The Onion Router (Tor): Onion Encryption Served Three Ways

Martijn Stam

Simula
UiB

COINS Winterschool in Finse, May 2019

Tor: The Second-Generation Onion Router

Dingledine, Mathewson, Syverson (Usenix'04)

What is Tor

Tor is a tool to advance anonymity on the Internet.

Designers' Aim of Tor

> *Tor seeks to frustrate attackers from linking communication partners, or from linking multiple communications to or from a single user.*

Tor has since grown into a project incl. a browser etc.

Outline

First half

1. **Aspects of Anonymity**

2. **How Tor works**
   - High Level
   - Low Level

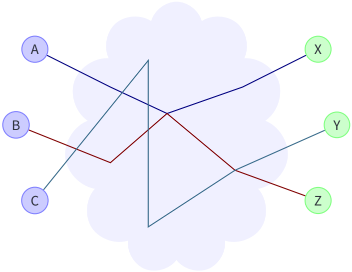
3. **Threats to Tor**
   - Traffic Analysis
   - Tagging Attacks

Outline

Second half

Aims of Anonymity

User-Centric

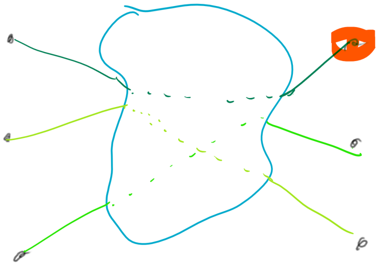


User's Perspective

- Prevent websites from tracking me
- Access web services that are otherwise blocked
- Hide which websites I'm visiting
- Publish a websites without revealing my location etc.

Tracking Users

Prevent websites from tracking me

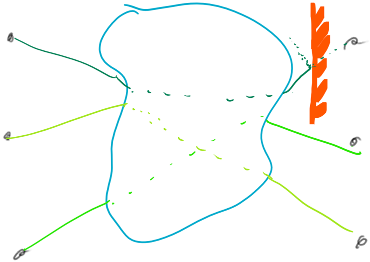


Fingerprinting Websites

- Adversary is the website being visited
- Goals could be identifying or linking users
- This talk: Out of scope
- *TOR-browser* can help protect you

Censoring

Access web services that are otherwise blocked

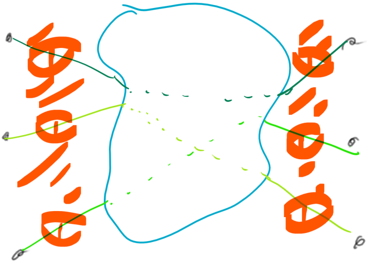


Fingerprinting Websites

- Adversary might be your ISP
- Goals is to filter out “bad” traffic
- This talk: Out of scope
- *Format Transforming Encryption* can help

Deanonymization

Hide which websites I'm visiting

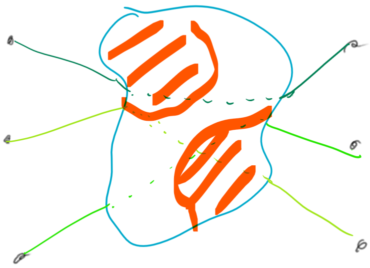


Different Goals

- Deanonymize as much traffic as possible
- Determine users of a specific website
- Determine which websites a specific user visits
- Link users across time and space

Deanonymization

Hide which websites I'm visiting

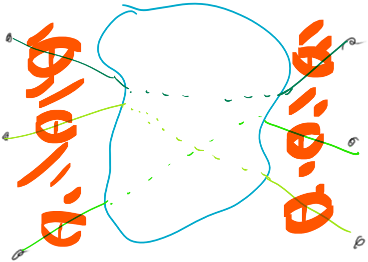


Adversarial Capabilities

- Seeing incoming and outgoing traffic
- Observing part of the network
- Controlling part of the network
- Plus possible some endpoints

Deanonymization

Hide which websites I'm visiting



User Expectations (Hypothetical)

- Noone can see who I am
- Noone can see what I am doing
- Noone can profile me

Tor: The Second-Generation Onion Router

Dingledine, Mathewson, Syverson (Usenix'04)

What is Tor

Tor is a tool to advance anonymity on the Internet.

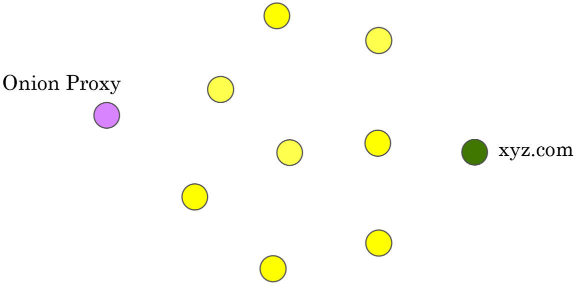
Designers' Aim of Tor

> *Tor seeks to frustrate attackers from linking communication partners, or from linking multiple communications to or from a single user.*

The main principle behind Tor is that of routing internet traffic through multiple hops

Onion Routing

Proxies, Routers, Circuits, and Streams

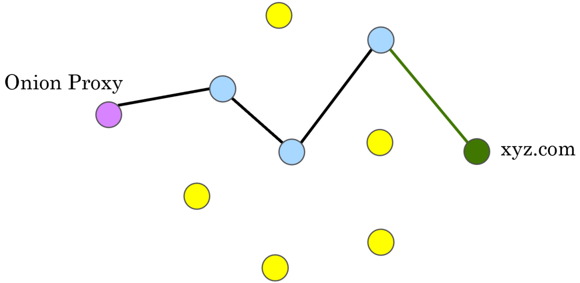


Involved Parties

- **Yellow** these are the *onion routers* comprising the Tor network
- **Purple** the *onion proxy*, run by the client to connect to the network
- **Green** my favourite *destination* or website, which doesn't run Tor

Onion Routing

Proxies, Routers, Circuits, and Streams

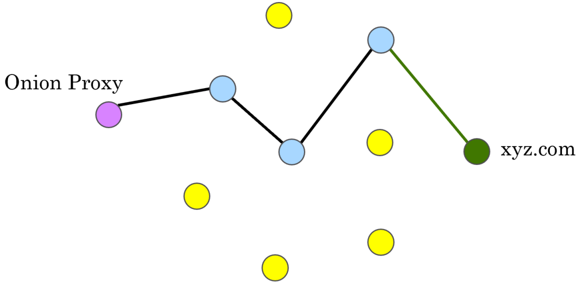


Circuits and Streams

1. The purple proxy knows the yellow routers comprising the Tor network
2. It selects some routers for its blue circuit
3. It runs a TCP stream over the circuit to the destination

Onion Routing

Proxies, Routers, Circuits, and Streams



Principle Idea

- Each hop, or onion router, mixes all the traffic that goes through it
- Ideally, you are hiding amongst the masses:
  if there are enough users and honest routers, you are “safe”

Tor: The Second-Generation Onion Router

Original design decisions

Efficiency

1. *Directory servers*
   Describing known routers and their current state
2. *Congestion control*
   Detect and deal with traffic bottlenecks
3. *Variable exit policies*
   Routers advertise which destinations and ports it supports

Tor: The Second-Generation Onion Router

Original design decisions

Functional

1. *Separation of "protocol cleaning" from anonymity*
   You can use e.g. Privoxy for the "cleaning" instead
2. *Rendezvous points and hidden services*
   Enables anonymously hosted `.onion` websites
3. *Many TCP streams can share one circuit*
   Improves both efficiency and security

Tor: The Second-Generation Onion Router

Original design decisions

Security Related

1. *No mixing, padding, or traffic shaping (yet)*
   Traffic shaping or low-latency mixing that work are hard to come by
2. *Perfect forward secrecy*
   Compromising a router does not reveal anything related to past communication
3. *Leaky-pipe circuit topology*
   The exit node need not be the last one in a circuit
4. *End-to-end integrity checking*
   Prevents “external” tagging attacks

Tor: The Second-Generation Onion Router

Protocol Design

Cryptographic components

Tor has four core protocols

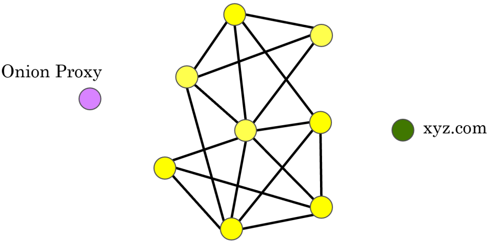
1. Link protocol
2. Circuit Extend protocol
3. Relay protocol
4. Stream protocol

Ignored non-cryptographic components

- How information about the network is distributed
- How onion proxies decide which circuits to build.

Core Tor Specification

Link Protocol (TLS)

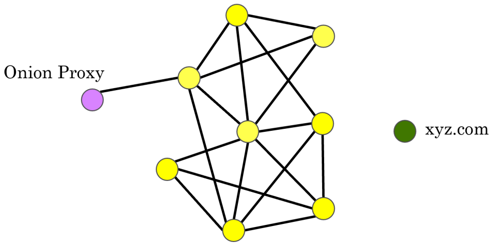


Link protocol

- Agree on Tor version/configuration
- Use TLS to establish secure OR-to-OR channels
- Establish a link from proxy to entry router

Core Tor Specification

Link Protocol (TLS)

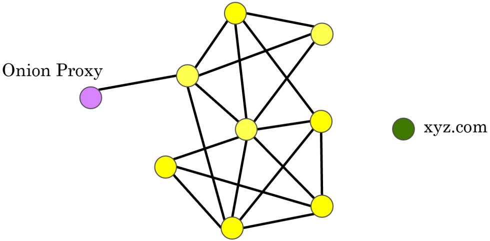


Link protocol

- Agree on Tor version/configuration
- Use TLS to establish secure OR-to-OR channels
- Establish a link from proxy to entry router

Core Tor Specification

Circuit Extend Protocol



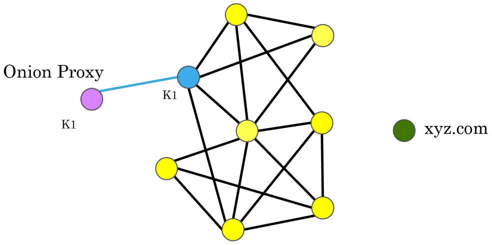
Circuit extend protocol

Used by the onion proxy to create a circuit

- Uses a *telescopic* concept
- Results in the proxy sharing a key with each of its routers

Core Tor Specification

Circuit Extend Protocol



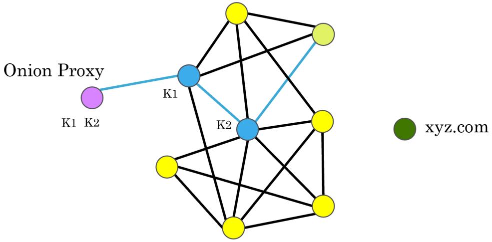
Circuit extend protocol

Used by the onion proxy to create a circuit

- Uses a *telescopic* concept
- Results in the proxy sharing a key with each of its routers

Core Tor Specification

Circuit Extend Protocol



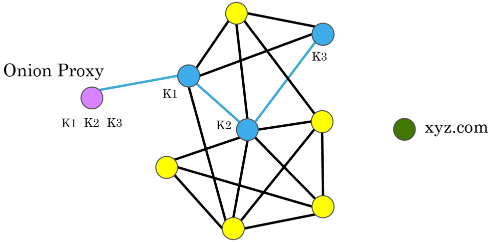
Circuit extend protocol

Used by the onion proxy to create a circuit

- Uses a *telescopic* concept
- Results in the proxy sharing a key with each of its routers

Core Tor Specification

Circuit Extend Protocol



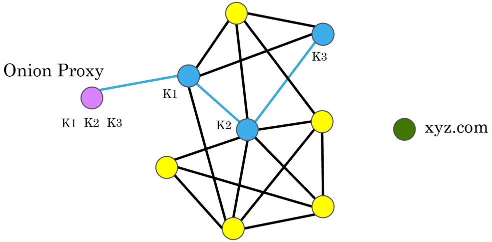
Circuit extend protocol

Used by the onion proxy to create a circuit

- Uses a *telescopic* concept
- Results in the proxy sharing a key with each of its routers

Core Tor Specification

Circuit Extend Protocol



Circuit identifiers

For any given circuit, a router only knows:

1. the key it shares with the anonymous proxy
2. the router preceding and following it on the circuit
3. an incoming and an outgoing circuit identifier

Core Tor Specification

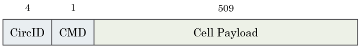
Relay Protocol

Cells are 514 bytes (v4+)

Route

`CircID` Circuit Identifier

`CMD` Cell type (3 or 9) RELAY (3) or RELAY_EARLY



Core Tor Specification

Relay Protocol

Payloads are 509 bytes (v4+)

Encode

- `CircID` Circuit Identifier
- `CMD` Cell type
- `Rec` Recognised field (0x0000)
- `Digest` seeded running hash (truncated SHA-1)

Used for e2e authentication

4	1	509
CircID	CMD	Cell Payload

4	1	1	2	2	4	2	498
CircID	CMD	rCMD	Rec	SID	Digest	Len	Data

Core Tor Specification

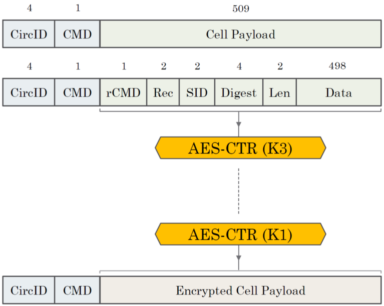
Relay Protocol

Encrypt

Repeated CTR mode in AES

Should provide

- confidentiality
- unlinkability



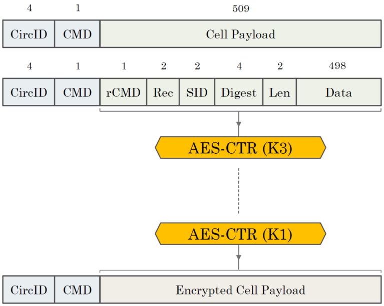
Core Tor Specification

Relay Protocol

Cell Decryption

Performed by Onion Routers

1. Use CircID to identify circuit
2. Undo one AES-CTR layer
3. Check integrity:
   - forward
   - output message
   - reject



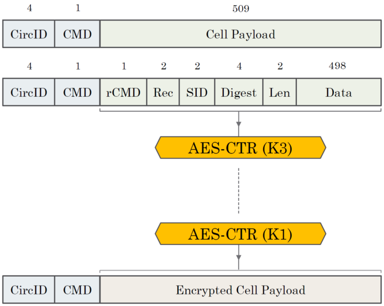
Core Tor Specification

Relay Protocol

Summary

The core cryptographic component is authenticated encryption implemented by

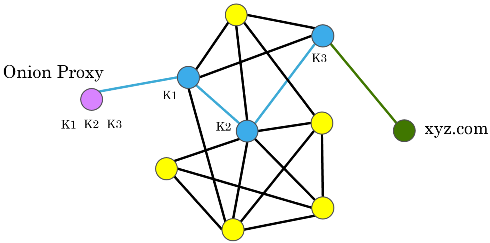
1. encode (Rec and Digest)
2. encrypt (AES-CTR, repeated)



Dodgy mode-of-operation for ordinary AE, but *maybe* ok here?

Core Tor Specification

Stream Protocol



Stream Protocol

Used to serve a TCP connection to host `xyz.com`

Ideally uses `https`-connection between proxy and host

Traffic Analysis

Just a flavour

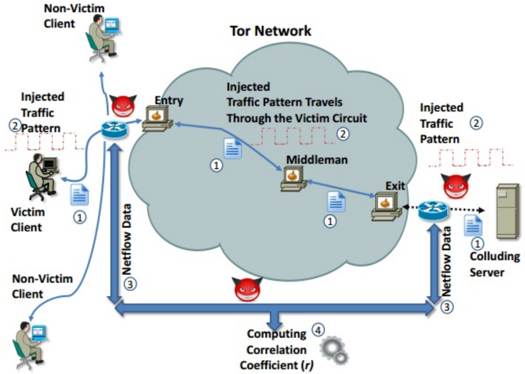
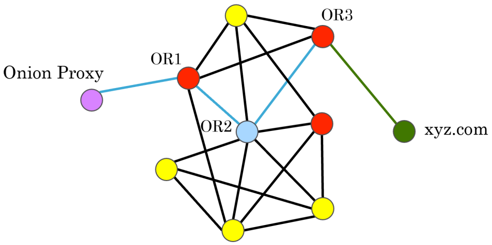


Fig. 1. NetFlow-based traffic analysis against Tor: The client is forced to download a file from the server ①, while the server induces a characteristic traffic pattern ②. After the connection is terminated, the adversary obtains flow data corresponding to the server-to-exit and entry-to-client traffic ③, and computes their correlation coefficient ④.

Source: Chakravarty et al. / PAM 2014

Tagging Attacks

High Level Concept

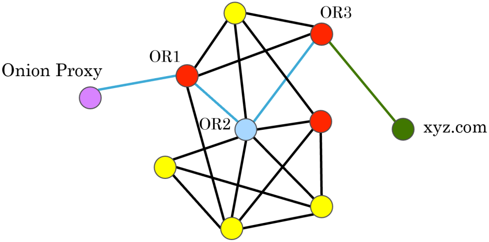


Aim of Tagging Attack

- Assume the adversary controls some onion routers.
- Goal is for *OR1* and *OR3* to link their circuits
- Similar to traffic correlation attacks, where linking is achieved by matching traffic patterns between input and output edges

Tagging Attacks

High Level Concept



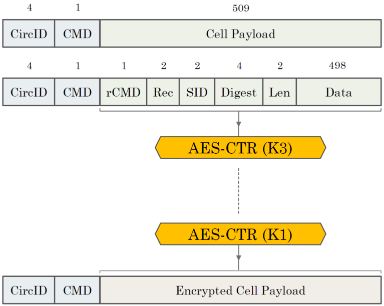
How to Tag

1. *OR1* receives a legitimate cell from the proxy
2. *OR1* processes then modifies the cell before forwarding to *OR2*
3. *OR2* behaves honestly
4. *OR3* detects and undoes *OR1*'s modification

Tagging Attacks
Low Level Details

How to tag

1. *OR1* receives a legitimate cell from the proxy
2. *OR1* processes then modifies the cell before forwarding to *OR2*
3. *OR2* behaves honestly
4. *OR3* detects and undoes *OR1*'s modification



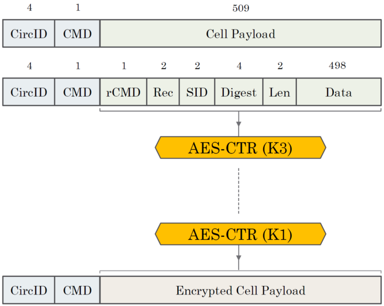
The adversary can confirm whether two edges belong to the same circuit.

Tagging Attacks

Low Level Details

How to tag

1. *OR1* receives a legitimate cell from the proxy
2. OR1 flips a bit in a cell and forwards it over.
3. *OR2* behaves honestly
4. OR3 flips that bit back and tests if decryption succeeds.



Attack works as CTR mode is malleable

Tagging Attacks

Perceptions

2004 Tagging attacks were known to the Tor designers, but protecting against them was deemed pointless since traffic correlation attacks would be possible anyway.

> *“our design is vulnerable to end-to-end timing attacks; so tagging attacks performed within the circuit provide no additional information to the attacker”*

Tagging Attacks

Perceptions

2004 Tagging attacks were known to the Tor designers, but protecting against them was deemed pointless since traffic correlation attacks would be possible anyway.

2008 *The23rd Raccoon:* How I Learned to Stop Ph34ring NSA and Love the Base Rate Fallacy.

2009 Tagging attacks rediscovered by Fu and Ling and presented at Black Hat 2009 - Tor project's response: *Nothing new here!*

2012 *The23rd Raccoon:* Analysis of the Relative Severity of Tagging Attacks. Tor project decides to protect the relay protocol against tagging attacks, leading to Tor proposal 261.

Tagging Attacks

Perceptions

2004 Tagging attacks were known to the Tor designers, but protecting against them was deemed pointless since traffic correlation attacks would be possible anyway.

2008 *The23rd Raccoon:* How I Learned to Stop Ph34ring NSA and Love the Base Rate Fallacy.

2009 Tagging attacks rediscovered by Fu and Ling and presented at Black Hat 2009 - Tor project's response: *Nothing new here!*

2012 *The23rd Raccoon:* Analysis of the Relative Severity of Tagging Attacks. Tor project decides to protect the relay protocol against tagging attacks, leading to Tor proposal 261.

Tagging Attacks

Implications

The23rd Raccoon's Observations

- Consider a network with 10,000 concurrent circuits, and a TC adversary controlling 30% of the entry/exit nodes.
- Due to noise, correlation detectors inevitably exhibit false positives. Let us assume a false positive rate of 0.5%.
- The probability that a pair of edges truly belong to the same circuit when a match is detected is $\sim$2% (base rate fallacy).
- This effect becomes more pronounced as the number of circuits increases, but tagging attacks are immune to this.
- The 2012 post describes an amplification effect and argues that tagging attacks require less resources.

Tagging Attacks

Thwarting

Recap

Tagging attacks are enabled by

- the malleability of counter mode encryption
- the integrity checking being end-to-end only

Tagging Attacks

Thwarting

Recap

Tagging attacks are enabled by

- the malleability of counter mode encryption
- the integrity checking being end-to-end only

Intermediate Integrity Checking

- A naive fix would be to append a MAC tag at each layer of encryption, but this leaks information!
- This leakage can be prevented with appropriate padding to ensure the cell size is constant throughout.

Tagging Attacks

Thwarting

Recap

Tagging attacks are enabled by

- the malleability of counter mode encryption
- the integrity checking being end-to-end only

Improved Modes-of-Operation

An alternative approach, resulting in a higher throughput, is to depart from counter mode

- Proposal 261 (Mathewson)
- Proposal 295 (Ashur, Dunkelman, Luykx)

Thwarting Tagging Attacks

Proposal 261 by Mathewson

1. Digest set to 0x00000000
2. AES-CTR replaced by TWBC

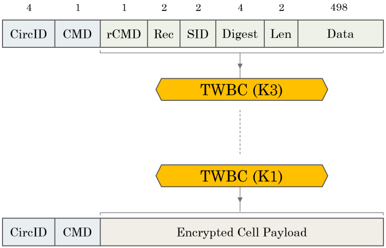
4	1	1	2	2	4	2	498
CircID	CMD	rCMD	Rec	SID	Digest	Len	Data

3. Verification checks a total 55 bits
4. End-to-end integrity via encode-then-encipher.

Thwarting Tagging Attacks

Proposal 261 by Mathewson

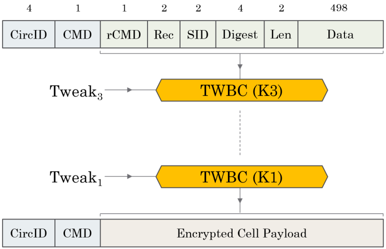
1. Digest set to 0x00000000
2. AES-CTR replaced by TWBC
   - Separate tweak per layer, updated with each cell.
   - Tweak includes CMD (RELAY or RELAY_EARLY).
3. Verification checks a total 55 bits
4. End-to-end integrity via encode-then-encipher.



Thwarting Tagging Attacks

Proposal 261 by Mathewson

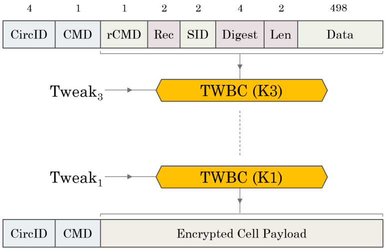
1. Digest set to 0x00000000
2. AES-CTR replaced by TWBC
   - Separate tweak per layer, updated with each cell.
   - Tweak includes CMD (RELAY or RELAY_EARLY).
3. Verification checks a total 55 bits
4. End-to-end integrity via encode-then-encipher.



Thwarting Tagging Attacks

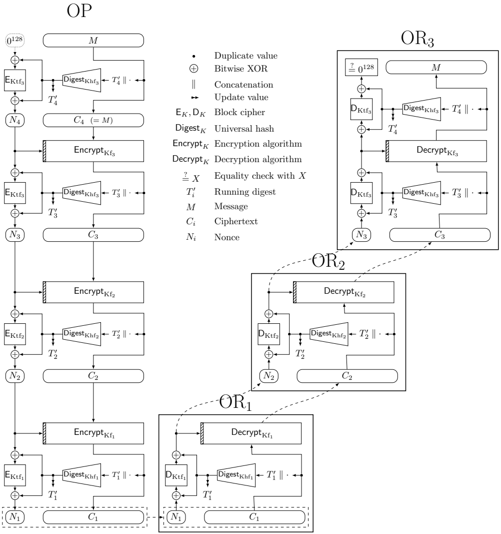
Proposal 261 by Mathewson

1. Digest set to 0x00000000
2. AES-CTR replaced by TWBC
   - Separate tweak per layer, updated with each cell.
   - Tweak includes CMD (RELAY or RELAY_EARLY).
3. Verification checks a total 55 bits
4. End-to-end integrity via encode-then-encipher.



Thwarting Tagging Attacks II

Proposal 295 by Ashur, Dunkelman, Luykx



Questions so Far?

(Plus a microbreak)

?

Outline of Part II

Real World Crypto Sandwich

Keywords

Deploy
- Many users with passage of time
- User-centric APIs including key management

Realize
- One or two users only, frozen in eternity
- Designer-centric with trusted device and keys

Implement
- Single device under adversarial attack
- Concrete efficiency measures, e.g. cycles/byte

Real World Crypto Sandwich

Keywords

Deploy
- Articulated threat models (assets and adversaries)
- Some formal verification possible

Realize
- Abstract security models (UC or game-based)
- Cryptographic combinatorics/reductions for concrete bounds

Implement
- Attacks, countermeasures and validation
- Strong focus on key recovery

Real World Crypto Sandwich

Keywords

Deploy

- Security models might not match threat models (too weak)
- Concrete bounds do not scale well

Realize

- Security models might not match threat models (too strong)
- Key recovery security is necessary, not sufficient

Implement

- Mapping security models to physical world problematic
- Concrete bounds (for key recovery) often weak

Real World Crypto Sandwich

Keywords

Deploy
- Hope that modularity leads to concrete composability
- Include more details into security models

Realize
- Consider multi-user multi-query security
- Care about concrete bounds

Implement
- Ensure deployment deals with key compromises
- ???

Modeling Tor

How cryptology can help protect you!

State of play

- Countermode TOR is susceptible to tagging attacks.
- TOR-261 and TOR-295 are designed to prevent tagging attacks.

But do they?

1. What security is breached by tagging attacks?
2. Can we formally define the relevant security?
3. Can we prove TOR-261 and TOR-295 are secure?

Modeling Tor

How cryptology can help protect you!

Ideal of provable security

Given a secure TWBC, TOR-261 is a secure onion encryption scheme

Reality of provable security

Why provably secure constructions may get broken in practice

Proof The security claim is incorrect
Solutions: automated proof checking, modularity of proofs

Bound The security claim is quantitively too weak
Solution: derive concrete multi-user bounds

Model The security claim is qualitatively too weak
Solution: carefully refine the model

Modeling Tor

How cryptology can help protect you!

Abstraction Levels

Tor exists in different levels of granularity:

1. Tor aims to implement an anonymous channel
2. Using the principles of onion routing
3. Based on the Tor standard
4. As implemented in Tor software

A security model needs to decide which details are pertinent

Choice 1: Abstraction level

Different levels of abstractions lead to models with varying scope and relevance to practice

Modeling Tor

How cryptology can help protect you!

Tor Use Cases

Tor aims to improve privacy and security on the Internet in a variety of ways. People use Tor to

- Keep websites from tracking them
- Access web services that are otherwise blocked
- Hide which websites are visited
- Publish websites without revealing their location

Choice 2: Security goal

Different aims might call for different orthogonal security models

Modeling Tor

How cryptology can help protect you!

Adversarial capabilities

Imagine an adversary:

- Controlling part of the network
- Correlating traffic
- Injecting/modifying traffic

Choice 3: Adversarial powers

Different threat models lead to more or less potent security models

Modeling Tor

How cryptology can help protect you!

Modeling Choices

Abstraction Which aspects of the protocol are modelled

Aim What is an adversary trying to achieve

Capability What powers does an adversary have

Two models capturing tagging attacks

PETS More abstract, less powerful adversaries, cleaner

Eurocrypt More detailed, more powerful adversaries, messier

How do results in your model relate to real world deployment?

PETS Model

Rogaway and Zhang (2018)

Modeling authenticated onion encryption

Goal distinguish an onion encryption scheme from an idealized primitive

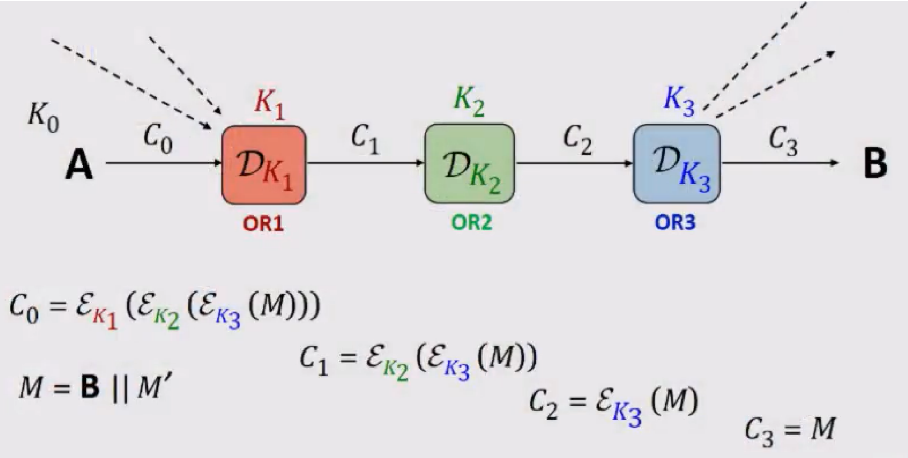
Powers querying the keyed component algorithms

Assumptions

- keys are magically pre-distributed (extend protocol)
- cell routing is out of scope (relay protocol)
- ignore streams (stream protocol)

PETS model

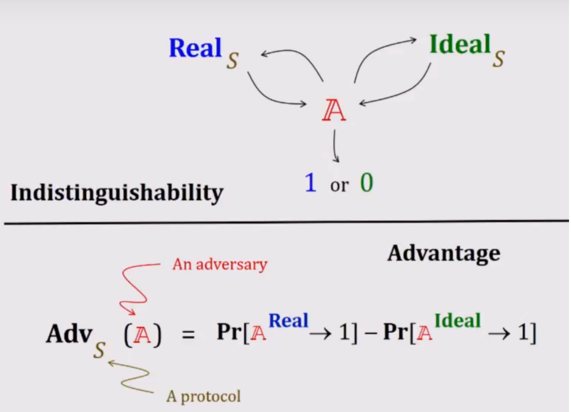
Syntax



Source: Phil Rogaway, PETS 2018

PETS model

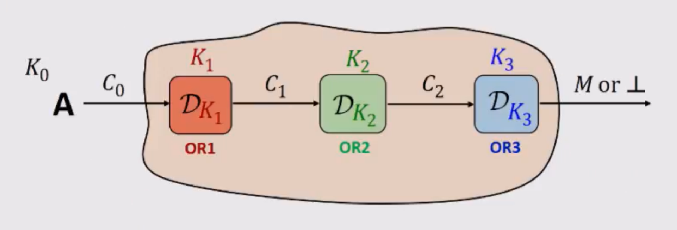
Security



Source: Phil Rogaway, PETS 2018

PETS model

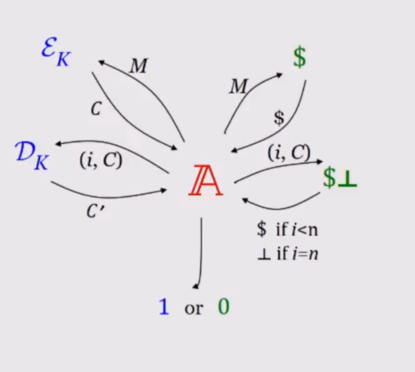
Security



Source: Phil Rogaway, PETS 2018

PETS model

Security



Source: Phil Rogaway, PETS 2018

Eurocrypt Model

Degabriele and Stam (2018)

Modeling the relay protocol

Goal learn information about the circuits' topology beyond what is inevitably leaked through node corruptions

Powers choose the messages that get encrypted; reorder, inject, and manipulate cells on the network; selectively corrupt routers

Assumptions

- keys are magically pre-distributed (extend protocol)
- node-to-node links are secured (link protocol)
- ignore streams (stream protocol)

Eurocrypt Model

Syntax

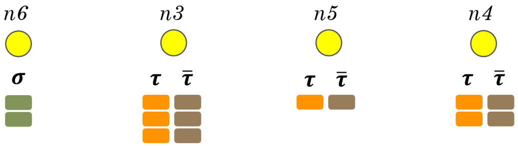
Setting

Consider a circuit with

- an onion proxy: *n6*
- (here) three onion routers: *n3*, *n5* and *n4*

Eurocrypt Model

Syntax

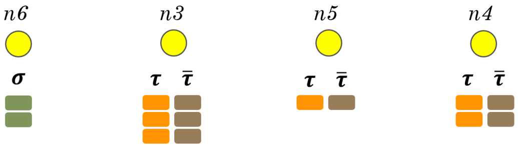


Party's State

- A party's state is circuit-based: for each circuit it keeps some state
- For onion *routers*, this state is split in two: a routing component and a processing component

Eurocrypt Model

Syntax

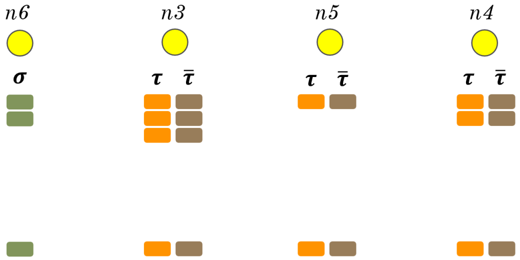


Four algorithms

1. G for key generation
2. E for encryption
3. D for routing
4. $\bar{D}$ for decryption

Eurocrypt Model

Syntax

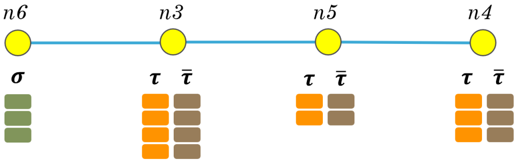


G for key generation

1. Initiated by proxy on input the path of the circuit
2. The proxy and the router obtain state information for the new circuit
3. The new information is added to their respective states so far

Eurocrypt Model

Syntax

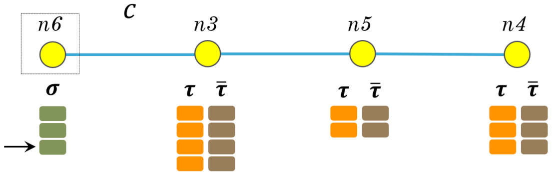


G for key generation

1. Initiated by proxy on input the path of the circuit
2. The proxy and the router obtain state information for the new circuit
3. The new information is added to their respective states so far

Eurocrypt Model

Syntax

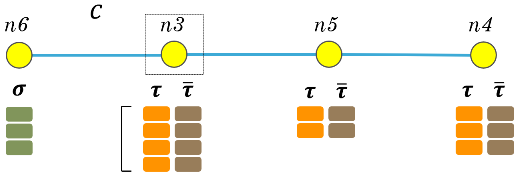


E for encryption

- Run by the proxy
- As input the state of the relevant circuit
- And some message m
- Results in a cell C for first router on circuit

Eurocrypt Model

Syntax

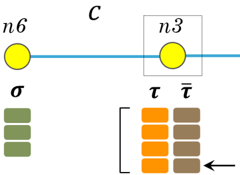


D for routing

- Run by router when receiving a cell C
- To identify which circuit the cell belongs to
- Use the first part τ of all circuit states
- Leave the states τ untouched

Eurocrypt Model

Syntax

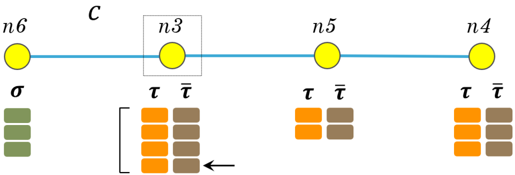


$\bar{D}$ for decryption

- Run by router when processing a cell C
- Using the $\bar{\tau}$ part of the relevant circuit state
- Results deterministically in $\perp$, M or C'
- May update the circuit state $\bar{\tau}$

Eurocrypt Model

Syntax

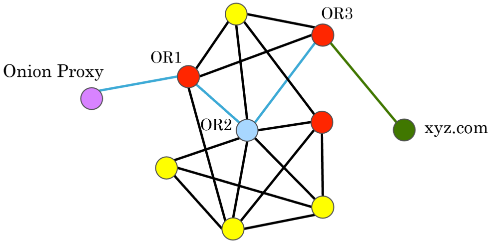


Why the vector of split states?

- We want to include circuit routing in our model
- We want to model the problem, not Tor's solution
- We do not want too much interference between circuits

Secure Channel

Confidentiality and Integrity



Left-or-Right End-to-End Indistinguishability

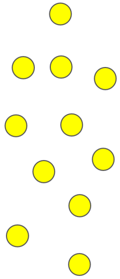
An adversary with all-but-one decryption keys of a circuit cannot distinguish whether m_0 or m_1 was encrypted by an onion proxy

Plaintext Integrity

An adversary cannot trick a router into outputting an message out of order

Circuit Hiding

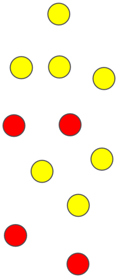
Left-or-Right Topology Indistinguishability



Let’s consider a network of onion routers

Circuit Hiding

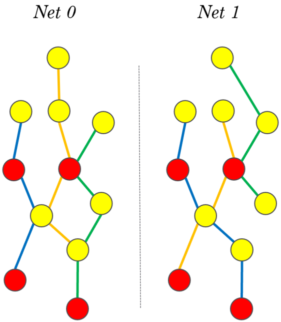
Left-or-Right Topology Indistinguishability



The adversary gets to *corrupt* some of the routers

Circuit Hiding

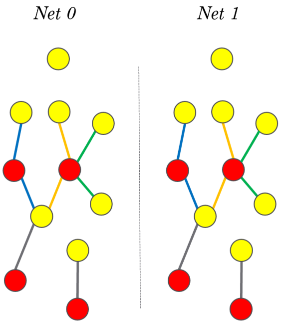
Left-or-Right Topology Indistinguishability



The adversary selects *two* sets of potential circuits
the game implements either the left-or-right configuration

Circuit Hiding

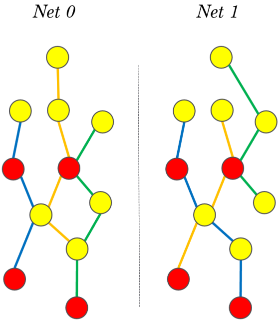
Left-or-Right Topology Indistinguishability



Both configurations need to “coincide on” the corrupted routers

Circuit Hiding

Left-or-Right Topology Indistinguishability



The adversary gets to interact with the honest nodes in a restricted fashion

Is it in the left or right configuration?

Circuit Hiding

Left-or-Right Topology Indistinguishability

Intricacies

- Many controls to ensure interface is the same
- So length of circuit and node's relative position remain hidden
- Protects against reordering and replay of cells
- Cells need to be re-injected simultaneously, one for each circuit
- Adversary may corrupt at most two segments of a circuit

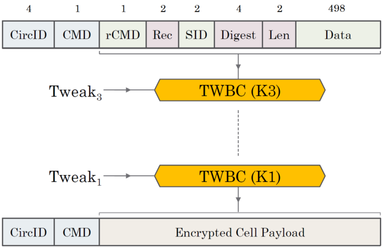
The adversary gets to interact with the honest nodes in a restricted fashion
Is it in the left or right configuration?

Circuit Hiding

Proposal 261

P261 is not circuit hiding

- Use the cell header's CMD field to tag cells, by switching its value from RELAY to RELAY_EARLY
- Authentication of CMD in the tweak is ineffective
- Similarities to the 2014 CMU incident on Tor's Onion Services which took down Silk Road.

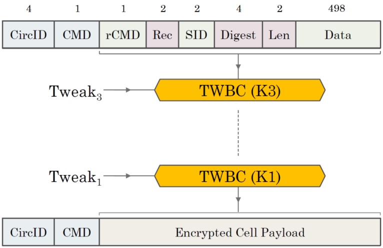


Circuit Hiding

Proposal 261

P261 *almost* circuit hiding

- Practical exploitability and efficacy of this attack is limited
- RELAY_EARLY cell type limits the circuit size and its use is restricted
- Fixing CMD to RELAY provides provable circuit hiding



Comparison

Eurocrypt versus PETS models

Commonalities

- Target the core relay protocol
- To prevent tagging attacks
- Consider only unidirectional traffic
- Ignore leaky pipes
- Abstract away key generetion
- Use game-based formalization

Comparison

Eurocrypt versus PETS models

Differences

Eurocrypt	*v.*	*PETS*
Protocol-centric		Primitive-centric
Includes routing		Excludes routing
Multi-user		Single-user
Includes Corruptions		No Corruptions
Aspirational		Best-possible
End-to-end security		Cell security
Explicit suppression		Silencing

Challenges

Quantify the power of tagging attacks more rigourously

Find middle-ground between the PETS and the Eurocrypt models

Prove the security of Proposal 295

Improve upon existing proposals

Expand the provable security treatment to include

- the other protocols and bidirectionality
- other security objectives (e.g. forward security)

Conclusion

Modeling Tor

Onion encryption can be modelled in various ways

- The Eurocrypt model identified circuit hiding as its anonymity goal
- The PETS model gave an authenticated encryption treatment instead

The Eurocrypt model shows that the routing mechanism affects anonymity

Abstraction is a double-edged sword

> *the next step in an ongoing evolution of most appropriate and important onion routing adversaries, away from abstracting reality till it matches models and towards better matching models to reality*

—Syverson