stam [13] have reconsidered security notions in the Tor setting. We were curious about the relationship between the two and conducted this study. In addition to *CircuitHiding*, Degabriele and Stam also present notions for confidentiality and integrity. We briefly mention them in Appendix C, showing that *OnionAE* is, in fact, stronger than them.

2 Preliminaries

2.1 Notations

We use lowercase letters to denote integers and strings, while bold lowercase letters denote one-dimensional vectors. For $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of all n-bit binary strings. We denote the set of all positive integers up to n by $[n] \coloneqq \{1, ..., n\}$. We write ε to denote the empty string and $\{0,1\}^*$ to denote the set of arbitrary-length strings. For $x \in \{0,1\}^*$, $|x|$ denotes its length. For $x, y \in \{0,1\}^*$, $x \oplus y$ denotes their bitwise XOR and $x \| y$ denotes their concatenation. For a non-negative integer z, $\langle z \rangle_l$ denotes the unsigned l-bit binary representation of z. For a one-dimensional vector $\mathbf{x}$, denote by $\mathbf{x}_k$ its k-th element, and by $|\mathbf{x}|$ its size. Additionally, [] denotes an empty list (vector). We use Euler-style uppercase letters to denote sets ($\mathcal{K}, \mathcal{M}, \mathcal{C}$, etc. representing different spaces) and calligraphic-style uppercase letters to denote algorithms ($\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{A}$, etc.) Commonly, $\mathcal{A}$ denotes an adversary which may be a randomized algorithm. In addition, we use sans-serif font to denote oracle predicates (Enc, Dec, sfVer, etc.). For a finite set $\mathcal{X}$, $x \leftarrow\!\$\ \mathcal{X}$ denotes the uniform and random sampling of x from $\mathcal{X}$. If $\mathcal{D}$ is a distribution over $\mathcal{X}$, we denote by $x \leftarrow\!\$\ \mathcal{D}$ the element x sampled from $\mathcal{X}$ according to $\mathcal{D}$. We use $\perp$ to indicate an authentication failure. For an oracle predicate, it returns $\lightning$ to indicate an output is suppressed as part of the notion's definition. Conventionally, we use a vector of nodes $\mathbf{p} \coloneqq [v_0, v_1, ..., v_n]$ to denote a circuit, where v_0 refers to the OP, v_1 refers to the first OR node, and v_n refers to the last OR node.

2.2 Syntax

As our treatment objects (notions) rely on different syntaxes, we list them and provide the corresponding interpretations, i.e., syntax commonality, to enable our comparisons.

Syntax in *CircuitHiding* Conceptually, an OE scheme $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D}, \overline{\mathcal{D}})$ includes four subroutines, with associated OR space $\mathcal{P} \subseteq \{0,1\}^*$, state space Ω, message space $\mathcal{M} \subseteq \{0,1\}^*$, cell space $\mathcal{C} \subseteq \{0,1\}^*$, encryption state space $\Sigma \subseteq \{0,1\}^*$, local circuit index space $\mathcal{W}$, and decryption state spaces $\mathcal{T} \subseteq \{0,1\}^*$ and $\overline{\mathcal{T}} \subseteq \{0,1\}^*$.

- $\mathcal{G} : \mathcal{P}^{n+1} \times \Omega \to \Omega \times \Sigma \times \mathcal{T}^n \times \overline{\mathcal{T}}^n$ is a stateful (probabilistic) algorithm that, for a circuit size $n \geq 3$, it takes a path $\mathbf{p} \in \mathcal{P}^{n+1}$ and an (unspecified) state variable $\omega \in \Omega$, updates the state variable ω, and outputs an initial encryption state $\sigma \in \Sigma$ for the OP ($\mathbf{p}[0]$) and two initial decryption state vectors ($\mathbf{t} \in \mathcal{T}^n$ and $\overline{\mathbf{t}} \in \overline{\mathcal{T}}^n$) for the OR nodes of the built circuit. Especially, $|\mathbf{t}| = |\overline{\mathbf{t}}| = n$, where each component in $\mathbf{t}$ and $\overline{\mathbf{t}}$ is added to the corresponding local vectors $\boldsymbol{\tau}$ and $\overline{\boldsymbol{\tau}}$. For example, for circuit $\mathbf{p} = [a, b, c, d]$, $\boldsymbol{\tau}_b$.append($\mathbf{t}[1]$), $\boldsymbol{\tau}_c$.append($\mathbf{t}[2]$), $\boldsymbol{\tau}_d$.append($\mathbf{t}[3]$), $\overline{\boldsymbol{\tau}}_b$.append($\overline{\mathbf{t}}[1]$), $\overline{\boldsymbol{\tau}}_c$.append($\overline{\mathbf{t}}[2]$), $\overline{\boldsymbol{\tau}}_d$.append($\overline{\mathbf{t}}[3]$).
- $\mathcal{E} : \mathcal{M} \times \Sigma \to \Sigma \times \mathcal{C} \times \mathcal{P}$ is a deterministic stateful algorithm used by an OP that takes in a plaintext $m \in \mathcal{M}$, an encryption state $\boldsymbol{\sigma}[w] \in \Sigma$ (where w represents the OP's local circuit index). It updates the encryption state $\boldsymbol{\sigma}[w]$ and outputs a cell $c \in \mathcal{C}$, the first OR $d \in \mathcal{P}$ in the circuit.
- $\mathcal{D} : \mathcal{T}^* \times \mathcal{C} \times \mathcal{P} \to \mathcal{W} \cup \{\perp\}$ is a deterministic algorithm used by an OR. It takes in a decryption state vector $\boldsymbol{\tau} \in \mathcal{T}^*$, an input cell $c \in \mathcal{C}$, the source OR $s \in \mathcal{P}$, and returns a local circuit index $w \in \mathcal{W}$ for determining the actual decryption state (i.e., $\overline{\boldsymbol{\tau}}[w] \in \overline{\mathcal{T}}$) or a special symbol $\perp$ which indicates a failure event to associate the cell c to a circuit.
- $\overline{\mathcal{D}} : \overline{\mathcal{T}} \times \mathcal{C} \times \mathcal{P} \to \overline{\mathcal{T}} \times (\mathcal{M} \cup \mathcal{C} \cup \{\perp\}) \times (\mathcal{P} \cup \{\oslash\})$ is a deterministic stateful algorithm used by an OR. It takes in a decryption state $\overline{\boldsymbol{\tau}}[w] \in \overline{\mathcal{T}}$, a cell $c \in \mathcal{C}$, and the source OR $s \in \mathcal{P}$, and updates $\overline{\boldsymbol{\tau}}[w]$, outputs a string $m \in (\mathcal{M} \cup \mathcal{C})$ (cell or decrypted message, depending on OR's location) or a special symbol $\perp$ indicating a decryption failure, and returns the next OR $d \in \mathcal{P}$ or a special symbol $\oslash$ indicating the conceptual end of a circuit.

Remark 1 (Routing behavior in CircuitHiding's syntax). A remarkable part in *CircuitHiding*'s syntax is that, besides OE, it considers routing behavior from an intuitive perspective, thereby reflecting some Tor-specific routing design choices and matching its own anonymity-oriented formalism. In terms of the treatment object, the focus is on the cell, which includes both the cell header and payload (onion) components. The payload portion, also commonly referred to as the onion, is derived from the original plaintext through onion encryption. The header portion is composed of a four-byte circuit identifier and a single-byte command field. The former determines which circuit a cell is associated with, and the latter indicates what action should be taken with the cell's payload (e.g., whether circuit extending or cell's relaying). A notable example of an attack that compromises anonymity by manipulating the header portion is the relay early attack [2], and reference [13] provides another example (see [13], Section 6.3). As for encryption, $\mathcal{E}$ outputs the first OR explicitly. As for decryption, the two-stage model $(\mathcal{D}, \overline{\mathcal{D}})$ formulates a clear division of responsibilities: the former for figuring out the relevant circuit and the latter for the actual onion decryption. As emphasized in reference [13], "Our split in two stages, coupled with the restrictions on how the state looks and can be affected, guarantees that the processing of cells for one circuit cannot unduly influence the later processing of a cell associated to a different circuit," this approach provides a separation of the cell's routing and decryption state's updating, thus facilitating the independence of individual circuits.

Remark 2 (Statefulness of $\mathcal{G}$). The reason for $\mathcal{G}$ being stateful is due to updating ω. Although ω is unspecified in the syntax section of [13], it is later described as a set of triples recording first-hop information for the description of proposal 261 [24]. Therefore, we may treat ω as an environment variable recording the available circuits in an OP.

Syntax in *OnionAE* An OE scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, with associated message space $\mathcal{M} \subseteq \{0,1\}^*$, ciphertext (onion) space $\mathcal{C} \subseteq \{0,1\}^*$, key space $\mathcal{K} \subseteq \{0,1\}^*$, encryption state space Σ, and decryption state space $\mathcal{S} \subseteq \{0,1\}^*$ (conventionally, $\mathcal{S} = \Sigma$).

- $\mathcal{K} : \mathbb{N} \to \mathcal{K}^*$ is a probabilistic algorithm that, given a circuit size $n \geq 3$, outputs a list of $n+1$ strings $(k_0, k_1, k_2, ..., k_n)$ in the key space $\mathcal{K}$ (k_0 for encryption and the rest for decryption).
- $\mathcal{E} : \mathcal{K} \times \mathcal{M} \times \Sigma \to \mathcal{C} \times \Sigma$ is a deterministic algorithm that takes in the encryption key $k_0 \in \mathcal{K}$, a plaintext $m \in \mathcal{M}$, an encryption state $\sigma \in \Sigma$. It outputs a ciphertext $c \in \mathcal{C}$ and an updated encryption state σ.
- $\mathcal{D} : \mathcal{K} \times \mathcal{C} \times \mathcal{S} \to (\mathcal{M} \cup \mathcal{C} \cup \{\perp\}) \times \mathcal{S}$ is a deterministic algorithm. It takes in a decryption key $k \in \mathcal{K}$, an input ciphertext $c \in \mathcal{C}$, a decryption state $s \in \mathcal{S}$. It outputs an updated decryption state s, a decrypted string $m \in \mathcal{M} \cup \mathcal{C}$ (onion or message, depending on the OR's location) or a special symbol $\perp$ indicating a decryption failure.

Syntax commonality between *CircuitHiding* and *OnionAE* At first glance, *CircuitHiding* and *OnionAE* correspond to different syntaxes, thus putting their notion comparison in an incomplete position; yet, actually, they share a high commonality from another perspective.

Deliberately or by chance, their differences are mainly due to different understandings of which components belong to OE and different note-taking habits. Obviously, the syntax in *OnionAE* is quite succinct, listing the essentials for OE. The complexity in *CircuitHiding* is mainly caused by considering routing behavior additionally, although the corresponding abstraction is very intuitive and concise. One distinction between them is that *CircuitHiding* focuses on the cell, while *OnionAE* only focuses on the onion. As mentioned in Remark 1, the two-stage model $(\mathcal{D}, \overline{\mathcal{D}})$ views figuring out the relevant circuit as an independent choice. Besides, both $\mathcal{E}$ and $\overline{\mathcal{D}}$ consider outputting the relevant OR to be forwarded to. Although reference [13] emphasizes that "On the one hand, it reflects practical protocol designs such as Tor, without being overly prescriptive on quite how routing has to work", in the Tor network, when an OR knows which circuit identifier a cell comes from, it does know where the cell goes to, regardless of the actual decryption. Hence, the next OR $d \in \mathcal{P}$ for $\overline{\mathcal{D}}$ can be migrated to $\mathcal{D}$, without introducing any misunderstanding of OE. Furthermore, the differences between $\mathcal{G}$ and $\mathcal{K}$ are mainly caused by different note-taking habits, like state vector versus individual state, considering an environment variable (ω) versus not considering such a variable, treating n as a conventional parameter versus treating it as an input variable, considering encryption/decryption states in $\mathcal{G}$ beforehand versus providing them in $\mathcal{E}$ and $\mathcal{D}$ directly, and treating the generated keys as part of encryption/decryption states versus listing them in $\mathcal{E}$ and $\mathcal{D}$ explicitly. Especially, we believe the omission of encryption/decryption states in $\mathcal{K}$ for *OnionAE* is likely accidental, since they actually appear in $\mathcal{E}$ and $\mathcal{D}$.

Hence, by restricting OE to consider only the onion portion and excluding the cell header, and aligning those different note-taking habits, the syntaxes of *CircuitHiding* and *OnionAE* are essentially identical.

Our adopted syntax In spirit, we follow the syntax of *OnionAE*. We only focus on the common onion portion; for *CircuitHiding*, we no longer consider the cell header. This allows for a fair comparison of the two, avoiding meaningless discussions. As demonstrated by the example of header manipulation in reference [13], even if an OE scheme is *OnionAE* secure in its onion portion, we can still compromise *CircuitHiding* as long as its cell header is not designed with sufficient security. This separation result is somewhat redundant and holds little significance for the community. Perhaps more importantly, focusing on the onion portion encourages us to accurately reveal the security essence of OE (in terms of the onion portion). This is helpful for guiding the design and evaluation of OE schemes. So, unless otherwise specified in the subsequent text, ciphertext refers to onion. Yet, for elegance and greater concreteness, and for potential needs, we have made minor adjustments and expansions to the syntax of *OnionAE*.

Firstly, regarding $\mathcal{K}$, we adjust it to $\mathcal{K} : \mathbb{N} \to \mathcal{K}^* \times \Sigma^*$. It outputs a list of $2n+2$ strings $(k_0, k_1, ..., k_n, s_0, s_1, ..., s_n)$, where the first half belongs to the key space $\mathcal{K}$, and the second half belongs to the state space Σ (k_0, s_0 for encryption and the rest for decryption).

Furthermore, we decompose $\mathcal{E}$ and $\mathcal{D}$ into $(\overline{\mathcal{E}}_1, ..., \overline{\mathcal{E}}_n)$ and $(\overline{\mathcal{D}}_1, ..., \overline{\mathcal{D}}_n)$ respectively, where $\overline{\mathcal{E}}_1$ is the encryption algorithm for the outermost onion layer, and so forth, with $\overline{\mathcal{E}}_n$ being the algorithm for the innermost onion layer. $(\overline{\mathcal{D}}_1, ..., \overline{\mathcal{D}}_n)$ follows a similar interpretation. Formally, $\overline{\mathcal{E}}_i : \mathcal{K} \times \mathcal{C} \times \Sigma \to \mathcal{C} \times \Sigma$ $(i \in [n-1])$ and $\overline{\mathcal{E}}_n : \mathcal{K} \times \mathcal{M} \times \Sigma \to \mathcal{C} \times \Sigma$. And so forth, $\overline{\mathcal{D}}_i : \mathcal{K} \times \mathcal{C} \times \Sigma \to \mathcal{C} \times \Sigma$ $(i \in [n-1])$ and $\overline{\mathcal{D}}_n : \mathcal{K} \times \mathcal{C} \times \Sigma \to (\mathcal{M} \cup \{\perp\}) \times \Sigma$. This aligns with Tor's end-to-end authentication, while the intermediate onion layers solely involve encryption and decryption (we rule out the leaky-pipe feature). In Section 3, we will use these refined notations to formalize our onion layer-centric cryptographic experiments (Definition 3, 4, and 5).

2.3 Conventions

We make a convention that all primitives and security experiments in our paper share a common security parameter r. Hence, the terminology "probabilistic polynomial time (PPT)" and "negligible" is relative to r. We leave out r for convenience hereinafter. We assume that $\mathcal{M} = \{0,1\}^{n_1}$ and $\mathcal{C} = \{0,1\}^{n_2}$ with $n_1 \leq n_2$ (n_1 and n_2 are polynomials in r.) In addition, n is the number of OR nodes in a circuit. We use $\mathbf{c}^{(i)}$ and $\overline{\mathbf{c}}^{(i)}$ to denote the "correct" ciphertext sequence and the actual received ciphertext sequence at the i-th OR, respectively.

3 Newly Introduced Notions for Comparison

Firstly, we describe the newly introduced notions IND$-CPA, IPR/IPR$^+$, INT-sfCTXT. For IND$-CPA, we provide a re-formalization to align it with the Tor setting, facilitating its comparison with IPR/IPR$^+$. Clearly, IPR$^+$ is a stronger variant of IPR in terms of providing randomness when the circuit goes out-of-sync. INT-sfCTXT is directly adopted from Bellare et al. [6].

Definition 3 (IND$-CPA advantage). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OE scheme. For any i-th ($i \in [n]$) onion layer of Π and $b \in \{0,1\}$, the advantage of an adversary $\mathcal{A}$ playing the IND$-CPA security game $\boldsymbol{Exp}^{IND\$\text{-}CPA\text{-}(i,b)}_{\Pi,\mathcal{A}}$, as described in Figure 2, is defined by:*

$$\boldsymbol{Adv}^{IND\$\text{-}CPA\text{-}(i)}_{\mathcal{A},\Pi} = Pr\left[\boldsymbol{Exp}^{IND\$\text{-}CPA\text{-}(i,1)}_{\Pi,\mathcal{A}} = 1\right] - Pr\left[\boldsymbol{Exp}^{IND\$\text{-}CPA\text{-}(i,0)}_{\Pi,\mathcal{A}} = 1\right] \quad (3)$$

IND$-CPA ensures a very basic level of privacy for each onion layer. It says that each onion layer provides fresh randomness upon each encryption, making the resulting ciphertext indistinguishable from a uniformly random string. Distinguished from the traditional definition of IND$-CPA [33,7], in Definition 3, an adversary $\mathcal{A}$ is provided with two oracles: an actual encryption oracle Enc and an additional helper oracle Proc, with the requirement that the input to Enc is the output of Proc. Essentially, Proc prepares the intermediate ciphertexts (the encryption result of the previous onion layer) needed for Enc as inputs, and the inputs to Enc are no longer wholly adaptive. This adjustment considers the inherent characteristics of OE schemes. On the one hand, the input to the Enc oracle is inherently the encryption result of the previous onion layer. On the other hand, in some OE schemes[9] (such as proposal 295 [4] and proposal 308 [12] using a GCM-RUP variant), the encryption nonce is included in the output of the previous layer, making it part of the intermediate ciphertext input to the next onion layer; then the encryption

[9] Appendix D provides an introduction to the Tor proposals on OE [23,24,4,12], offering further insight into the current research on OE schemes.

nonce serve as the initial counter of AES counter mode computation for the remaining part of the input intermediate ciphertext. In such cases, achieving IND$-CPA as defined traditionally is not feasible. To provide meaningful treatment, appropriate weakening is necessary here.

$\mathbf{Exp}^{\text{IND\$-CPA-(i,b)}}_{\Pi,\mathcal{A}}$

$(k_0, k_1, ..., k_n, s_0, s_1, ..., s_n) \leftarrow \mathcal{K}$
$\mathbf{c}^{(i+1)} \leftarrow [], p \leftarrow 1, u \leftarrow 1$
$b' \leftarrow \mathcal{A}^{\text{Proc}(\cdot,i),\text{Enc}(\cdot,i,b)}$
return b'

$\text{Enc}(m, i, b)$

require $u < p$ **and** $m = \mathbf{c}_u^{(i+1)}$
$(c, s_i) \leftarrow \overline{\mathcal{E}}_i(k_i, m, s_i)$
if $b = 1$
  $c \leftarrow\$ \{0,1\}^{n_2}$
$u \leftarrow u + 1$
return c

$\text{Proc}(m, i)$

require $|m| = n_1$
if $i = n$
  $\mathbf{c}_p^{(i+1)} \leftarrow m$
else
  $c_{n+1} \leftarrow m$
  for $j \leftarrow n$ **downto** $i + 1$ **do**
    $(c_j, s_j) \leftarrow \overline{\mathcal{E}}_j(k_j, c_{j+1}, s_j)$
  $\mathbf{c}_p^{(i+1)} \leftarrow c_{i+1}$
$p \leftarrow p + 1$
return $\mathbf{c}_{p-1}^{(i+1)}$

Fig. 2. Experiment of IND$-CPA. Adversary $\mathcal{A}$ is provided with two oracles: $\mathsf{Enc}(\cdot, i, b)$ and $\mathsf{Proc}(\cdot, i)$. $\mathsf{Proc}(\cdot, i)$ takes an original plaintext as input, which undergoes $n - i$ iterations of encryption, generating an intermediate ciphertext that is recorded in the ciphertext sequence $\mathbf{c}^{(i+1)}$. $\mathsf{Enc}(\cdot, i, b)$ takes an in-order ciphertext from $\mathbf{c}^{(i+1)}$ as input, which undergoes the encryption of the i-th onion layer, producing an output, either the final ciphertext or a uniformly random string, depending on the challenge bit b. p and u are counter values used for in-order bookkeeping.

Definition 4 (IPR advantage). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OE scheme. For any i-th ($i \in [n-1]$) onion layer of Π and $b \in \{0,1\}$, the advantage of an adversary $\mathcal{A}$ playing the IPR security game $\boldsymbol{Exp}^{IPR\text{-}(i,b)}_{\Pi,\mathcal{A}}$, as described in Figure 3 (with the boxed parts), is defined by:*

$$\boldsymbol{Adv}^{IPR\text{-}(i)}_{\mathcal{A},\Pi} = Pr\left[\boldsymbol{Exp}^{IPR\text{-}(i,1)}_{\Pi,\mathcal{A}} = (*, 0)\right] + Pr\left[\boldsymbol{Exp}^{IPR\text{-}(i,0)}_{\Pi,\mathcal{A}} = (*, 0)\right] + \left|Pr\left[\boldsymbol{Exp}^{IPR\text{-}(i,1)}_{\Pi,\mathcal{A}} = (1, *)\right] - Pr\left[\boldsymbol{Exp}^{IPR\text{-}(i,0)}_{\Pi,\mathcal{A}} = (1, *)\right]\right| \quad (4)$$

IPR aims to provide an additional dimension of randomness beyond IND$-CPA to stymie cryptographic tagging attacks. It says that when an adversary introduces an illegal ciphertext into the circuit, causing the circuit (onion layer) to go out-of-sync[10], in the *vast majority* of cases, the output on the decryption side is indistinguishable from an independently and uniformly random string, regardless of the inputs and outputs on the encryption side[11]. The *only exception* is that when the input on the decryption side equals the output on the encryption side, the output on the decryption side remains consistent with the input on the encryption side. The former case, which involves random output, is aimed at stymying tagging attacks from a common perspective: when an adversary corrupts the precursor node of a relay node and introduces ciphertext manipulation (e.g., tagging), the uniform randomness of decryption will result in receiving meaningless garbage at the successor node, preventing the adversary from gaining any useful information for correlation through any disparity between encryption and decryption. The latter case may represent a seemingly reasonable but actually incorrect understanding that stymying tagging attacks only requires achieving the former case, and there is no impact when the output on the decryption side equals the input on the encryption side. This viewpoint will be negated by the counterexample for Theorem 4 later, where a simple stateful method is applied, leading to a construction that, although satisfying IPR, does not satisfy *CircuitHiding*. In fact, the latter case is likely to occur for stateful encryption schemes when employing a relatively trivial stateful method, as is the case with the counterexample for

[10] In the stateful context, the concept of out-of-sync is of great importance. It is first introduced in the seminal work [6] to describe a situation where the ciphertext received by the decryption party does not match the ciphertext generated by the encryption party. Reference [13] extends it to the circuit context and formalizes it within *CircuitHiding*. Reference [35] uses the equivalent term "unsilenced" to describe out-of-sync within *OnionAE*. IPR/IPR$^+$ also formalizes it within itself.

[11] This is also why we named this notion 'independent pseudorandomness of the intermediate node' (IPR).

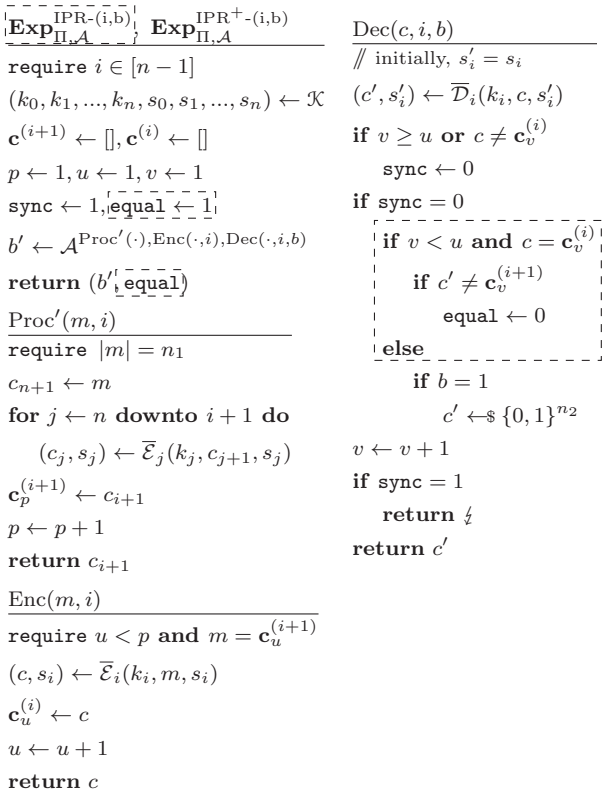

Fig. 3. Experiments of IPR and IPR$^+$. The IPR experiment has additional dashed rectangular boxes compared to the IPR$^+$ experiment. Adversary $\mathcal{A}$ is provided with three oracles: $\mathsf{Proc}'(\cdot, i)$, $\mathsf{Enc}(\cdot, i)$, and $\mathsf{Dec}(\cdot, i, b)$. $\mathsf{Proc}'(\cdot, i)$ takes an original plaintext as input, producing an intermediate ciphertext that is recorded in the ciphertext sequence $\mathbf{c}^{(i+1)}$. $\mathsf{Enc}(\cdot, i)$ takes an in-order ciphertext from $\mathbf{c}^{(i+1)}$ as input, generating an i-th onion layer encrypted ciphertext that is recorded in the ciphertext sequence $\mathbf{c}^{(i)}$. $\mathsf{Dec}(\cdot, i, b)$ takes an out-of-sync ciphertext as input and returns an output, either the decrypted ciphertext or a uniformly random string, which depends on the challenge bit b. The flag `sync` indicates whether the i-th onion layer goes out-of-sync. p, u, and v are counter values; the former two are used for in-order bookkeeping between $\mathsf{Proc}'(\cdot, i)$ and $\mathsf{Enc}(\cdot, i)$; the latter two (along with the ciphertext sequence $\mathbf{c}^{(i)}$) are used for out-of-sync bookkeeping (through setting `sync`). Solely for IPR, the outcome flag `equal` is introduced and Dec $(\cdot, i, b)$ embodies an additional dashed box for capturing the only exception case.

Theorem 4. Our other ongoing work indicates that proposal 295 [4] has similar issues. Therefore, the latter case is not entirely a result of our artificial design; it also reflects a practical oversight, or what can be termed as insecure practice.

Technically, in the IPR experiment, as shown in Figure 3, an adversary $\mathcal{A}$ is provided with three oracles: a helper oracle Proc', an actual encryption oracle Enc, and a decryption oracle Dec. Proc' serves as a helper oracle for Enc, preparing the needed intermediate ciphertexts for the latter. The encryption oracle Enc returns the actual encryption-side output for the i-th onion layer. The decryption oracle Dec suppresses those in-sync ciphertexts, and based on the challenge bit b and the input out-of-sync ciphertext c, determines the output. Specifically, when b equals 0, Dec returns the decryption of the out-of-sync ciphertext c for the i-th onion layer; when b equals 1, Dec returns a uniformly random string. In order to capture the latter case, namely that the output on the decryption side equals the input on the encryption side when the input on the decryption side is the output on the encryption side, a binary outcome flag *equal* is introduced. When the latter case occurs, Dec sets *equal* to 1. The IPR experiment returns *both* the guessing bit b' *and* the flag *equal*. Correspondingly, the IPR advantage consists of two parts: one part $(\Pr\left[\mathbf{Exp}^{\text{IPR-(i,1)}}_{\Pi,\mathcal{A}} = (*, 0)\right] + \Pr\left[\mathbf{Exp}^{\text{IPR-(i,0)}}_{\Pi,\mathcal{A}} = (*, 0)\right])$ quantifies the probability of *equal* being 1, while the other part $(\left|\Pr\left[\mathbf{Exp}^{\text{IPR-(i,1)}}_{\Pi,\mathcal{A}} = (1, *)\right] - \Pr\left[\mathbf{Exp}^{\text{IPR-(i,0)}}_{\Pi,\mathcal{A}} = (1, *)\right]\right|)$ quantifies the probability of distinguishing the decryptions of out-of-sync ciphertexts from uniformly random strings from a typical perspective.

Definition 5 (IPR$^+$ advantage). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OE scheme. For any i-th ($i \in [n-1]$) onion layer of Π and $b \in \{0, 1\}$, the advantage of an adversary $\mathcal{A}$ playing the IPR$^+$ security game $\boldsymbol{Exp}^{IPR^+\text{-}(i,b)}_{\Pi,\mathcal{A}}$,*

as described in Figure 3 (without the boxed parts), is defined by

$$\boldsymbol{Adv}^{IPR^+\text{-}(i)}_{\mathcal{A},\Pi} = Pr\left[\boldsymbol{Exp}^{IPR^+\text{-}(i,1)}_{\Pi,\mathcal{A}} = 1\right] - Pr\left[\boldsymbol{Exp}^{IPR^+\text{-}(i,0)}_{\Pi,\mathcal{A}} = 1\right] \tag{5}$$

IPR^+ is an enhanced version of IPR, removing the only exception case in the latter. IPR^+ says that when the circuit (onion layer) goes out-of-sync, *all*[12] the outputs on the decryption side remain indistinguishable from an independently and uniformly random string, regardless of the inputs and outputs on the encryption side. The adversary would find it difficult to gain any useful information from the decryption side. This is precisely a crucial step in defending against tagging attacks (this will be confirmed by Theorem 3 later). Technically, in the IPR^+ experiment, the dashed box in the Dec oracle is removed and only the guessing bit b' is returned.

Definition 6 (INT-sfCTXT advantage). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a stateful AE scheme. The advantage of an adversary $\mathcal{A}$ playing the INT-sfCTXT security game $\boldsymbol{Exp}^{INT\text{-}sfCTXT}_{\Pi,\mathcal{A}}$, as described in Figure 4, is defined by*

$$\boldsymbol{Adv}^{INT\text{-}sfCTXT}_{\mathcal{A},\Pi} = Pr\left[\boldsymbol{Exp}^{INT\text{-}sfCTXT}_{\Pi,\mathcal{A}} = 1\right] \tag{6}$$

INT-sfCTXT [6] says that when an adversary sends an illegal ciphertext to the last relay node, besides this (first) illegal ciphertext, any subsequent ciphertext it sends will encounter authentication failure. In this game, the adversary is provided with two oracles: an encryption oracle Enc and a decryption (verification) oracle sfVer. Enc produces the output on the encryption side, while sfVer returns the output on the decryption side. Specifically, sfVer only responds to out-of-sync ciphertexts, suppressing queries related to in-sync ciphertexts. The counters u, v, and the ciphertext sequence $\mathbf{c}$ are used for out-of-sync bookkeeping. INT-sfCTXT provides a robust characterization of ciphertext integrity in a stateful setting.

4 Notion Relationships

4.1 Preliminary theorems

Section 4.1 will provide one side of the coin (i.e., implication results) for the overall separation results. In Section 4.2, we will present the other side of the coin (i.e., separation results) required for the overall separation results.

$\underline{\mathbf{Exp}^{\text{INT-sfCTXT}}_{\Pi}(\mathcal{A})}$
$(k, \delta, \varrho) \leftarrow \mathcal{K}$
$u \leftarrow 0, v \leftarrow 0, \mathbf{c} \leftarrow []$
$\mathtt{sync} \leftarrow 1, \mathtt{win} \leftarrow 0$
$\mathcal{A}^{\text{Enc}(\cdot),\text{sfVer}(\cdot)}$
return win

$\underline{\text{Enc}(m)}$
$(c, \delta) \leftarrow \mathcal{E}(k, m, \delta)$
$u \leftarrow u + 1; \mathbf{c}_u \leftarrow c$
return c

$\underline{\text{sfVer}(c)}$
$v \leftarrow v + 1$
$(m, \varrho) \leftarrow \mathcal{D}(k, c, \varrho)$
if $v > u$ **or** $c \neq \mathbf{c}_v$
  then sync $\leftarrow 0$
if sync $= 0$ **and** $m \neq \perp$
  then win $\leftarrow 1$
if $m \neq \perp$
  then $m \leftarrow \lightning$
return m

Fig. 4. Experiment of INT-sfCTXT. Here, $\mathcal{E}$ and $\mathcal{D}$ actually represent $\overline{\mathcal{E}_n}$ and $\overline{\mathcal{D}_n}$, respectively.

To show IND\$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT $\Rightarrow$ *OnionAE*, we need Lemma 1, to show that when the *OnionAE* adversary introduces the first out-of-sync ciphertext in the real world G_0, it inevitably triggers the out-of-sync concept in INT-sfCTXT of the innermost onion layer. This allows us to substitute the authentication guarantee of *OnionAE* with INT-sfCTXT, enabling an IND\$-CPA or IPR adversary to simulate the *OnionAE* adversary when needed.

[12] When we say IPR^+ is a stronger variant of IPR, we refer to it in terms of randomness. IPR^+ does not imply IPR due to IPR's the only exception case. A reader needs to make an accurate conceptual understanding.

Lemma 1. *Let Π be an OE scheme that is IPR secure. For any OnionAE adversary $\mathcal{A}$ against Π, when it introduces the first out-of-sync ciphertext on the i-th Dec oracle ($i \leq n$) in the real world G_0, there exists an IPR-(i) adversary $\mathcal{B}$ such that: $\mathcal{A}$ violates the out-of-sync concept in INT-sfCTXT at the n-th Dec oracle with a probability of at most $(n-1)\cdot(\frac{1}{2^{n_2}} + \boldsymbol{Adv}^{IPR\text{-}(i)}_{\mathcal{B},\Pi}) + \frac{1}{2^{n_2}}$, where $\mathcal{B}$ introduces only one (i.e., the first caused by $\mathcal{A}$) out-of-sync ciphertext to its own $\mathsf{Dec}(\cdot, i, b)$ oracle (with $b = 0$).*

Proof sketch. The core of the proof lies in identifying the cases in which the *OnionAE* adversary can violate the out-of-sync concept in INT-sfCTXT. Given the adaptive nature of the *OnionAE* adversary, we thoroughly examine all possible scenarios in the real world G_0 to identify exceptional cases and derive upper bounds for their probabilities. Specifically, due to the unique position of the n-th Dec oracle, we split the analysis into two main cases: one concerning the first out-of-sync ciphertext targeting the n-th Dec oracle and another concerning the first out-of-sync ciphertext targeting the i-th Dec oracle ($i < n$).

In the first main case, there is only one sub-case in which $\mathcal{A}$ has the opportunity to succeed by relying on random guessing for the predetermined correct ciphertext. The probability of this occurring is bounded by $\frac{1}{2^{n_2}}$. As a representative sub-case of the second main case, and the one with the highest exceptional probability, the first out-of-sync (and modified) ciphertext targets the first Dec oracle. In this case, constrained by IPR, the adversary $\mathcal{A}$ has at most $(n-1)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{B},\Pi}) + \frac{1}{2^{n_2}}$ probability to violate the out-of-sync concept in INT-sfCTXT of the innermost onion layer. The reasoning is summarized as follows: If $\mathcal{A}$ directly queries the n-th Dec oracle with a guessing input, the probability is bounded by $\frac{1}{2^{n_2}}$. If $\mathcal{A}$ directly queries the i-th Dec oracle ($1 \leq i < n$) with a guessing input such that the output of the i-th Dec oracle equals the predetermined correct ciphertext, the probability is bounded by $\mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{B},\Pi} + \frac{1}{2^{n_2}}$. $\mathcal{A}$ has $n-1$ such opportunities.

A detailed proof for Lemma 1 is provided in Appendix E.1.

Theorem 1 (IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT $\Rightarrow$ *OnionAE*). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OE scheme. For any OnionAE adversary $\mathcal{A}$, making at most polynomial-bounded queries on Enc and Dec oracles, there exists an INT-sfCTXT adversary $\mathcal{B}$ using all these queries, an IND$-CPA adversary $\mathcal{C}$ using all the queries on Enc and the first $n-1$ Dec oracles, an IPR adversary $\mathcal{F}_1$ using all the queries on Enc and the first $n-1$ Dec oracles, and IPR adversary $\mathcal{F}_2$ using only one (i.e., the first caused by $\mathcal{A}$) out-of-sync query on the i-th Dec oracle (with i indicating the position associated with the first out-of-sync query), such that: $\boldsymbol{Adv}^{OnionAE}_{\mathcal{A},\Pi} \leq \boldsymbol{Adv}^{\mathsf{INT-sfCTXT}}_{\mathcal{B},\Pi} + \boldsymbol{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi} + \boldsymbol{Adv}^{\mathsf{IPR}}_{\mathcal{F}_1,\Pi} + (n-1)\cdot(\frac{1}{2^{n_2}} + \boldsymbol{Adv}^{IPR\text{-}(i)}_{\mathcal{F}_2,\Pi}) + \frac{1}{2^{n_2}}$.*

Proof sketch. The proof involves three rounds of game hopping. We first hop from the real-world G_0 in *OnionAE* to a scenario where, when the *OnionAE* adversary introduces the first out-of-sync ciphertext, it triggers the out-of-sync concept in INT-sfCTXT of the innermost onion layer. Using Lemma 1, the advantage gap for this hopping is bounded by $(n-1)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{F}_2,\Pi}) + \frac{1}{2^{n_2}}$.

We then hop to a new game where the last Dec oracle always returns $\perp$ for any received out-of-sync ciphertext. To bound the advantage gap for this hopping, we construct an INT-sfCTXT adversary $\mathcal{B}$ that uses all the queries made by $\mathcal{A}$ to break the INT-sfCTXT game. $\mathcal{B}$ utilizes the keys and initial state information of the outer $n-1$ onion layers to perfectly simulate the *OnionAE* game.

Next, we hop to another game where both the Enc oracle and the first $n-1$ Dec oracles always output a uniformly random string upon receiving a query. At this point, the final game becomes identical to the ideal-world G_1 in *OnionAE*. Since the last Dec oracle always returns $\perp$ upon receiving an out-of-sync ciphertext, we only need to show that the outputs of the Enc oracle and the first $n-1$ Dec oracles prior to the last hopping are computationally indistinguishable from a uniformly random string for the *OnionAE* adversary $\mathcal{A}$. If all onion layers satisfy IND$-CPA, and the outer $n-1$ onion layers satisfy the IPR property, the above conclusion holds. First, if the outermost onion layer satisfies IND$-CPA, then the outputs of the Enc oracle fulfill the required condition. For the i-th Dec oracle ($1 \leq i < n$), two cases need to be considered further. If the i-th Dec oracle receives a query that matches the ciphertext at the corresponding position in the "correct" ciphertext sequence $\mathbf{c}^{(i)}$, even if this Dec oracle has already received an out-of-sync ciphertext, the IPR property ensures that the oracle always outputs a ciphertext that is the encryption result of the $(i+1)$-th onion layer. If the i-th Dec oracle receives a query that does not match the ciphertext at the corresponding position in $\mathbf{c}^{(i)}$, the IPR property ensures that the oracle outputs a (pseudo)random string. Therefore, overall, the advantage gap for this hopping is bounded by the advantage of an IND$-CPA adversary $\mathcal{C}$ and the advantage of an IPR adversary $\mathcal{F}_1$. Both $\mathcal{C}$ and $\mathcal{F}_1$ use at most all the queries to the Enc oracle and the first $n-1$ Dec oracles.

A detailed proof for Theorem 1 is provided in Appendix E.2.

Theorem 2 (*OnionAE* $\Rightarrow$ INT-sfCTXT). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OE scheme. For any INT-sfCTXT adversary $\mathcal{A}$ of the innermost onion layer, making at most polynomial-bounded queries on Enc*

and sfVer *oracles, there exists an OnionAE adversary* $\mathcal{B}$ *using all these queries, such that:* $\boldsymbol{Adv}^{\mathsf{INT-sfCTXT}}_{\mathcal{A},\Pi} \leq \boldsymbol{Adv}^{OnionAE}_{\mathcal{B},\Pi}$.

Proof. The proof is intuitive. As a special case of *OnionAE*, the front $n-1$ nodes of the circuit remain in-sync, with only the queries to the last node causing the circuit to become out-of-sync. In this case, we construct adversary $\mathcal{B}$ for *OnionAE* by invoking adversary $\mathcal{A}$ to access the Enc and Dec oracles on the circuit. We utilize the output of the $(n-1)$-th Dec oracle of $\mathcal{B}$ as the response to adversary $\mathcal{A}$'s Enc oracle and the output of the last Dec oracle as the response to $\mathcal{A}$'s sfVer oracle. Adversary $\mathcal{B}$ provides a perfect simulation for adversary $\mathcal{A}$. When adversary $\mathcal{A}$ generates a valid ciphertext query to sfVer that decrypts correctly, adversary $\mathcal{B}$ breaks the *OnionAE* game. □

Theorem 3 (IND$-CPA $\wedge$ IPR$^+$ $\Rightarrow$ *CircuitHiding*). *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an OE scheme. For any CircuitHiding adversary* $\mathcal{A}$*, making* q_1 *queries to* ENC *and* q_2 *queries to* NET*, there exists an IPR*$^+$ *adversary* $\mathcal{B}_1$ *making at most* q_1 *queries to its* Proc′ *and* Enc *oracles and* q_2 *queries to its* Dec *oracle, an IPR*$^+$ *adversary* $\mathcal{B}_2$ *introducing only one out-of-sync ciphertext to its* Dec *oracle, and an IND$-CPA adversary* $\mathcal{C}$ *making at most* q_1 *queries to its* Proc *and* Enc *oracles, such that:* $\boldsymbol{Adv}^{CircuitHiding}_{\mathcal{A},\Pi} \leq 2l \cdot (\boldsymbol{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_1,\Pi} + \boldsymbol{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi} + (n_{max}-3) \cdot (\boldsymbol{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_2,\Pi} + \frac{1}{2^{n_2}}))$*, where* l *is the number of circuits with an honest proxy in the network topology* $\mathcal{W}_0$ *(or* $\mathcal{W}_1$*) and* n_{max} *is the maximum circuit length.*

Proof sketch. The proof involves three rounds of game hopping. Firstly, we further constrain the impact of the first out-of-sync vector query to the NET oracle, ensuring that this query not only causes the first OR of the honest middle segment in each circuit (that has an honest OP and two corrupt segments) to receive an out-of-sync ciphertext but also causes the out-of-sync concept to propagate iteratively to the last OR of the honest middle segment. For this hopping, the advantage gap is bounded using the identical-until-bad model. When an OR receives an out-of-sync ciphertext, its decryption output remains an out-of-sync ciphertext for the next OR. The probability of the bad event is bounded by $\mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_2,\Pi} + \frac{1}{2^{n_2}}$. There can be at most $n_{max}-3$ honest ORs before the last OR of the honest middle segment.

Next, we replace the decryption layer associated with the last OR of the honest middle segment by outputting a uniformly random string upon receiving an out-of-sync ciphertext, thus providing the same observed distribution at the second corrupt segment in the two worlds $\mathcal{W}_0$ and $\mathcal{W}_1$. For this hopping, we construct an IPR$^+$ adversary $\mathcal{B}_1$ on each altered circuit which invokes queries to the ENC and NET oracles to break the independent pseudorandomness of the replaced onion layer as per Definition 5. With a standard hybrid argument, the advantage gap is bounded by $2l \cdot \mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_1,\Pi}$.

Finally, we replace the encryption layer associated with the honest OR of the first entry edge of each circuit that has an honest OP (either one or two corrupt segments) by outputting a uniformly random string upon receiving an intermediate ciphertext. This operation will result in the same observed distribution at the first corrupt segment in the scope of those altered circuits. Here, we construct an IND$-CPA adversary $\mathcal{C}$ on each circuit that invokes the queries to the ENC oracle, breaking the pseudorandomness of the replaced onion layer as per Definition 3. Since the type of circuit that has a corrupt OP does not contribute to the distinguishing advantage, $\mathcal{A}$ has no advantage in distinguishing the two worlds in the final game.

A detailed proof for Theorem 3 is provided in Appendix E.3.

Remark 3 (Role differences among different onion layers). More precisely, in the proof of Theorem 3, we actually only rely on the fact that the onion layers corresponding to the honest ORs between the first and second corrupt segments (if any) satisfies IND$-CPA $\wedge$ IPR$^+$. This indicates that, for *CircuitHiding*, the security role is entirely borne by the onion layers corresponding to these honest ORs. Specifically, in the typical topology setting ($\mathcal{W}_0$, $\mathcal{W}_1$) in *CircuitHiding* as shown in Figure 5, where each topology contains only two circuits and each circuit consists of exactly three ORs, the security role is entirely fulfilled by the onion layer corresponding to the honest middle OR (v_5). As shown in Lemma 1, when the outer $n-1$ onion layers satisfy IPR, INT-sfCTXT replaces the authentication property of *OnionAE*. Therefore, it is evident that the security roles of different onion layers differ. Hence, when different onion layers can be treated heterogeneously, we might explore other questions, such as the security role played by the outermost onion layer and whether all onion layers are necessary in the Tor setting.

4.2 Separations

This subsection presents the corresponding separation results, specifically Theorem 4 and Theorem 5. Before discussing the specifics of these theorems, we first recap the LBE scheme in [35], as the relevant counterexamples for both theorems are derived from it.

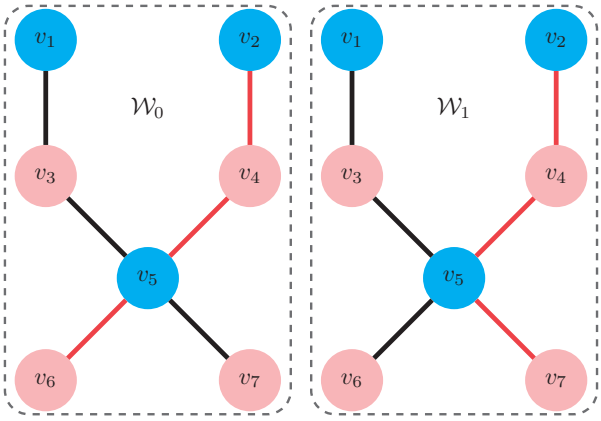

Fig. 5. A typical topology setting ($\mathcal{W}_0$, $\mathcal{W}_1$) in *CircuitHiding*, with red nodes as corrupt ORs and cyan nodes as honest OPs and ORs.

Recapping the LBE scheme in [35]. The LBE scheme in [35] is adapted from "Design 1: Large-block encryption" in Tor proposal 202 [23] by removing the leaky-pipe design. In this scheme, each onion layer employs a tweakable (wide-block) block cipher (TBC) [22], denoted as $(\mathtt{E}, \mathtt{D})$, for encryption and decryption. Compared to a block cipher, a TBC accepts an additional parameter, the tweak, as input to provide a second dimension of randomness[13], which is desirable for onion encryption. For end-to-end authentication, the innermost onion layer follows the encode-then-encipher paradigm. It appends a string of all zeros $0^{n_2-n_1}$ to the plaintext m as part of the input for authentication. This extended input of length n_2 is then encrypted by the TBC at each layer. Importantly, the tweak at each layer encodes the cumulative ciphertext history up to that point, effectively acting as a state. Any midway manipulation will corrupt the tweak and cause decryption that outputs random garbage.

We present a counterexample denoted as Π^* (Figure 6) for Theorem 4. Π^* essentially retains the original design, where each onion layer employs a TBC, and the innermost onion layer follows the encode-then-encipher paradigm. The only modification in Π^* lies in changing the stateful method of the outer $n-1$ onion layers to a simple counter value, as opposed to the original ciphertext cumulating method. This subtle adjustment still allows Π^* to satisfy IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT, thus satisfying *OnionAE*. However, in the *CircuitHiding* game, the adversary can easily determine the actual interacting world.

Theorem 4 (IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT $\not\Rightarrow$ *CircuitHiding*). *There exists an OE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that is IND$-CPA $\wedge$ IPR $\wedge$ INT-sfCTXT secure, but not CircuitHiding secure.*

Proof sketch. (1) IND$-CPA. Π^* utilizes a non-repeating state (either a counter value or a cumulative string) as a tweak for the adopted TBC in each layer. As the tweaks differ, every onion layer provides an independent pseudorandom permutation for each encryption. If the TBC satisfies the STPRP property, IND$-CPA holds.

(2) IPR. For any i-th ($i \in [n-1]$) onion layer, if the ciphertext received by the Dec oracle is the same as the ciphertext generated in the same occurrence by the Enc oracle, then, because both the encryption and decryption sides use the same counter value as a tweak, the decryption at the Dec oracle will match the intermediate ciphertext input into the Enc oracle. If the ciphertexts differ, due to the STPRP property of the TBC, the decryption of the Dec oracle will be a random string.

(3) INT-sfCTXT. The innermost onion layer maintains a non-repeating tweak by ciphertext cumulating. When the $\mathsf{sfVer}(\cdot)$ oracle receives the first out-of-sync ciphertext, the STPRP property of the TBC ensures that the encoding part $0^{n_2-n_1}$ is decrypted into a random string (leading to authentication failure). Furthermore, all subsequent tweaks on the decryption side will deviate from those used on the encryption side. Consequently, all subsequent out-of-sync ciphertexts will result in authentication failure.

(4) $\neg$*CircuitHiding*. We construct a *CircuitHiding* adversary $\mathcal{A}$ such that $\mathbf{Adv}^{CircuitHiding}_{\mathcal{A},\Pi^*}$ is close to 1. The first-stage adversary $\mathcal{A}_1$ specifies the topology setting ($\mathcal{W}_0$, $\mathcal{W}_1$) as shown in Figure 5 for the second-stage adversary $\mathcal{A}_2$. To distinguish $\mathcal{W}_0$ and $\mathcal{W}_1$, $\mathcal{A}_2$ queries two plaintexts at both v_1 and v_2, but the first plaintext received by v_1 is not equal to the first plaintext received by v_2. At both v_3 and v_5, $\mathcal{A}_2$ decrypts the two received ciphertexts normally, but tampers with the first decryption result by

[13] The standard security for TBC is defined as STPRP (strong tweakable pseudorandom permutation). Further descriptions of TBC and STPRP are provided in Appendix B.

$\mathcal{K}(n)$

for $i \leftarrow 1$ **to** n **do**
  $k_i \leftarrow\$ \mathcal{K}$
  $ctr_i \leftarrow 0, u_i \leftarrow \varepsilon$
  $s_i \leftarrow (ctr_i, u_i)$
$\boldsymbol{k} \leftarrow (k_1, ..., k_n)$
$\boldsymbol{s} \leftarrow (s_1, ..., s_n)$
return $(\boldsymbol{k}, k_1 ..., k_n, \boldsymbol{s}, s_1 ..., s_n)$

$\mathcal{D}(k_i, c, s_i)$

parse s_i **as** (ctr_i, u_i)
if $i = n$
  $c' \leftarrow \mathtt{D}(k_i, u_i, c)$
  $u_i \leftarrow u_i \| c$
else
  $c' \leftarrow \mathtt{D}(k_i, \langle ctr_i \rangle_l, c)$
  $ctr_i \leftarrow ctr_i + 1$
$s_i \leftarrow (ctr_i, u_i)$
if $i = n$ **and** $c'[n_1 + 1..n_2] = 0^{n_2 - n_1}$
  then return $(c'[1..n_1], s_i)$
if $i = n$
  return $(\perp, s_i)$
return (c', s_i)

$\mathcal{E}(\boldsymbol{k}, m, \boldsymbol{s})$

parse $\boldsymbol{k}$ **as** $(k_1, ..., k_n)$
parse $\boldsymbol{s}$ **as** $(s_1, ..., s_n)$
for $i \leftarrow 1$ **to** n **do**
  parse s_i **as** (ctr_i, u_i)
$c \leftarrow m \| 0^{n_2 - n_1}$
for $i \leftarrow n$ **downto** 1 **do**
  if $i = n$
    $c \leftarrow \mathtt{E}(k_i, u_i, c)$
    $u_i \leftarrow u_i \| c$
  else
    $c \leftarrow \mathtt{E}(k_i, \langle ctr_i \rangle_l, c)$
    $ctr_i \leftarrow ctr_i + 1$
for $i \leftarrow n$ **downto** 1 **do**
  $s_i \leftarrow (ctr_i, u_i)$
$\boldsymbol{s} \leftarrow (s_1, ..., s_n)$
return $(c, \boldsymbol{s})$

Fig. 6. Counterexample Π^* for Theorem 4. $(\mathtt{E}, \mathtt{D})$ denotes a TBC at each layer, where $\mathtt{E}$ is the encryption function and $\mathtt{D}$ is the decryption function. s_i represents the formal state at the i-th layer, which is either instantiated as a counter value ctr_i $(1 \leq i < n)$ or as a cumulative string u_i $(i = n)$. c' represents the decryption output at the n-th layer, and $c'[n_1 + 1..n_2]$ denotes its substring starting from $n_1 + 1$ and ending at n_2.

flipping the last bit. At v_6, $\mathcal{A}_2$ uses corrupted key and state to reverse engineer the encryption of the first plaintext received by v_1, and further uses this encryption to reset the decryption state at v_6. Similarly, at v_7, $\mathcal{A}_2$ uses corrupted key and state to reverse engineer the encryption of the first plaintext received by v_2, and uses this encryption to reset the decryption state at v_7. This operation leads to the following: In $\mathcal{W}_1$, both the second ciphertexts reaching v_6 and v_7 will be correctly decrypted, while in $\mathcal{W}_0$, both the second ciphertexts reaching v_6 and v_7 will fail to decrypt with negligible probability of exception. This discrepancy arises because, under the IPR property of the onion layer associated with the honest middle OR v_5, in $\mathcal{W}_1$, the reverse engineering and state-resetting operation synchronize the encryption and decryption tweaks at the innermost onion layer of the two circuits. In contrast, in $\mathcal{W}_0$, due to the fact that the first plaintexts received by v_1 and v_2 differ, the reverse engineering and state-resetting operation fails to synchronize the encryption and decryption tweaks at the innermost onion layer of the two circuits, causing both the second ciphertexts reaching v_6 and v_7 to fail decryption. A detailed proof for Theorem 4 is provided in Appendix E.4.

Based on Theorem 1 and Theorem 4, we obtain Corollary 1.

Corollary 1 (*OnionAE* $\not\Rightarrow$ *CircuitHiding*). *There exists an OE scheme that is OnionAE secure, but not CircuitHiding secure.*

We present a counterexample denoted as $\Pi^{\S}$ (Figure 7) for Theorem 5. $\Pi^{\S}$ retains the core design of the LBE scheme, where each onion layer employs a TBC, and the innermost onion layer follows the encode-then-encipher paradigm. The only modification in $\Pi^{\S}$ lies in changing the stateful method of the innermost onion layer to a simple counter value, as opposed to the original ciphertext cumulating method. This subtle adjustment ensures that $\Pi^{\S}$ satisfies IND\$-CPA $\wedge$ IPR$^+$, but it no longer satisfies INT-sfCTXT. Consequently, $\Pi^{\S}$ fails to satisfy *OnionAE*.

Theorem 5 (IND\$-CPA $\wedge$ IPR$^+$ $\not\Rightarrow$ *OnionAE*). *There exists an OE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that is IND\$-CPA $\wedge$ IPR$^+$ secure, but not OnionAE secure.*

$\mathcal{K}(n)$

for $i \leftarrow 1$ **to** n **do**
  $k_i \leftarrow\$ \mathcal{K}$
  $ctr_i \leftarrow 0, u_i \leftarrow \varepsilon$
  $s_i \leftarrow (ctr_i, u_i)$
$\boldsymbol{k} \leftarrow (k_1, ..., k_n)$
$\boldsymbol{s} \leftarrow (s_1, ..., s_n)$
return $(\boldsymbol{k}, k_1 ..., k_n, \boldsymbol{s}, s_1 ..., s_n)$

$\mathcal{D}(k_i, c, s_i)$

parse s_i **as** (ctr_i, u_i)
if $i = n$
  $c' \leftarrow \mathtt{D}(k_i, \langle ctr_i \rangle_l, c)$
  $ctr_i \leftarrow ctr_i + 1$
else
  $c' \leftarrow \mathtt{D}(k_i, u_i, c)$
  $u_i \leftarrow u_i \| c$
$s_i \leftarrow (ctr_i, u_i)$
if $i = n$ **and** $c'[n_1 + 1..n_2] = 0^{n_2 - n_1}$
  then return $(c'[1..n_1], s_i)$
if $i = n$
  return $(\perp, s_i)$
return (c', s_i)

$\mathcal{E}(\boldsymbol{k}, m, \boldsymbol{s})$

parse $\boldsymbol{k}$ **as** $(k_1, ..., k_n)$
parse $\boldsymbol{s}$ **as** $(s_1, ..., s_n)$
for $i \leftarrow 1$ **to** n **do**
  parse s_i **as** (ctr_i, u_i)
$c \leftarrow m \| 0^{n_2 - n_1}$
for $i \leftarrow n$ **downto** 1 **do**
  if $i = n$
    $c \leftarrow \mathtt{E}(k_i, \langle ctr_i \rangle_l, c)$
    $ctr_i \leftarrow ctr_i + 1$
  else
    $c \leftarrow \mathtt{E}(k_i, u_i, c)$
    $u_i \leftarrow u_i \| c$
for $i \leftarrow n$ **downto** 1 **do**
  $s_i \leftarrow (ctr_i, u_i)$
$\boldsymbol{s} \leftarrow (s_1, ..., s_n)$
return $(c, \boldsymbol{s})$

Fig. 7. Counterexample $\Pi^{\S}$ for Theorem 5. $(\mathtt{E}, \mathtt{D})$ denotes a TBC at each layer, where $\mathtt{E}$ is the encryption function and $\mathtt{D}$ is the decryption function. s_i represents the formal state at the i-th layer, which is either instantiated as a counter value ctr_i $(i = n)$ or as a cumulative string u_i $(1 \leq i < n)$. c' represents the decryption output at the n-th layer, and $c'[n_1 + 1..n_2]$ denotes its substring starting from $n_1 + 1$ and ending at n_2..

Proof sketch. (1) IND$-CPA. Each onion layer in $\Pi^{\S}$ satisfies IND$-CPA because it uses a non-repeating state (either a counter or cumulative string) as a tweak for the TBC. When the TBC satisfies the STPRP property, IND$-CPA follows trivially.

(2) IPR$^+$. The outer $n - 1$ onion layers use a non-repeating tweak u_i via ciphertext cumulating. When the decryption oracle Dec receives the first out-of-sync ciphertext, the STPRP property of the TBC guarantees that it is decrypted into a random string. As a result, the tweaks on the decryption side will diverge from those on the encryption side, leading to the continued decryption of all out-of-sync ciphertexts into random strings. Hence, when the TBC satisfies STPRP, IPR$^+$ trivially holds.

(3) ¬INT-sfCTXT ($\neg$*OnionAE*). We demonstrate a straightforward attack against INT-sfCTXT security. The adversary $\mathcal{A}$ first obtains two legitimate ciphertexts c_1, c_2 for messages m_1, m_2 via the encryption oracle. $\mathcal{A}$ then submits a modified version of c_1 (by flipping its last bit) to the verification oracle, which fails to decrypt due to the STPRP property of the TBC. However, when $\mathcal{A}$ subsequently submits the unmodified c_2, it decrypts correctly because the verification oracle uses the same tweak ctr_n as was used during encryption. This violates the INT-sfCTXT security notion, as a successful decryption occurs after an out-of-sync query. Therefore, $\mathbf{Adv}^{\text{INT-sfCTXT}}_{\mathcal{A},\Pi^{\S}} = 1$. A detailed proof for Theorem 5 is provided in Appendix E.5.

Based on Theorem 3 and Theorem 5, we obtain Corollary 2.

Corollary 2 (*CircuitHiding* $\not\Rightarrow$ *OnionAE*). *There exists an OE scheme that is CircuitHiding secure, but not OnionAE secure.*

4.3 Summary

Based on Theorem 3, we have Corollary 3.

Corollary 3 (IND$-CPA $\wedge$ IPR$^+$ $\wedge$ INT-sfCTXT $\Rightarrow$ *CircuitHiding*). *If an OE scheme is IND$-CPA $\wedge$ IPR$^+$ $\wedge$ INT-sfCTXT secure, it is CircuitHiding secure.*

Based on Theorem 1, we have Corollary 4.

Corollary 4 (IND$-CPA ∧ IPR$^+$ ∧ INT-sfCTXT ⇒ *OnionAE*). *If an OE scheme is IND$-CPA ∧ IPR$^+$ ∧ INT-sfCTXT secure, it is OnionAE secure.*

Given that *OnionAE* and *CircuitHiding* are mutually separable, we recommend using the combination "IND$-CPA ∧ IPR$^+$ ∧ INT-sfCTXT" to simultaneously satisfy *CircuitHiding* and *OnionAE*. Another notable feature of this combination is that all notions are onion layer-centric and, thus, easy to use[14].

Remark 4 (Existence of the combination). It is trivial to obtain an OE scheme that satisfies IND$-CPA ∧ IPR$^+$ ∧ INT-sfCTXT; simply set the stateful method of the innermost onion layer in the counterexample $\Pi^{\S}$ for Theorem 5 to ciphertext cumulating. This construction is precisely the LBE scheme in [35], where the authors provided a weaker treatment compared to ours.

5 Limitation and Future Work

To facilitate a comparison between *CircuitHiding* and *OnionAE*, this paper focuses on the onion part. Therefore, the results of this paper establish a common necessary security formalism for OE. However, the onion part corresponds only to one of the highly severe covert channel vectors related to the Tor protocol, namely cryptographic tagging, while ignoring cell header manipulation and dropped cells. All three of these are serious protocol information leaks, as recently recognized by the Tor team [27]. Although the proposed new combination helps enhance OE security, it does not address the other two leaks. In this sense, the newly introduced notion set seems to regress compared to *CircuitHiding*, as the latter also considers cell header manipulation. To address the real-world problems faced by Tor, it is imperative to propose a stronger security model that simultaneously considers all three leak vectors.

Regarding future work, on the one hand, we may explore how to construct a reasonable and stronger security model to address protocol information leaks as a whole. Since protocol leaks caused by dropped cells seem to involve semantic awareness, we may attempt to introduce a zero-knowledge framework to integrate with the current security notions, which may involve reintegration of the security notions. On the other hand, we may also investigate whether the security principles revealed by the treatment in this paper (such as the heterogeneous roles of different onion layers) are helpful in constructing a secure and efficient OE scheme, which we believe is also a valuable endeavor.

References

1. Ando, M., Lysyanskaya, A.: Cryptographic shallots: A formal treatment of repliable onion encryption. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13044, pp. 188–221. Springer (2021). https://doi.org/10.1007/978-3-030-90456-2_7
2. (arma), R.D.: Tor security advisory: "relay early" traffic confirmation attack (July 2014), https://blog.torproject.org/tor-security-advisory-relay-early-traffic-confirmation-attack/
3. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 3–33. Springer, Santa Barbara, CA, USA (2017). https://doi.org/10.1007/978-3-319-63697-9_1
4. Ashur, T., Dunkelman, O., Luykx, A.: 295-relay-crypto-with-adl (February 2018), https://gitweb.torproject.org/torspec.git/tree/proposals/295-relay-crypto-with-adl.txt
5. Backes, M., Goldberg, I., Kate, A., Mohammadi, E.: Provably secure and practical onion routing. In: Proceedings of the 2012 IEEE 25th Computer Security Foundations Symposium. p. 369–385. CSF '12, IEEE Computer Society, USA (2012). https://doi.org/10.1109/CSF.2012.32, https://doi.org/10.1109/CSF.2012.32
6. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. ACM Trans. Inf. Syst. Secur. **7**(2), 206–241 (2004). https://doi.org/10.1145/996943.996945
7. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8424, pp. 367–390. Springer, Singapore (2013). https://doi.org/10.1007/978-3-662-43933-3_19

[14] A discerning reader may notice that verifying IND$-CPA, IPR, and INT-sfCTXT properties in Theorem 4, and verifying IND$-CPA, IPR$^+$, and INT-sfCTXT properties in Theorem 5 are relatively straightforward.

8. Camenisch, J., Lysyanskaya, A.: A formal treatment of onion routing. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 169–187. Springer, Santa Barbara, California, USA (2005). https://doi.org/10.1007/11535218_11
9. Chen, C., Asoni, D.E., Barrera, D., Danezis, G., Perrig, A.: HORNET: high-speed onion routing at the network layer. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015. pp. 1441–1454. ACM (2015). https://doi.org/10.1145/2810103.2813628
10. Chen, C., Asoni, D.E., Perrig, A., Barrera, D., Danezis, G., Troncoso, C.: TARANET: traffic-analysis resistant anonymity at the network layer. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018. pp. 137–152. IEEE (2018). https://doi.org/10.1109/EUROSP.2018.00018
11. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. In: 30th IEEE Symposium on Security and Privacy (SP 2009), 17-20 May 2009, Oakland, California, USA. pp. 269–282. IEEE Computer Society (2009). https://doi.org/10.1109/SP.2009.15
12. Degabriele, J.P., Melloni, A., Stam, M.: Counter galois onion: A new proposal for forward-secure relay cryptography (September 2019), https://gitweb.torproject.org/torspec.git/tree/proposals/308-counter-galois-onion.txt
13. Degabriele, J.P., Stam, M.: Untagging tor: A formal treatment of onion encryption. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 259–293. Springer, Tel Aviv, Israel (2018). https://doi.org/10.1007/978-3-319-78372-7_9
14. Dingledine, R., Mathewson, N.: Tor protocol specification (2024), https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt
15. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13. p. 21. SSYM'04, USENIX Association, USA (2004)
16. Fu, X., Ling, Z.: One cell is enough to break tor's anonymity. In: Proceedings of Black Hat Technical Security Conference. pp. 578–589 (2009)
17. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 15–44. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_2
18. Hogan, K., Servan-Schreiber, S., Newman, Z., Weintraub, B., Nita-Rotaru, C., Devadas, S.: Shortor: Improving tor network latency via multi-hop overlay routing. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022. pp. 1933–1952. IEEE (2022). https://doi.org/10.1109/SP46214.2022.9833619
19. Jansen, R., Traudt, M., Geddes, J., Wacek, C., Sherr, M., Syverson, P.: KIST: kernel-informed socket transport for tor. ACM Trans. Priv. Secur. **22**(1), 3:1–3:37 (2019). https://doi.org/10.1145/3278121
20. Kuhn, C., Beck, M., Strufe, T.: Breaking and (partially) fixing provably secure onion routing. In: 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020. pp. 168–185. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00039
21. Kuhn, C., Hofheinz, D., Rupp, A., Strufe, T.: Onion routing with replies. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 573–604. Springer (2021). https://doi.org/10.1007/978-3-030-92075-3_20
22. Liskov, M.D., Rivest, R.L., Wagner, D.A.: Tweakable block ciphers. J. Cryptol. **24**(3), 588–613 (2011)
23. Mathewson, N.: Two improved relay encryption protocols for tor cells (June 2012), https://gitweb.torproject.org/torspec.git/tree/proposals/202-improved-relay-crypto.txt
24. Mathewson, N.: Aez for relay cryptography (October 2015), https://gitweb.torproject.org/torspec.git/tree/proposals/261-aez-crypto.txt
25. Mathewson, N.: Cryptographic directions in tor: past and future. In: Real World Cryptography Conference (2016)
26. pavel: Tor 2023: Year in review (December 2023), https://blog.torproject.org/2023-year-in-review/
27. Perry, M.: Prioritizing protocol information leaks in tor (December 2023), https://spec.torproject.org/proposals/344-protocol-info-leaks.html
28. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017. pp. 1199–1216. USENIX Association (2017)
29. Project, T.T.: mark old counter galois onion proposal as superseded (December 2023), https://gitlab.torproject.org/tpo/core/torspec/-/commit/c49b5ad4e27df87dc317e79636b3de90198bf719

30. Project, T.T.: Tor metrics portal (jul 2023), https://metrics.torproject
31. 23rd Raccoond, T.: Analysis of the relative severity of tagging attacks (March 2012), https://archives.seul.org/or/dev/Mar-2012/msg00019.html
32. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. IEEE J. Sel. Areas Commun. **16**(4), 482–494 (1998). https://doi.org/10.1109/49.668972
33. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3), 365–403 (2003). https://doi.org/10.1145/937527.937529
34. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques. p. 373–390. EUROCRYPT'06, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11761679_23, https://doi.org/10.1007/11761679_23
35. Rogaway, P., Zhang, Y.: Onion-ae: Foundations of nested encryption. Proc. Priv. Enhancing Technol. **2018**(2), 85–104 (2018). https://doi.org/10.1515/popets-2018-0014
36. Rogaway, P., Zhang, Y.: Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 3–32. Springer, Santa Barbara, CA, USA (2018). https://doi.org/10.1007/978-3-319-96881-0_1
37. Schadt, D., Coijanovic, C., Weis, C., Strufe, T.: Polysphinx: Extending the sphinx mix format with better multicast support. In: IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024. pp. 4386–4404. IEEE (2024). https://doi.org/10.1109/SP54263.2024.00044
38. Scherer, P., Weis, C., Strufe, T.: Provable security for the onion routing and mix network packet format sphinx. Proc. Priv. Enhancing Technol. **2024**(4), 755–783 (2024). https://doi.org/10.56553/POPETS-2024-0140
39. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 405–423. Springer (2013). https://doi.org/10.1007/978-3-642-42033-7_21

A Detailed Notions

A.1 Detailed *CircuitHiding*

We provide the detailed description of *CircuitHiding* game in [13] for completeness. The formalization of the *CircuitHiding* game is provided in Figure 8, directly borrowing from [13].

CircuitHiding provides an anonymity-flavor treatment, namely, how effectively the cryptographic capabilities of an OE scheme can hide the topology of the circuits within Tor from an adversary with static corruption power. To this end, it considers a multiple-user network setting (rather than a single circuit in *OnionAE*), where multiple users create several circuits, across different OP nodes. To better match the anonymity formalism, *CircuitHiding* integrates some technical parts for routing processing and topology (circuit) bookkeeping (As mentioned in Section 2.2, its syntax considers routing behavior besides OE), providing detailed descriptions of the Tor setting. Furthermore, it takes into account critical aspects to ensure the soundness and meaningfulness of this cryptographic treatment. Next, we explain its technical designs in more detail.

Game *CircuitHiding*

$(\mathcal{W}_0, \mathcal{W}_1, \mathcal{P}_c, \mathbf{st}) \leftarrow \mathcal{A}_1$
if $\neg\mathsf{VALID}(\mathcal{W}_0, \mathcal{W}_1, \mathcal{P}_c)$
  return false
$\forall i\ \mathsf{sync}_i \leftarrow \mathbf{true}$
$\varrho \leftarrow \varepsilon;\ n \leftarrow 0;\ b \leftarrow\$ \{0,1\}$
$\mathsf{INIT-CIRC}(\mathcal{W}_b)$
$\tau_{\mathcal{P}_c} \leftarrow \{(v, \boldsymbol{\sigma}_v, \boldsymbol{\tau}_v, \overline{\boldsymbol{\tau}}_v) \mid v \in \mathcal{P}_c\}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{ENC},\mathsf{NET}}(\mathbf{st}, \tau_{\mathcal{P}_c})$
return $b' = b$

NET($\mathbf{z}$)

$\forall i\ \mathsf{assc}_i \leftarrow 0; \mathbf{x} \leftarrow []$
for $i' = 1$ **to** $|\mathbf{z}|$
  $(s, v, c) \leftarrow \mathbf{z}[i']$
  $w \leftarrow \mathcal{D}(\boldsymbol{\tau}_v, s, c)$
  if $s \notin \mathcal{P}_c \vee v \in \mathcal{P}_c \vee w = \perp$
    return $\lightning$
for $i' = 1$ **to** $|\mathbf{z}|$
  $(s, v, c) \leftarrow \mathbf{z}[i']; c^* \leftarrow c$
  $w \leftarrow \mathcal{D}(\boldsymbol{\tau}_v, s, c)$
  $(\overline{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \overline{\mathcal{D}}(\overline{\boldsymbol{\tau}}_v[w], s, c)$
  $(i, j) \leftarrow \mathsf{map}^{-1}(v, w)$
  while $d \notin \mathcal{P}_c \wedge d \neq \oslash$
    $s \leftarrow v, v \leftarrow d$
    $w \leftarrow \mathcal{D}(\boldsymbol{\tau}_v, s, c)$
    $(\overline{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \overline{\mathcal{D}}(\overline{\boldsymbol{\tau}}_v[w], s, c)$
  if $d \in \mathcal{P}_c$
    $\mathbf{x}.\mathsf{append}(v, d, c)$
  if $d \in \mathcal{P}_c \vee i \in \mathcal{I}_{\mathsf{NOP}}$
    $\mathsf{assc}_i \leftarrow \mathsf{assc}_i + 1$
    if $c^* \neq Q^i.\mathsf{dequeue}()$
      $\mathsf{sync}_i \leftarrow \mathbf{false}$
if $\bigvee_{i \in \mathcal{I}_{\mathsf{EN}}} (\mathsf{sync}_i \vee \mathsf{assc}_i \neq 1)$
  return $\lightning$
return $\mathsf{sort}(\mathbf{x})$

INIT − CIRC($\mathcal{W}$)

for $i = 1$ **to** $|\mathcal{W}|$
  $n \leftarrow n + 1;\ \mathbf{p}_n \leftarrow \mathcal{W}[i]$
  $(\varrho, \sigma, \mathbf{t}, \overline{\mathbf{t}}) \leftarrow \mathcal{G}(\varrho, \mathbf{p}_n)$
  $\mathsf{sync}_n \leftarrow \mathbf{true}$
  $\boldsymbol{\sigma}_{\mathbf{p}_n[0]}.\mathsf{append}(\sigma)$
  for $j = 1$ **to** $|\mathbf{p}_n|$
    $v \leftarrow \mathbf{p}_n[j]$
    $\boldsymbol{\tau}_v.\mathsf{append}(\mathbf{t}[j])$
    $\overline{\boldsymbol{\tau}}_v.\mathsf{append}(\overline{\mathbf{t}}[j])$
    if $\mathsf{EN}(\mathbf{p}_n, \mathcal{P}_c) \wedge \mathbf{p}_n[0] \notin \mathcal{P}_c$
      $\mathcal{I}_{\mathsf{EN}} \leftarrow \mathcal{I}_{\mathsf{EN}} \cup \{i\}$
      if $\mathsf{NOP}(\mathbf{p}_n, \mathcal{P}_c)$
        $\mathcal{I}_{\mathsf{NOP}} \leftarrow \mathcal{I}_{\mathsf{NOP}} \cup \{i\}$
foreach v
  $\mathsf{Shuffle}(\boldsymbol{\sigma}_v, \boldsymbol{\tau}_v, \overline{\boldsymbol{\tau}}_v)$

ENC($i,\ m$)

$(v, w) \leftarrow \mathsf{map}(i, 0)$
if $v \in \mathcal{P}_c$
  return $\lightning$
$(\boldsymbol{\sigma}_v[w], d, c) \leftarrow \mathcal{E}(\boldsymbol{\sigma}_v[w], m)$
while $d \notin \mathcal{P}_c$
  $s \leftarrow v, v \leftarrow d$
  $w \leftarrow \mathcal{D}(\boldsymbol{\tau}_v, s, c)$
  $(\overline{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \overline{\mathcal{D}}(\overline{\boldsymbol{\tau}}_v[w], s, c)$
$(v^*, d^*, c^*) \leftarrow (v, d, c)$
while $d \in \mathcal{P}_c$
  $s \leftarrow v, v \leftarrow d$
  $w \leftarrow \mathcal{D}(\boldsymbol{\tau}_v, s, c)$
  $(\overline{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \overline{\mathcal{D}}(\overline{\boldsymbol{\tau}}_v[w], s, c)$
if $d \neq \oslash$
  $(i, j) \leftarrow \mathsf{map}^{-1}(v, w)$
  $Q^i.\mathsf{enqueue}(c)$
return (v^*, d^*, c^*)

Fig. 8. The *CircuitHiding* game in reference [13].

In *CircuitHiding*, an indistinguishability game is played between a two-stage adversary $\mathcal{A}$ ($\mathcal{A}_1$ and $\mathcal{A}_2$) and a challenger. In this game, $\mathcal{A}_1$ generates two generic (and default indistinguishable) network topologies $\mathcal{W}_0$ and $\mathcal{W}_1$, in which some of their ORs are designated as corrupted nodes (by the set $\mathcal{P}_c$). The challenger instantiates a randomly sampled topology $\mathcal{W}_b$ ($b \leftarrow\!\!\$\ \{0,1\}$), and the relevant corrupt states is provided with $\mathcal{A}_2$. The actual adversary $\mathcal{A}_2$, with static corruption power, can query two oracles to interact with $\mathcal{W}_b$: an encryption oracle, ENC, which outputs cells that the adversary can manipulate (e.g., tagging, replaying, reordering, inserting) at a corrupt OR at the front of the circuit, and a network oracle, NET, which outputs cells that the adversary can obtain at a corrupt OR at the back of the circuit. After making polynomial-bounded adaptively chosen queries on ENC and NET, $\mathcal{A}_2$ is asked to guess the actual interacting topology ($\mathcal{W}_0$ or $\mathcal{W}_1$).

Critically, to ensure a fair starting point, $\mathcal{W}_0$ and $\mathcal{W}_1$ should provide no prior advantage to $\mathcal{A}_2$. That is, the choice of topologies by $\mathcal{A}_1$ should not allow $\mathcal{A}_2$ to trivially win. In *CircuitHiding*, a network topology consists of several circuits, some of which contain OR nodes corrupted by $\mathcal{A}_2$. Notably, *CircuitHiding* permits generic topologies, allowing the circuits in $\mathcal{W}_0$ and $\mathcal{W}_1$ to vary in form. For example, a circuit's length can exceed the typical three-node length of Tor; an adversary may corrupt a contiguous sequence of nodes (referred to as a "corrupt segment" in [13]), instead of just the entry node in a canonical tagging attack; and a circuit may include honest OR nodes following a second corrupt segment. Accordingly, for the notion's soundness, the predicate VALID in *CircuitHiding* is introduced to verify a set of conditions that prevent trivial wins. For example, $\mathcal{W}_0$ and $\mathcal{W}_1$ must contain an equal number of circuits, and each circuit in both topologies must include at least one corrupt OR node. (For further details on the VALID predicate and how to prepare a valid pair of $\mathcal{W}_0$ and $\mathcal{W}_1$, refer to [13].)

In terms of the notion's effectiveness, *CircuitHiding* especially considers those circuits with an honest OP. To this end, two sets of circuit indices, $\mathcal{I}_{\mathsf{EN}}$ and $\mathcal{I}_{\mathsf{NOP}}$, are introduced. The former tracks the indices of all circuits that have an honest proxy and contain an entry edge (an edge from a corrupt OR to an honest OR, where the predicate EN returns true), and the latter tracks those in $\mathcal{I}_{\mathsf{EN}}$ that do not have a second corrupt segment. The rationale here is that only the circuits in $\mathcal{I}_{\mathsf{EN}}$ (possibly) provide meaningful observation information for an adversary. It's important to note that the size of $\mathcal{I}_{\mathsf{EN}} \backslash \mathcal{I}_{\mathsf{NOP}}$ may be less than that of $\mathcal{I}_{\mathsf{EN}}$. Therefore, the adversary should not be able to infer the source of cells received at a second corrupt segment from the entry edges.

Furthermore, *CircuitHiding* incorporates some crucial technical tricks to ensure a meaningful cryptographic treatment. As *CircuitHiding* involves a multi-user setting with multiple circuits, specific techniques are used to avoid trivial wins based on time-based analysis. First, a shuffle operation is performed at the end of $\mathsf{INIT}-\mathsf{CIRC}$ to randomize the states observable by the adversary, thus disconnecting the nodes' local view from the order in which circuits were created. To facilitate this, the functions map and map^{-1} are adopted to switch on local and global perspectives. Second, the NET oracle enforces collective access, restricting the adversary to interact with the honest segments of $\mathcal{I}_{\mathsf{EN}}$ in parallel with an input vector $\mathbf{z}$. Additionally, the NET oracle explicitly requires that all circuits are out-of-sync (with bookkeeping performed via the queue Q^i and the variable sync_i) and that only one input cell is present (using assc_i for this purpose), thereby preventing trivial wins.

It is important to note that *CircuitHiding* confines the adversary's influence to the network interface of the corrupt OR nodes. The adversary gains access to the OR's state information, including future updates, can observe all incoming cells to that OR, and has full control over the cells sent to other OR nodes. Thus, in conjunction with the two-stage model, it ensures that the processing of cells in one circuit does not impact another. Furthermore, each cell consistently follows its intended trajectory, maintaining trajectory integrity as per reference [13]. These conditions simply the analysis of *CircuitHiding*, intuitively reducing the multi-user notion to a single-circuit one intuitively.

A.2 Detailed *OnionAE*

Rooted in a different philosophical approach, *OnionAE* focuses on the AE properties of a single circuit, instead of a network-level anonymity formalism in *CircuitHiding*. The formal description of *OnionAE* game is provided in Figure 9.

OnionAE offers a tailored AE treatment for OE. In contrast to the classical stateful AE notions such as INT-sfCTXT and IND-sfCCA [6], the most notable feature in *OnionAE* is the provision of an encryption oracle for the OP and a decryption oracle for each OR in the circuit. This means that it provides multiple decryption oracles simultaneously, thereby aligning with the Tor setting. An adversary can query these oracles arbitrarily, thereby enabling various adversarial behaviors, including replaying, reordering, and more. In this regard, this capability is essentially equivalent to that found in *CircuitHiding*, assuming that the stateful information from corrupt ORs in *CircuitHiding* is not considered.

Formally, in the *OnionAE* game, the adversary's goal is to distinguish between two possible worlds: the real-world game $\mathbf{G}_0$, instantiated by a practical OE scheme, and the ideal-world game $\mathbf{G}_1$, which specifies the expected properties for OE (similar to nonce-based AE notions RUPAE [3] and IND$-CCA3 [34]). Both $\mathbf{G}_0$ and $\mathbf{G}_1$ involve three types of the oracle: the Key oracle for initialization, the Enc oracle for OP encryption, and the Dec oracles for decryption at each OR in the circuit. The adversary $\mathcal{A}$ is allowed to make polynomial-bounded adaptive queries on the Enc and (each) Dec oracles.

Similar to other conventional stateful AE notions, such as INT-sfCTXT and IND-sfCCA [6], *OnionAE* employs crucial bookkeeping techniques to meaningfully address statefulness in the context of Tor. A key feature is the predicate Ψ, called Silence, which tracks the synchronization status of the circuit. Specifically, Ψ monitors all queries made so far to determine whether they form a stack of end-to-end chains, which would indicate the expected correctness of the OE scheme Π. As a result, adversarial behaviors such as such as replaying or reordering can be captured by Ψ. When Ψ returns true, the n-th layer decryption oracle Dec suppresses its output to prevent trivial wins. However, if Ψ returns false, it remains false, and the circuit becomes unsilenced (out-of-sync). In fact, in reference [35], a more simplified notion choice driven by the oracle silencing technique [36] was preferred, which automatically suppresses oracle responses. Nevertheless, for convenience, we adopt the concrete silencing predicate used in that work, avoiding the introduction of additional background complexity.

In terms of the expected properties, the ideal-world game $\mathbf{G}_1$ adopts the IND$ randomness flavor for confidentiality. Specifically, it defines that the Enc oracle outputs a uniformly random string; for the first n-1 ORs in a circuit, the Dec oracle also outputs a uniformly random string. Regarding authenticity, when the circuit becomes unsilenced, the n-th Dec always outputs a $\perp$ symbol indicating an authentication failure.

G_0 (real world)	G_1 (ideal world)
$\underline{\mathsf{Key}(n')}$	$\underline{\mathsf{Key}(n')}$
if $n \neq \perp$ **then return** Err	**if** $n \neq \perp$ **then return** Err
$q \leftarrow 0$	$q \leftarrow 0$
$n \leftarrow \mathrm{n}'$	$n \leftarrow \mathrm{n}'$
$(k_0, k_1, ..., k_n) \leftarrow\$\ \mathcal{K}(\mathrm{n})$	$\underline{\mathsf{Enc}(m)}$
$\underline{\mathsf{Enc}(m)}$	**if** $n = \perp$ **then return** Err
if $n = \perp$ **then return** Err	$q{+}{+}$
$q{+}{+}$	$c \leftarrow\$\ \mathcal{C}$
$(c, u) \leftarrow \mathcal{E}(k_0, m, u)$	$(x_q.\mathrm{type}, x_q.\mathrm{msg}, y_q) \leftarrow (\mathsf{Enc}, m, c)$
$(x_q.\mathrm{type}, x_q.\mathrm{msg}, y_q) \leftarrow (\mathsf{Enc}, m, c)$	**return** c
return c	$\underline{\mathsf{Dec}(c, i)}$
$\underline{\mathsf{Dec}(c, i)}$	**if** $n = \perp$ **then return** Err
if $n = \perp$ **then return** Err	$q{+}{+}$
$q{+}{+}$	**if** $i = n$ **then** $d \leftarrow \perp$
$(d, s_i) \leftarrow \mathcal{D}(k_i, c, s_i)$	**else** $d \leftarrow\$\ \mathcal{C}$
$(x_q.\mathrm{type}, x_q.\mathrm{ctxt}, x_q.\mathrm{idx}, y_q) \leftarrow (\mathsf{Dec}, c, i, d)$	$(x_q.\mathrm{type}, x_q.\mathrm{ctxt}, x_q.\mathrm{idx}, y_q) \leftarrow (\mathsf{Dec}, c, i, d)$
if $\Psi(n, x_1, y_1, ..., x_{q-1}, y_{q-1}, x_q)$ **then** $y_q \leftarrow \lightning$	**if** $\Psi(n, x_1, y_1, ..., x_{q-1}, y_{q-1}, x_q)$ **then** $y_q \leftarrow \lightning$
return y_q	**return** y_q

$\underline{\Psi(n, x_1, y_1, ..., x_{q-1}, y_{q-1}, x_q)}$
$v \leftarrow 0$
for $i \in [n]$ **do** $w_i \leftarrow 0$
for $i \in [q-1]$ **do**
  if $x_i.\mathrm{type} = \mathsf{Enc}$ **then** $v \leftarrow v + 1; (M_v, C_v) \leftarrow (x_i.\mathrm{msg}, y_i)$
  else $j \leftarrow x_i.\mathrm{idx}; w_j \leftarrow w_j + 1; (S[w_j][j-1], D[w_j][j]) \leftarrow (x_i.\mathrm{ctxt}, y_i)$
$w_n{+}{+}; S[w_n][n\text{-}1] \leftarrow x_q.\mathrm{ctxt}$
return $(x_q.\mathrm{idx} = n) \wedge (v \geq w_n) \wedge ((\forall t \in [w_n])\ C_t = S[t][0]) \wedge ((\forall t \in [w_n - 1])\ D[t][n] \neq \perp)$
$\wedge\ ((\forall t \in [w_n]\ \forall j \in [n-1])\ D[t][j] = S[t][j])$

Fig. 9. The *OnionAE* notion in reference [35]. (The original symbol for indicating a suppression event is $\perp$; for semantic unity, we use $\lightning$ instead. Likewise, the original symbol for indicating a decryption failure is $\Diamond$; we use $\perp$ instead.)

B Tweakable Block Cipher

A tweakable block cipher [22] is a pair of functions $(\mathtt{E}, \mathtt{D})$, with $\mathtt{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{C}$ and $\mathtt{D} : \mathcal{K} \times \mathcal{T} \times \mathcal{C} \to \mathcal{M}$, where $\mathcal{K}$ is the key space, $\mathcal{T}$ the tweak space, $\mathcal{M}$ the message space, and $\mathcal{C}$ the ciphertext space. $\mathcal{M} = \mathcal{C}$. $\mathcal{T}$ may be $\{0,1\}^*$. A tweakable wide-block block cipher is a specific type of tweakable block cipher designed to handle inputs of varying lengths.

An STPRP adversary $\mathcal{A}$'s goal is to distinguish two worlds $\mathbf{G}_0$ and $\mathbf{G}_1$, where $\mathbf{G}_0$ is instantiated with $(\mathtt{E}_k, \mathtt{D}_k)$ and $\mathbf{G}_1$ is instantiated with (π, π^{-1}). Here, k is chosen uniformly at random from $\mathcal{K}$; $\mathtt{E}_k/\mathtt{D}_k$ denotes $\mathtt{E}_k(\cdot,\cdot)/\mathtt{D}_k(\cdot,\cdot)$; (π, π^{-1}) is a family of independent, uniformly distributed random permutations over $\mathcal{M}/\mathcal{C}$ indexed by $\mathcal{T}$. The STPRP advantage of $\mathcal{A}$ with respect to $(\mathtt{E}, \mathtt{D})$ is defined by:

$$\mathbf{Adv}^{\mathrm{STPRP}}_{\mathcal{A},(\mathtt{E},\mathtt{D})} = \Pr\left[\mathcal{A}^{\mathbf{G}_0} = 1\right] - \Pr\left[\mathcal{A}^{\mathbf{G}_1} = 1\right].$$

C Authentication and Confidentiality in [13]

Authentication. In [13], a notion of plaintext integrity (referred to as PINT) is proposed. In this notion, if an adversary can generate a successfully decrypted plaintext that was not originally sent by the honest OP, the adversary wins the game. PINT is considered weaker than the authentication in *OnionAE*, as it primarily emphasizes plaintext integrity, while the authentication in *OnionAE* focuses on ciphertext integrity. Additionally, PINT considers the presence of an adversary with corruption power.
Confidentiality. In [13], a left-or-right flavor of confidentiality notion is proposed to capture security under chosen cell attacks. Intuitively, *OnionAE* encompasses this notion, as *OnionAE* inherently implies the left-or-right confidentiality notion (based on the triangle inequality principle). However, the notion in [13] employs a plaintext-oriented suppression, meaning that its concept of out-of-sync is weaker than that of *OnionAE*. In this regard, *OnionAE* appears to provide a stronger characterization.

D Tor proposals on OE

After the recognition of the severity of cryptographic tagging attacks [31], proposal 202 [23] initiated the process of replacing the current OE scheme, a process that has now been underway for over a decade. Given that OE is foundational to the Tor network, and Tor has continuously prioritized enhancing user experience [19,18,26], the Tor team has been exceptionally cautious in selecting a new scheme for implementation.

In proposal 202 [23], Mathewson presented two candidate schemes: 'short-MAC-and-pad' and 'Large-block encryption'. The former is a hop-by-hop authentication scheme that utilizes a MAC and a stream cipher (e.g., AES counter-mode encryption). Each OR checks a MAC value and re-pads the cell to its chosen length before decryption. The latter, in contrast, is an end-to-end authentication scheme utilizing a tweakable wide-block block cipher. Subsequent proposals—261 [24], 295 [4], and 308 [12]—opted for end-to-end authentication. This decision was driven by the fact that hop-by-hop authentication would reduce the actual transmission weight of messages, which is detrimental to low-latency performance. Additionally, hop-by-hop authentication may introduce challenges related to circuit length and position leakage.

Proposal 261 [24] is an instantiation and improvement of 'Large-block encryption' from Proposal 202. It employs the AEZ construction [17] as a tweakable wide-block block cipher for enhanced efficiency. Additionally, by leveraging a Davies-Meyer-type configuration for computing a running value in the tweak encoding method, proposal 261 also aims to provide forward security. However, AEZ is complex to implement and relies on heuristic security analysis. Proposal 261 was deprecated in 2018.

Both proposal 295 [4] and proposal 308 [12] adopt a stateful variant of GCM-RUP [3], introducing randomness in the decryption of manipulated ciphertext. In implementing onion encryption, the GCM-RUP variant uses a portion of the previous onion layer's ciphertext as the nonce for AES counter-mode encryption. Additionally, the output of the inner TBC is incorporated into the ciphertext and serves as the nonce for the next onion layer's GCM-RUP variant. Despite the similarities, there are subtle differences in how the stateful methods are applied in the two proposals. Proposal 295 [4] uses the running digest of GHASH in the inner TBC as the state, whereas proposal 308 [12] uses the nonce as the state. Proposal 295 utilizes a PIV construction [39] in the innermost onion layer for end-to-end authentication, while proposal 308 adopts a key-evolving technique in the innermost layer to ensure forward secrecy and forward authentication. Both proposals leverage mature building blocks, such as GHASH (or POLYVAL) and the AES block cipher, aiming to strike an optimal balance between security and performance. However,

proposal 308 was superseded on August 15, 2023, with no identified successor, leaving proposal 295 as the only open proposal for OE.

E Proof details

E.1 Detailed proof for Lemma 1

Proof. To ensure clarity throughout the proof, we introduce certain notations and conventions.

Let c^* represent the first out-of-sync ciphertext in the *OnionAE* experiment. We associate c^* with the i-th Dec oracle, where the 0-th Dec oracle corresponds to the Enc oracle. We denote Σ_i and Σ_{i-1} as the cumulative number of received query ciphertexts at the i-th and $(i-1)$-th Dec oracles, respectively, when c^* occurs. Especially, Σ_n represents the cumulative number of received query ciphertexts at the n-th Dec oracle when c^* occurs (in the first main case mentioned below), or when the n-th Dec oracle receives one additional query ciphertext after c^* occurs (in the second main case mentioned below). Furthermore, Σ_0 denotes the cumulative number of received query plaintexts at the Enc oracle when Σ_n takes effect. We define the k-th round as the horizontal bookkeeping when the Enc oracle or a Dec oracle receives a k-th query. The k-th round is considered complete when both the Enc oracle and each Dec oracle have undergone a k-th query.

There are two major cases to consider:

(I). $i = n$. c^* targets the n-th Dec oracle. In this case, $\Sigma_i = \Sigma_n$, $\Sigma_{i-1} = \Sigma_{n-1}$. Three subcases need further consideration:

(1) $\Sigma_i \leq \Sigma_{i-1} \leq \Sigma_0$. This case corresponds to c^* not being equal to the decryption output of the $(n-1)$-th Dec oracle, thereby disrupting the stack of end-to-end chains on the circuit at the Σ_n-th round. Therefore, c^* corresponds to the first out-of-sync and modified ciphertext of INT-sfCTXT. In this case, the exceptional probability is zero.

(2) $\Sigma_i > \Sigma_{i-1} = \Sigma_0$. This case corresponds to c^* being a direct query to the n-th Dec oracle within the Σ_n-th round, while the cumulative number of queried ciphertexts at the previous $n-1$ Dec oracles and the Enc oracle is $\Sigma_n - 1$. Therefore, c^* corresponds to the first out-of-sync and inserted ciphertext of INT-sfCTXT. In this case, the exceptional probability is zero.

(3) $\Sigma_0 \geq \Sigma_i > \Sigma_{i-1}$. This case corresponds to c^* being a direct query to the n-th Dec oracle within the Σ_n-th round, while the Enc oracle has been queried at least Σ_n times. Since the adversary $\mathcal{A}$ does not know the ground truth of $\mathbf{c}_{\Sigma_n}^{(n)}$, the probability of violating INT-sfCTXT's out-of-sync is $\frac{1}{2^{n_2}}$.

(II). $i < n$. c^* targets one of the front $n-1$ Dec oracles. There are four subcases to further consider.

(4) $\Sigma_n \leq \Sigma_i \leq \Sigma_{i-1} \leq \Sigma_0$. This case corresponds to c^* not being equal to the decryption output of the $(i-1)$-th Dec oracle (thus a modified ciphertex), thereby disrupting the stack of end-to-end chains on the circuit at the Σ_i-th round. Meanwhile, when the adversary $\mathcal{A}$ queries the n-th Dec oracle, the n-th Dec oracle has been queried at most Σ_i times. The adversary $\mathcal{A}$ has two possible ways to violate the out-of-sync concept in INT-sfCTXT for the Σ_i-th round. One is by directly querying the n-th Dec oracle with a guessed input, with an exceptional probability of $\frac{1}{2^{n_2}}$. The other is by querying the n-th Dec oracle using the output of the $(n-1)$-th Dec oracle. In this case, the exceptional probability depends on the probability that the output of the $(n-1)$-th Dec oracle is equal to $\mathbf{c}_{\Sigma_n}^{(n)}$. If the adversary $\mathcal{A}$ directly queries the $(n-1)$-th Dec oracle (with a guessing input), according to Definition 4, the exceptional probability is bounded by $\frac{1}{2^{n_2}} + \mathbf{Adv}_{\mathcal{B},\Pi}^{\text{IPR-(n-1)}}$, where $\mathcal{B}$ introduces only one out-of-sync ciphertext to its own $\mathsf{Dec}(\cdot, n-1, 0)$ oracle. If the adversary $\mathcal{A}$ queries the $(n-1)$-th Dec oracle using the output of the $(n-2)$-th Dec oracle, this case can be reduced to the probability that the output of the $(n-2)$-th Dec oracle is equal to $\mathbf{c}_{\Sigma_n}^{(n-1)}$. Therefore, based on induction, the exceptional probability in the second case is bounded by $(n-i) \cdot (\frac{1}{2^{n_2}} + \mathbf{Adv}_{\mathcal{B},\Pi}^{\text{IPR-(i)}}) + \frac{1}{2^{n_2}}$, where $\mathcal{B}$ introduces only one (i.e., the first caused by $\mathcal{A}$) out-of-sync ciphertext to its own $\mathsf{Dec}(\cdot, i, 0)$ oracle[15]. For any intermediate round d ($\Sigma_n \leq d < \Sigma_i$, if $\Sigma_n < \Sigma_i$), similarly and clearly, the exceptional probability is less than $(n-i) \cdot (\frac{1}{2^{n_2}} + \mathbf{Adv}_{\mathcal{B},\Pi}^{\text{IPR-(i)}}) + \frac{1}{2^{n_2}}$.

(5) $\Sigma_n = \Sigma_i > \Sigma_{i-1} = \Sigma_0$. This case corresponds to c^* being a direct query to the i-th Dec oracle within the Σ_n-th round, while the $(\Sigma_n - 1)$-th round has been completed and the entire circuit maintains the stack of end-to-end chains. When querying the n-th Dec oracle for the Σ_n-th time, the Enc oracle has only been queried $\Sigma_n - 1$ times. Therefore, $\overline{\mathbf{c}}_{\Sigma_n}^{(n)}$ corresponds to the first out-of-sync and inserted ciphertext of INT-sfCTXT. In this case, the exceptional probability is zero.

[15] For convenience, here we use $\mathbf{Adv}_{\mathcal{A},\Pi}^{\text{IPR-(i)}}$ to replace the IPR advantage of other onion layers, and the same applies below.

(6) $\Sigma_n \leq \Sigma_{i-1} < \Sigma_i \ \wedge \ \Sigma_i \leq \Sigma_0$. This case corresponds to c^* being a direct query to the i-th Dec oracle within the Σ_i-th round, when the Σ_n-th round is yet to be completed. The adversary $\mathcal{A}$ may violate the out-of-sync concept in INT-sfCTXT either during the Σ_i-th round or in some d-th intermediate round ($\Sigma_n \leq d < \Sigma_{i-1}$, if $\Sigma_n < \Sigma_{i-1}$). By analogy with case (4), the exceptional probability is bounded by $(n-i)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{B},\Pi}) + \frac{1}{2^{n_2}}$.

(7) $\Sigma_0 \geq \Sigma_n = \Sigma_i > \Sigma_{i-1}$. This case corresponds to c^* being a direct query to the i-th Dec oracle within the Σ_n-th round. Meanwhile, when querying the n-th Dec oracle for the Σ_n-th time, the Enc oracle has been queried at least Σ_n times. Since the Enc oracle has been queried at least Σ_n times, referring to case (4), the exceptional probability is bounded by $(n-i)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{B},\Pi}) + \frac{1}{2^{n_2}}$.

Considering all the above cases, the adversary $\mathcal{A}$ can violate the out-of-sync concept in INT-sfCTXT with a probability of at most $(n-1)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{B},\Pi}) + \frac{1}{2^{n_2}}$, which is negligible.

□

E.2 Detailed proof for Theorem 1

Proof. The specific game hopping process is as follows.
Game $\mathbf{G}_0$: This corresponds to the unchanged real world G_0[16] in *OnionAE*.
Game $\mathbf{G}_1$: In this game, when the *OnionAE* adversary introduces the first out-of-sync ciphertext, it triggers the out-of-sync concept in INT-sfCTXT of the innermost onion layer.

By Lemma 1, the advantage gap from G_0 to G_1 is bounded by $(n-1)\cdot(\frac{1}{2^{n_2}} + \mathbf{Adv}^{\text{IPR-(i)}}_{\mathcal{F}_2,\Pi}) + \frac{1}{2^{n_2}}$, where $\mathcal{F}_2$ introduces only one out-of-sync ciphertext to its own $\mathsf{Dec}(\cdot, i, b)$ oracle.
Game $\mathbf{G}_2$: In this game, when $\mathcal{A}$ queries the last Dec oracle with an out-of-sync ciphertext, the last Dec oracle always returns $\perp$. Namely, no out-of-sync ciphertext is correctly decrypted.

To bound the advantage gap from G_1 to G_2, we construct an INT-sfCTXT adversary $\mathcal{B}$, using all the queries made by $\mathcal{A}$ to break the INT-sfCTXT game. Specifically, $\mathcal{B}$ utilizes the keys and initial state information of the outer $n-1$ onion layers to simulate the *OnionAE* game perfectly.

$\mathcal{B}$ works as follows:

- $\mathcal{B}$ initializes the symmetric keys and initial states for the outer $n-1$ onion layers.
- When $\mathcal{A}$ queries the Enc oracle with plaintext m, $\mathcal{B}$ first queries its own Enc oracle using m and obtains the intermediate ciphertext $\overline{c}$. Then $\mathcal{B}$ applies the onion encryption layer by layer using the available keys and states to compute the final ciphertext c, following the onion encryption rules. $\mathcal{B}$ updates the relevant encryption states. The ciphertext c is returned to the *OnionAE* adversary $\mathcal{A}$.
- When $\mathcal{A}$ queries one of the front $n-1$ Dec oracles with ciphertext c, $\mathcal{B}$ uses the corresponding key and state for the specified layer to decrypt c, obtaining the decrypted ciphertext c'. $\mathcal{B}$ updates the relevant decryption state. The result c' is returned to $\mathcal{A}$.
- When $\mathcal{A}$ queries the n-th Dec oracle with ciphertext c, $\mathcal{B}$ queries its own sfVer oracle using c. It obtains the response m and returns m to $\mathcal{A}$.

It is evident that the above simulation satisfies that the advantage gap of $\mathcal{A}$ from G_1 to G_2 is bounded by $\mathbf{Adv}^{\mathsf{INT-sfCTXT}}_{\mathcal{B},\Pi}$.
Game $\mathbf{G}_3$: In this game, both the Enc oracle and the first $n-1$ Dec oracles always output a uniformly random string for each query (and, of course, the last Dec oracle always returns $\perp$ for any received out-of-sync ciphertext). Consequently, game G_3 is identical to the ideal world G_1 in *OnionAE*. Since the last Dec oracle always returns $\perp$, it can be trivially simulated by any adversary. To bound the advantage gap from G_2 to G_3, we only need to prove that the outputs of the Enc and the first $n-1$ Dec oracles in game G_2 are computationally indistinguishable from a uniformly random string for $\mathcal{A}$.

Recall that the output of the Enc oracle in G_2 is the result of encryption by the outermost onion layer. If the outermost onion layer satisfies IND$-CPA, the output of the Enc oracle is computationally indistinguishable from a uniformly random string, regardless of what the query plaintext m to Enc is.

When examining the i-th Dec oracle ($1 \leq i < n$), we can further consider two cases:

(1) If the j-th query to the i-th Dec oracle (i.e., $\overline{\mathbf{c}}_j^{(i)}$) matches a predetermined ciphertext (i.e., $\mathbf{c}_j^{(i)}$), the output of the oracle corresponds to the result of encryption by the $(i+1)$-th onion layer, specifically $\mathbf{c}_j^{(i+1)}$. This is guaranteed by two aspects:

[16] Although the naming of G_0 here conflicts with its earlier usage, readers should be able to distinguish between them based on the context.

(a). $\overline{\mathbf{c}}_j^{(i)}$ is still an in-sync query (i.e., the i-th Dec oracle has not yet received any out-of-sync ciphertext). In this case, the decryption of the i-th Dec oracle is clearly equal to $\mathbf{c}_j^{(i+1)}$.

(b). $\overline{\mathbf{c}}_j^{(i)}$ is an out-of-sync query (i.e., the i-th Dec oracle has already received an out-of-sync ciphertext). In this case, by the guarantee of IPR-(i), the decryption of the i-th Dec oracle remains equal to $\mathbf{c}_j^{(i+1)}$.

Therefore, regardless of the scenario, if the $(i+1)$-th onion layer satisfies IND\$-CPA, the decryption of the i-th Dec oracle is computationally indistinguishable from a uniformly random string, irrespective of the plaintext m queried to the Enc oracle. Thus, the advantage gap of the adversary $\mathcal{A}$ from G_2 to G3 can be collectively covered by the advantage of another IND\$-CPA adversary $\mathcal{C}$ and the advantage of another IPR adversary $\mathcal{F}_1$, denoted as $\mathbf{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi}$ and $\mathbf{Adv}^{\mathsf{IPR}}_{\mathcal{F}_1,\Pi}$, respectively. Here, both $\mathcal{C}$ and $\mathcal{F}_1$ uses at most all the queries to the Enc oracle and the first $n-1$ Dec oracles.

(2) If the j-th query to the i-th Dec oracle (i.e., $\overline{\mathbf{c}}_j^{(i)}$) is not equal to a predetermined ciphertext $\mathbf{c}_j^{(i)}$, it triggers the independent randomness of the i-th Dec oracle. Consequently, the output of the i-th Dec oracle remains computationally indistinguishable from a uniformly random string, regardless of the query plaintext m to the Enc oracle. It follows that the advantage gap of adversary $\mathcal{A}$ from G_2 to G3 can be solely covered by the advantage of an IPR adversary $\mathcal{F}_1$, denoted as $\mathbf{Adv}^{\mathsf{IPR}}_{\mathcal{F}_1,\Pi}$. Here, $\mathcal{F}_1$ uses at most all these queries on Enc and the first $n-1$ Dec oracles.

For $\mathbf{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi}$ and $\mathbf{Adv}^{\mathsf{IPR}}_{\mathcal{D},\Pi}$, we omit the specific reduction process because the properties of Definition 3 and 4 are self-evident. Considering all things together, the result of Theorem 1 can be characterized. □

E.3 Detailed proof for Theorem 3

Proof. The specific game hopping process is as follows.

Game $\mathbf{G}_0$: This corresponds to the unchanged *CircuitHiding* game.

Game $\mathbf{G}_1$: In this game, in both $\mathcal{W}_0$ and $\mathcal{W}_1$, when the *CircuitHiding* adversary $\mathcal{A}$ makes the first out-of-sync vector query to the NET oracle, the last OR of the honest middle segment in each circuit that has an honest OP and two corrupt segments (i.e., circuits in the set $\mathcal{I}_{\mathsf{EN}}\backslash\mathcal{I}_{\mathsf{NOP}}$) receives an out-of-sync ciphertext. In other words, for the circuits in $\mathcal{I}_{\mathsf{EN}}\backslash\mathcal{I}_{\mathsf{NOP}}$, not only does the first OR of the honest middle segment receive an out-of-sync ciphertext due to this NET query, but the out-of-sync concept propagates iteratively to the last OR of the honest middle segment (if the honest middle segment consists of multiple ORs).

To bound the advantage gap from G_0 to G_1, we use the identical-until-bad model for reasoning. When an OR receives an out-of-sync ciphertext, its decryption output remains an out-of-sync ciphertext for the next OR. By the IPR$^+$ property, the bad probability is bounded by $\mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_2,\Pi} + \frac{1}{2^{n_2}}$, where the IPR$^+$ adversary $\mathcal{B}_2$ introduces only one out-of-sync ciphertext to its Dec oracle. Since there can be at most $n_{max}-3$ honest ORs before the last OR of the honest middle segment, it follows that the advantage gap from G_0 to G_1 is bounded by $2l(n_{max}-3)\cdot(\mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_2,\Pi} + \frac{1}{2^{n_2}})$.

Game $\mathbf{G}_2$: In this game, in both $\mathcal{W}_0$ and $\mathcal{W}_1$, we replace the decryption layer associated with the last OR of the honest middle segment in each circuit in the set $\mathcal{I}_{\mathsf{EN}}\backslash\mathcal{I}_{\mathsf{NOP}}$ by outputting a uniformly random string whenever an out-of-sync ciphertext is received.

To bound the advantage gap from G_1 to G_2, we construct an IPR$^+$ adversary $\mathcal{B}_1$ for each circuit, which invokes queries to the ENC and NET oracles and simulates them for the *CircuitHiding* adversary $\mathcal{A}$ by querying its Proc', Enc, and Dec oracles, thus breaking the independent pseudorandomness of the replaced onion layer as per Definition 5 (IPR$^+$). Specifically, consider a fixed circuit. $\mathcal{B}_1$ initializes the symmetric keys and states for the onion layers that encrypt after the replaced onion layer (with the corrupt keys and states associated with the first corrupt segment being directly provided to $\mathcal{B}_1$).

- When $\mathcal{A}$ queries the ENC oracle with a plaintext m, $\mathcal{B}_1$ first queries its Proc' oracle using m and obtains an intermediate ciphertext $\overline{c}$. Then, $\mathcal{B}_1$ queries its Enc oracle using $\overline{c}$ to obtain another intermediate ciphertext $\overline{c}'$. Subsequently, $\mathcal{B}_1$ performs the remaining onion-layer encryptions using the keys and states associated with the first corrupt segment and the honest ORs preceding the last OR of the honest middle segment, producing the final ciphertext c. $\mathcal{B}_1$ updates the relevant encryption states. Finally, c is returned to $\mathcal{A}$.
- When $\mathcal{A}$ queries the NET oracle with a ciphertext c, $\mathcal{B}_1$ first performs the relevant onion-layer decryptions using the available keys and states associated with the honest ORs preceding the last OR of the honest middle segment, obtaining an intermediate ciphertext $\overline{c}$. Then, it queries its Dec oracle using $\overline{c}$ to obtain the final ciphertext $\overline{c}'$. $\mathcal{B}_1$ updates the relevant decryption states. Finally, $\overline{c}'$ is returned to $\mathcal{A}$.

Clearly, $\mathcal{B}_1$ perfectly simulates the ENC and NET oracles for $\mathcal{A}$. Using a standard hybrid argument, the advantage gap from G_1 to G_2 is bounded by $2l \cdot \mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_1,\Pi}$.

Game G_3: In this game, for both $\mathcal{W}_0$ and $\mathcal{W}_1$, we replace the encryption layer associated with the honest OR of the first entry edge of each circuit that have an honest OP (i.e., circuits in the set $\mathcal{I}_{\mathsf{EN}}$) by outputting a uniformly random string whenever an intermediate ciphertext is input.

To bound the advantage gap from G_2 to G_3, we construct an IND$-CPA adversary $\mathcal{C}$ on each circuit that invokes the queries to the ENC oracle and simulates the ENC oracle for the *CircuitHiding* adversary $\mathcal{A}$ by querying its Proc and Enc oracles, thereby breaking the pseudorandomness of the replaced onion layer as per Definition 3 (IND$-CPA). Specifically, consider a fixed circuit. $\mathcal{C}$ is directly provided with the symmetric keys and states associated with the first corrupt segment. When $\mathcal{A}$ queries the ENC oracle with a plaintext m, $\mathcal{C}$ first queries its Proc oracle using m and obtains an intermediate ciphertext $\overline{c}$. Then, it queries its Enc oracle using $\overline{c}$ to obtain another intermediate ciphertext $\overline{c}'$. Subsequently, $\mathcal{C}$ performs the remaining onion-layer encryptions using the corrupt keys and states, producing the final ciphertext c. $\mathcal{C}$ updates the relevant encryption states. Finally, c is returned to $\mathcal{A}$. Clearly, $\mathcal{C}$ perfectly simulates the ENC oracle for $\mathcal{A}$. Using a standard hybrid argument, the advantage gap from G_2 to G_3 is bounded by $2l \cdot \mathbf{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi}$.

Thus far, in both $\mathcal{W}_0$ and $\mathcal{W}_1$, the encryption layer associated with the honest OR of the first entry edge of each circuit in $\mathcal{I}_{\mathsf{EN}}$ outputs a uniformly random string for each inputted intermediate ciphertext, and the last OR in the honest middle segment of each circuit in $\mathcal{I}_{\mathsf{EN}} \backslash \mathcal{I}_{\mathsf{NOP}}$ outputs a uniformly random string for each received out-of-sync ciphertext. Further, since the uniformly random strings produced by the encryption layer associated with the honest OR of the first entry edge would undergo encryption by the encryption layers of the first corrupted segment and the relevant keys and states associated with the first corrupt segment in $\mathcal{W}_0$ and $\mathcal{W}_1$ are identically distributed, the outputs of the ENC oracle in both $\mathcal{W}_0$ and $\mathcal{W}_1$ are also identically distributed. Additionally, as the outputs from the ENC oracle and those from the NET oracle are now independent, the circuits in $\mathcal{I}_{\mathsf{EN}}$ do not contribute any distinguishing advantage for the *CircuitHiding* adversary $\mathcal{A}$. Recall that in *CircuitHiding*, a circuit with a corrupt OP is required to be identical in both $\mathcal{W}_0$ and $\mathcal{W}_1$ by the predicate VALID. Furthermore, everything that happens within such a circuit is is deterministic for the adversary. Therefore, in the final game G_3, $\mathcal{A}$ has no advantage in distinguishing between the two worlds. Putting everything together we have $\mathbf{Adv}^{\mathit{CircuitHiding}}_{\mathcal{A},\Pi} \leq 2l \cdot (\mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_1,\Pi} + \mathbf{Adv}^{\mathsf{IND\$-CPA}}_{\mathcal{C},\Pi} + (n_{max} - 3) \cdot (\mathbf{Adv}^{\mathsf{IPR}^+}_{\mathcal{B}_2,\Pi} + \frac{1}{2^{n_2}}))$. □

E.4 Detailed proof for Theorem 4

Proof. We proceed to demonstrate that Π^* satisfies IND$-CPA ∧ IPR ∧ INT-sfCTXT and that Π^* does not satisfy *CircuitHiding*.

Claim 1. Π^* satisfies IND$-CPA for all onion layers.

Verifying that all onion layers of Π^* satisfy IND$-CPA is relatively straightforward. Π^* employs a non-repeating state (either a counter value or a cumulative string) as a tweak for the TBC $(\mathtt{E}, \mathtt{D})$. Therefore, upon each encryption, a new and non-repeating tweak is generated for the next encryption. As the tweaks differ, every onion layer provides an independent pseudorandom permutation for each encryption. Thus, if the TBC $(\mathtt{E}, \mathtt{D})$ employed by Π^* satisfies STPRP, IND$-CPA for all onion layers holds.

Claim 2. Π^* satisfies IPR for the outer $n-1$ onion layers.

For the i-th ($i \in [n-1]$) onion layer and any non-zero integer k, if the k-th actual received ciphertext at the decryption side (i.e., the Dec oracle) denoted as $\overline{\mathbf{c}}^{(i)}_j$ equals the k-th ciphertext generated at the encryption side (i.e., the Enc oracle) denoted as $\mathbf{c}^{(i)}_j$, the decryption of $\overline{\mathbf{c}}^{(i)}_j$ is the intermediate ciphertext (i.e., $\mathbf{c}^{(i+1)}_j$) for generating $\mathbf{c}^{(i)}_j$, because the encryption function $\mathtt{E}$ and the decryption function $\mathtt{D}$ use the same counter value for tweak. If $\overline{\mathbf{c}}^{(i)}_j \neq \mathbf{c}^{(i)}_j$, although $\mathtt{E}$ and $\mathtt{D}$ use the same counter value, due to the STPRP property of the TBC, the decryption of $\overline{\mathbf{c}}^{(i)}_j$ is indistinguishable from a uniformly random string. Finally, if $\overline{\mathbf{c}}^{(i)}_j$ consumes a counter unused by $\mathtt{E}$ yet, the decryption of $\overline{\mathbf{c}}^{(i)}_j$ is also indistinguishable from a uniformly random string due to the STPRP property of the TBC.

Claim 3. Π^* satisfies INT-sfCTXT for the innermost onion layer.

Note that the innermost onion layer maintains a non-repeating state u_n by ciphertext cumulating, using it as a tweak. On the decryption side (i.e., the $\mathsf{sfVer}(\cdot)$ oracle), when the first out-of-sync ciphertext is received, the tweakable block cipher $(\mathtt{E}, \mathtt{D})$—due to its STPRP property—decrypts the encoding part $0^{n_2-n_1}$ into a random string. As a result, all subsequent tweaks on the decryption side will deviate from those used on the encryption side (i.e., the $\mathsf{Enc}(\cdot)$ oracle). Consequently, the encoding part $0^{n_2-n_1}$ of

all subsequent out-of-sync ciphertexts will also be decrypted into random strings. Furthermore, when the TBC $(\mathtt{E}, \mathtt{D})$ is replaced by a family of independent, uniformly distributed random permutations, the adversary's probability of successfully authenticating out-of-sync ciphertexts is bounded by $\frac{q}{2^{n_2-n_1}}$ (negligible), where q is the total number of such ciphertexts received by the decryption side.

Claim 4. Π^* does not satisfy *CircuitHiding*.

We construct an *CircuitHiding* adversary $\mathcal{A}$ with $\mathbf{Adv}^{CircuitHiding}_{\mathcal{A},\Pi^*}$ close to 1.

$\mathcal{A}_1$ first outputs the topology setting $(\mathcal{W}_0, \mathcal{W}_1)$ as shown in Figure 5, where $\mathcal{W}_0 = [[v_1, v_3, v_5, v_7],[v_2, v_4, v_5, v_6]]$, $\mathcal{W}_1 = [[v_1, v_3, v_5, v_6],[v_2, v_4, v_5, v_7]]$, and v_3, v_4, v_6, v_7 are corrupted ORs. After $\mathcal{W}_b$ $(b \leftarrow\$ \{0,1\})$ is validated by the predicate VALID and initialized by the procedure INIT-CIRC, $\mathcal{A}_2$ is provided with the relevant corrupt keys and states. For an interacting world $\mathcal{W}_b$, adversary $\mathcal{A}_2$ operates as follows:

1. $\mathcal{A}_2$ queries the ENC oracle with the pairs $(1, m_1)$, $(1, m_2)$, $(2, m_3)$, $(2, m_4)$ (where $m_1 \neq m_3$) and receives the corresponding responses: (v_1, v_3, c_1), (v_1, v_3, c_2), (v_2, v_4, c_3), (v_2, v_4, c_4).
2. At v_3, $\mathcal{A}_2$ decrypts ciphertexts c_1 and c_2, obtaining c'_1 and c'_2, respectively. Similarly, at v_4, $\mathcal{A}_2$ decrypts c_3 and c_4, yielding c'_3 and c'_4. $\mathcal{A}_2$ then submits two queries to the NET oracle: $[(v_3, v_5, c'_1 \oplus 1)), (v_4, v_5, c'_3 \oplus \|1))]$ and $[(v_3, v_5, c'_2), (v_4, v_5, c'_4)]$. The first query, $[(v_3, v_5, c'_1 \oplus 1)), (v_4, v_5, c'_3 \oplus \|1))]$, cause both circuits in $\mathcal{W}_b$ to go out of sync, as the last bit of both c'_1 and c'_3 is flipped. As a result, both queries are valid and the NET oracle does not return $\lightning$ for them. Finally, $\mathcal{A}_2$ receives the sorted responses: $[(v_5, v_6, c''_1), (v_5, v_7, c''_3)]$ and $[(v_5, v_6, c''_2), (v_5, v_7, c''_4)]$.
3. $\mathcal{A}_2$ uses the corrupted key k_{v_6} and initial state s_{v_6} of v_6 to encrypt m_1 reversely, obtaining a hypothesized correct version c^a (i.e., the ciphertext reaching v_6 without flipping) for the first ciphertext (i.e., c''_1) arriving at v_6. Then, adversary $\mathcal{A}_2$ uses c^a to reset the decryption state required for the second ciphertext arriving at v_6 (i.e., c''_2).
   Likewise, adversary $\mathcal{A}_2$ uses the corrupted key k_{v_7} and initial state s_{v_7} of v_7 to encrypt m_3 reversely, obtaining a hypothesized correct version c^b for the first ciphertext (i.e., c''_3) arriving at v_7. Then, adversary $\mathcal{A}_2$ uses c^b to reset the decryption state required for the second ciphertext (i.e., c''_4) arriving at v_7.
4. At v_6, $\mathcal{A}_2$ decrypts the second ciphertext c''_2 using the reset decryption state.
   At v_7, $\mathcal{A}_2$ decrypts the second ciphertext c''_4 using the reset decryption state.
   If decryption is successful at both v_6 and v_7, determine that $\mathcal{W}_b$ is $\mathcal{W}_1$; otherwise, determine that $\mathcal{W}_b$ is $\mathcal{W}_0$.

When $\mathcal{W}_b$ is $\mathcal{W}_1$, v_6 and v_1 belong to the same circuit. The second ciphertext reaching v_6 (i.e., c''_2) is indeed the encryption of m_2 by the innermost onion layer. This holds true because, although $c'_1 \oplus 1$ causes c''_1 to be random garbage, the decryption of c'_2 remains intact due to the IPR property of the onion layer associated with v_5. Furthermore, the reset state at v_6 is indeed the tweak used to encrypt m_2 at v_1. Therefore, using the reset state, the second ciphertext c''_2 is guaranteed to be correctly decrypted. Similarly, at v_7, its second ciphertext c''_4 is guaranteed to be correctly decrypted.

When $\mathcal{W}_b$ is $\mathcal{W}_0$, v_6 and v_1 belong to different circuits, but v_6 and v_2 share the same circuit. In this case, the second ciphertext reaching v_6 (i.e., c''_2) is actually the encryption of m_4 by the innermost onion layer. To correctly decrypt c''_2, it is necessary to reverse engineer the encryption of m_3 by using the corrupted state at v_6. However, adversary $\mathcal{A}_2$ uses the corrupted state at v_6 to reverse engineer the encryption of m_1. Since $m_1 \neq m_3$, the two related ciphertexts are not identical. Therefore, for the encryption of m_4, the tweak used for decryption at v_6 is different from the one used for encryption at v_2, and v_6 will deterministically fail to decrypt (with negligible probability of exception). Similarly, the second ciphertext reaching v_7 (i.e., c''_4) will also fail to decrypt, with negligible probability of exception.

Therefore, $\mathcal{W}_0$ and $\mathcal{W}_1$ can be clearly distinguished. $\mathbf{Adv}^{CircuitHiding}_{\mathcal{A},\Pi^*}$ is (almost) 1. □

E.5 Detailed proof for Theorem 5

Proof. We consecutively demonstrate that $\Pi^{\S}$ satisfies IND$-CPA and IPR$^+$, but does not satisfy INT-sfCTXT.

Claim 1. $\Pi^{\S}$ satisfies IND$-CPA for all onion layers.

The reason that all onion layers satisfy IND$-CPA, as outlined in the proof of Claim 1 in Theorem 4, is that $\Pi^{\S}$ employs a non-repeating state (either a counter value or or a cumulative string) as a tweak for the TBC $(\mathtt{E}, \mathtt{D})$. When the TBC used by $\Pi^{\S}$ satisfies the STPRP property, IND$-CPA follows straightforwardly.

Claim 2. $\Pi^{\S}$ satisfies IPR$^+$ for the outer $n-1$ onion layers.

The outer $n-1$ onion layers maintain a non-repeating state u_i through ciphertext cumulating, which is used as a tweak. When the decryption oracle Dec receives the first out-of-sync ciphertext, the STPRP property of the TBC $(\mathtt{E}, \mathtt{D})$ ensures that it is decrypted into a random string. Subsequently, all tweaks on the decryption side will differ from those on the encryption side, and any further out-of-sync ciphertexts will continue to be decrypted into random strings. Therefore, when the TBC $(\mathtt{E}, \mathtt{D})$ satisfies STPRP, IPR$^+$ holds trivially.

Claim 3. $\Pi^{\S}$ does not satisfy INT-sfCTXT for the innermost onion layer.

We construct an adversary $\mathcal{A}$ that easily breaks the INT-sfCTXT experiment.

1. $\mathcal{A}$ queries the $\mathsf{Enc}(\cdot)$ oracle with plaintexts m_1 and m_2, obtaining ciphertexts c_1 and c_2.
2. Then, $\mathcal{A}$ flips the last bit of c_1 and queries the $\mathsf{sfVer}(\cdot)$ oracle with the modified ciphertext. This first out-of-sync ciphertext will fail to decrypt (with a negligible probability of exception), due to the STPRP property of the tweakable block cipher $(\mathtt{E}, \mathtt{D})$.
3. Finally, $\mathcal{A}$ directly queries the $\mathsf{sfVer}(\cdot)$ oracle with c_2. Since the $\mathsf{sfVer}(\cdot)$ oracle uses the same tweak ctr_n as the one used in the Enc oracle for encrypting m_2, c_2 decrypts correctly.

Thus, $\mathcal{A}$ breaks the INT-sfCTXT experiment with $\mathbf{Adv}^{\text{INT-sfCTXT}}_{\mathcal{A},\Pi^{\S}} = 1$.

□