

# Obscuro: A Bitcoin Mixer using Trusted Execution Environments

*Muoi Tran*, Loi Luu, Min Suk Kang, Iddo Bentov, and Prateek Saxena  
ACSAC 2018 | December 3–7 | San Juan, Puerto Rico, USA



Bitcoin addresses are *linkable*



# Bitcoin addresses are *linkable*

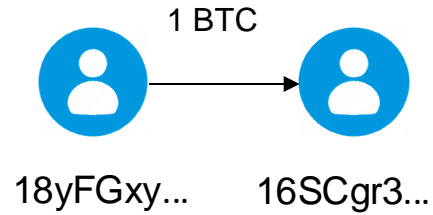


18yFGxy...

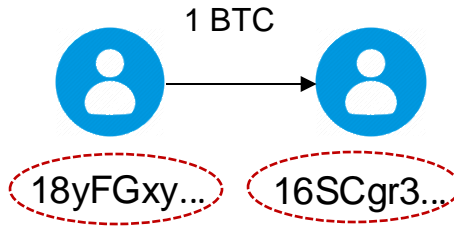


16SCgr3...

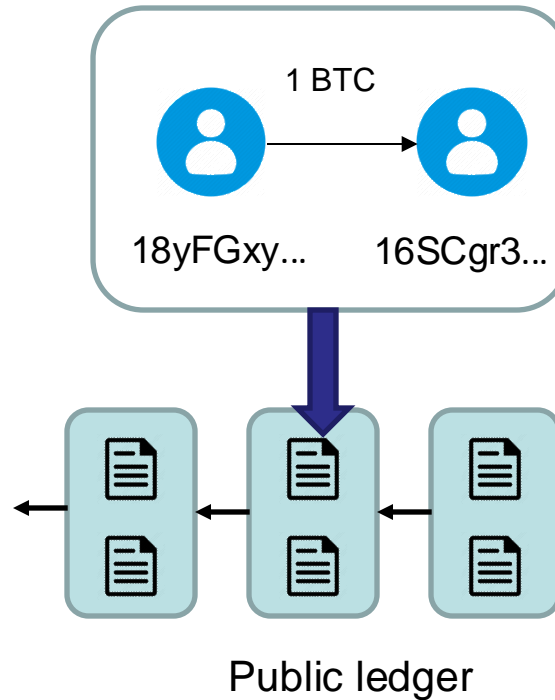
# Bitcoin addresses are *linkable*



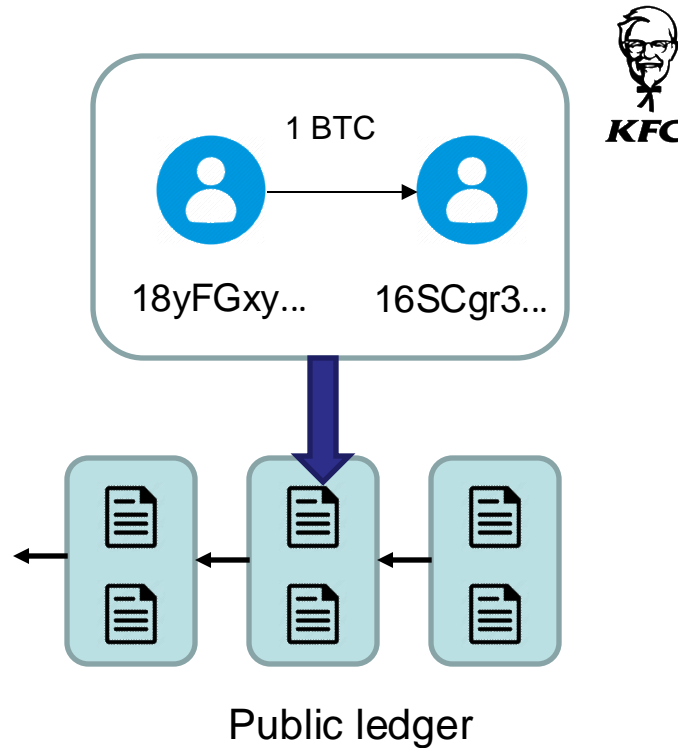
# Bitcoin addresses are *linkable*



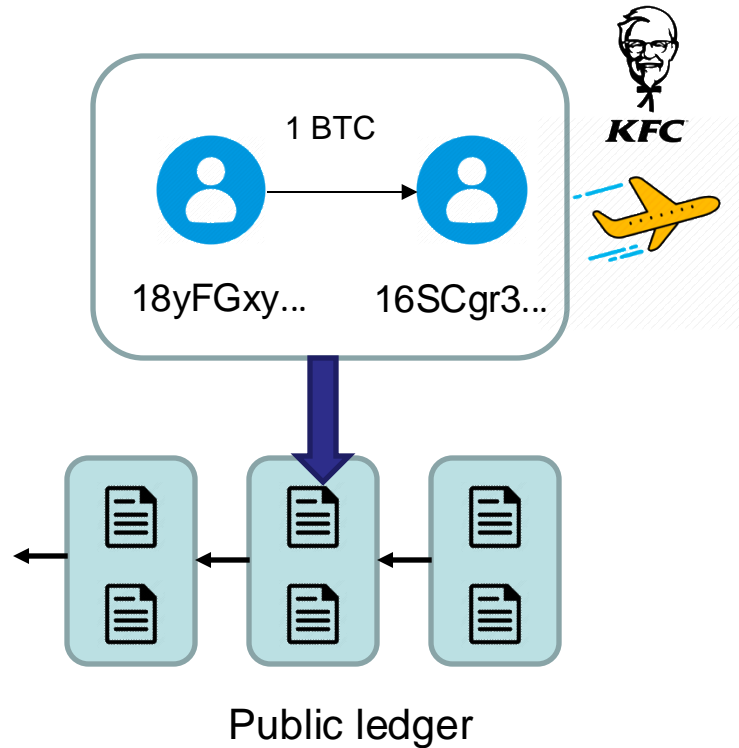
# Bitcoin addresses are *linkable*



# Bitcoin addresses are *linkable*

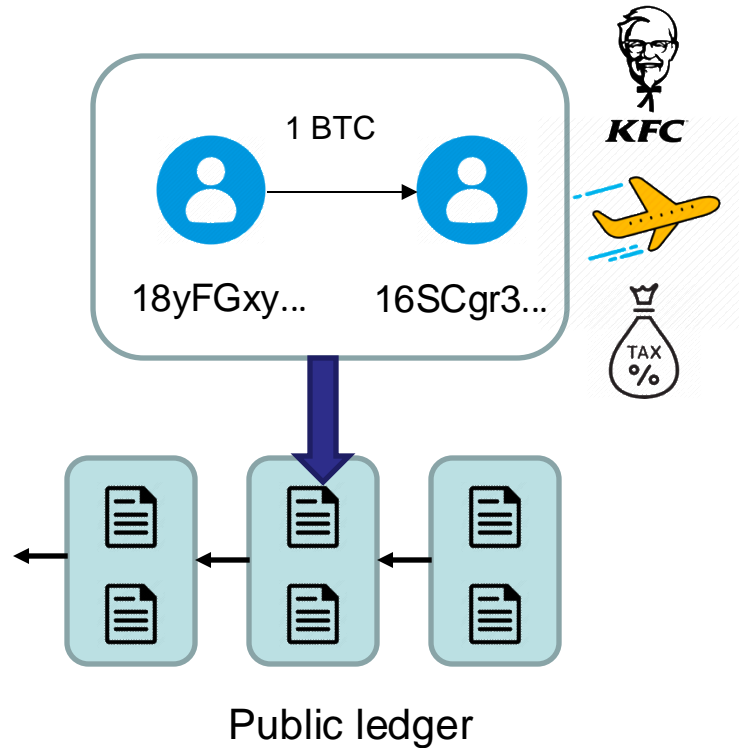


# Bitcoin addresses are *linkable*

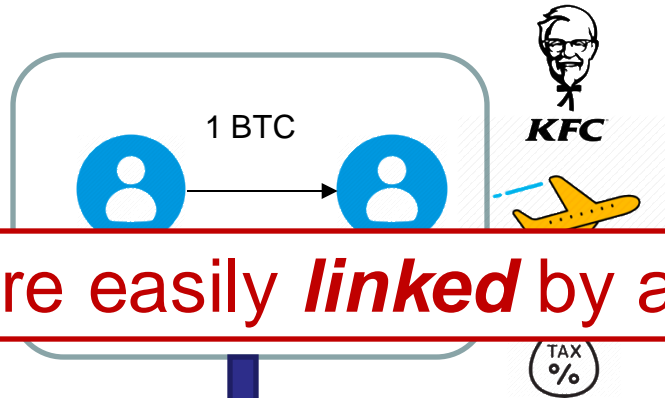




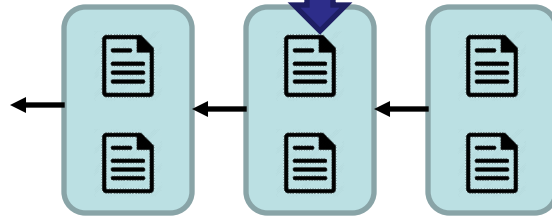
# Bitcoin addresses are *linkable*



# Bitcoin addresses are *linkable*

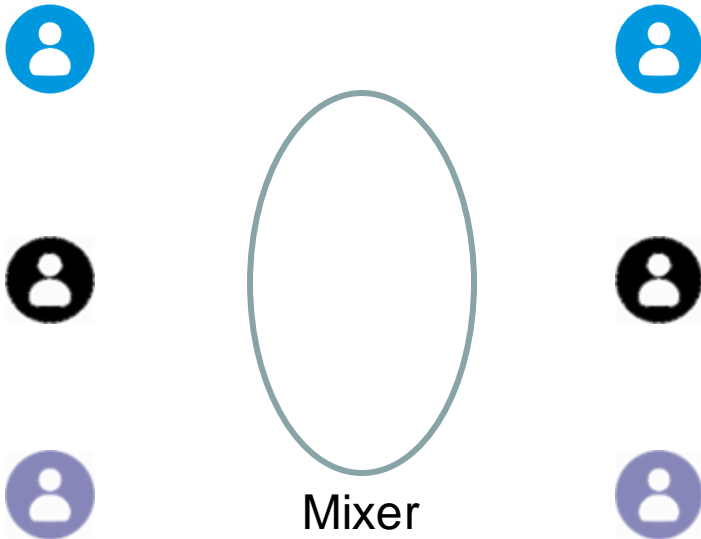


Addresses are easily *linked* by anyone!

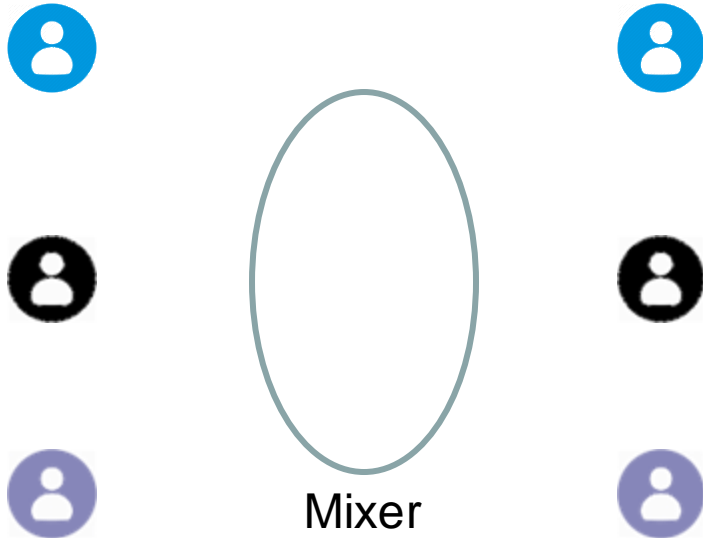


Public ledger

# *Mixer* makes addresses *unlinkable*

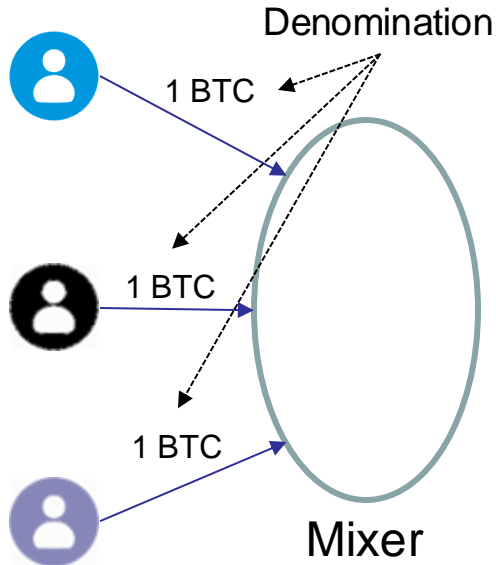


# Mixer makes addresses *unlinkable*



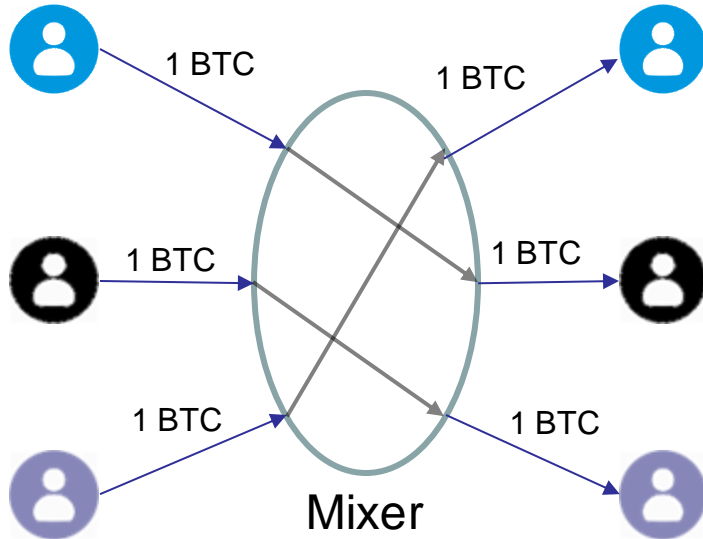
- Multiple senders submit *denomination* coins to their *randomly shuffled* recipients

# Mixer makes addresses *unlinkable*



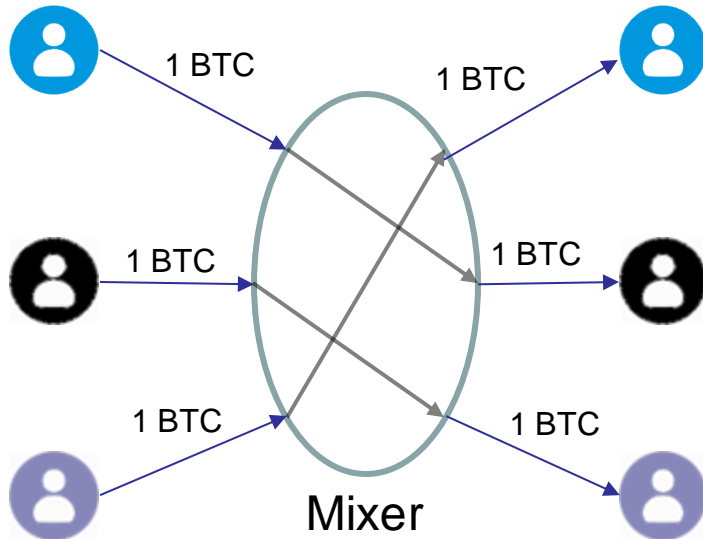
- Multiple senders submit ***denomination*** coins to their ***randomly shuffled*** recipients

# Mixer makes addresses *unlinkable*



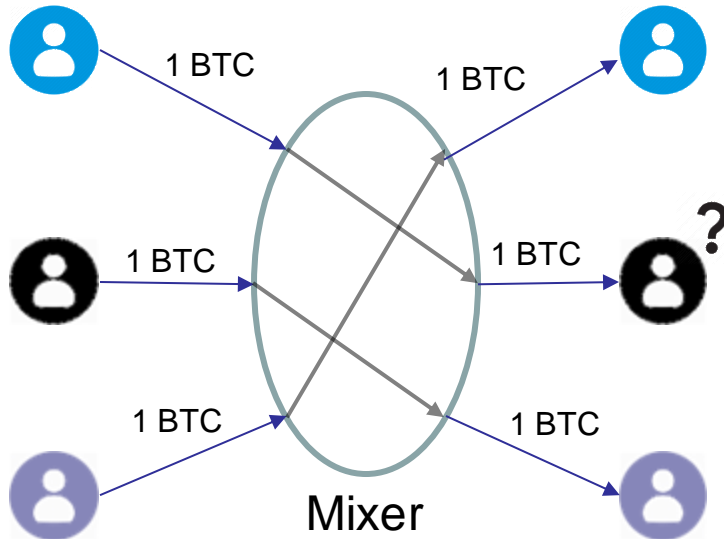
- Multiple senders submit **denomination** coins to their **randomly shuffled** recipients

# Mixer makes addresses *unlinkable*



- Multiple senders submit ***denomination*** coins to their ***randomly shuffled*** recipients
- Difficult to identify sender of a given recipient:

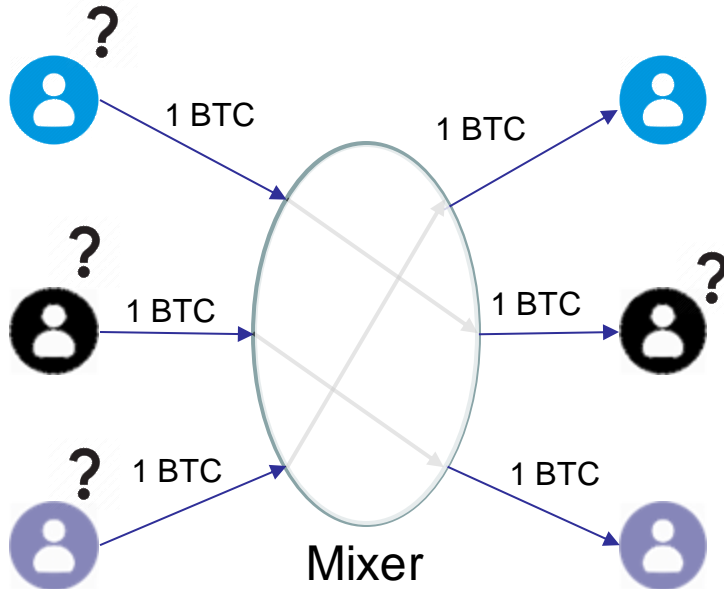
# Mixer makes addresses *unlinkable*



- Multiple senders submit **denomination** coins to their **randomly shuffled** recipients
- Difficult to identify sender of a given recipient:



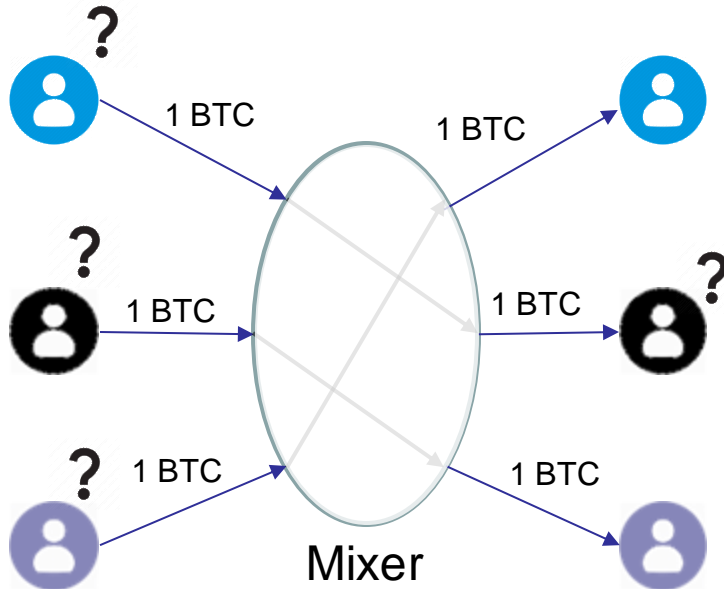
# Mixer makes addresses *unlinkable*



- Multiple senders submit **denomination** coins to their **randomly shuffled** recipients
- Difficult to identify sender of a given recipient:

***Probability =  $1/\{\# \text{ users in a mix}\}$***

# Mixer makes addresses *unlinkable*



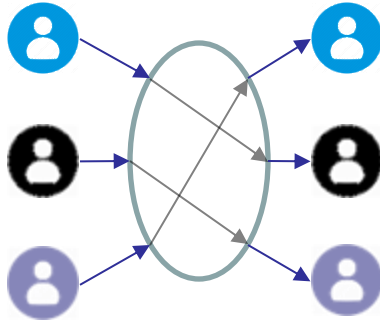
- Multiple senders submit **denomination** coins to their **randomly shuffled** recipients
- Difficult to identify sender of a given recipient:

**Probability =  $1/\{\# \text{ users in a mix}\}$**

**Security parameter:  
Anonymity set size, or the number of users per mixing round**

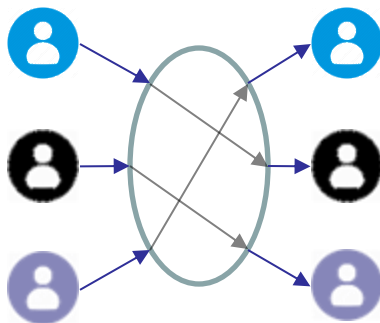
# Two categories of existing mixers

# Two categories of existing mixers



Centralized mixer

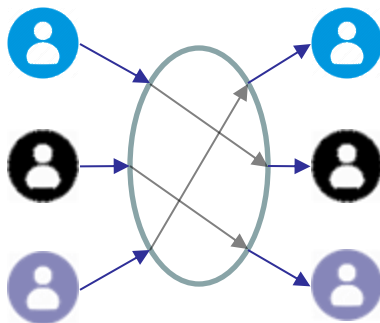
# Two categories of existing mixers



Centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)

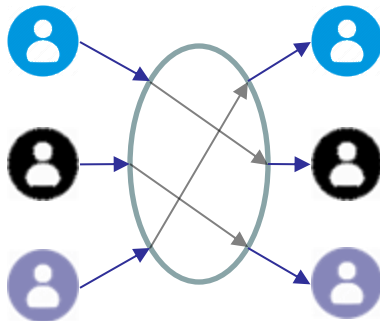
# Two categories of existing mixers



Centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***

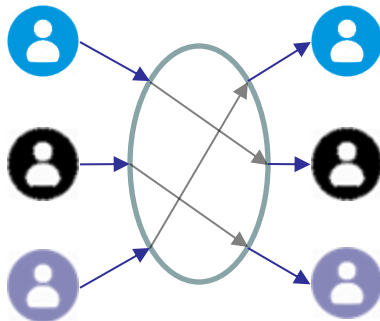
# Two categories of existing mixers



Centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation

# Two categories of existing mixers

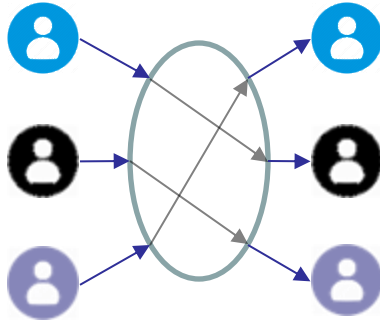


Centralized mixer

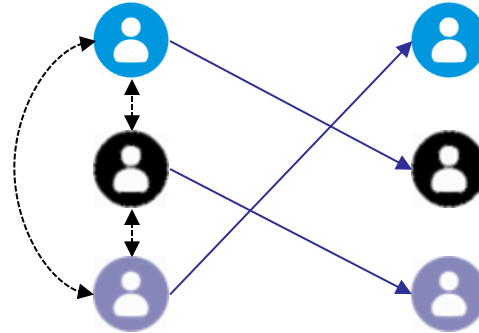
- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])



# Two categories of existing mixers



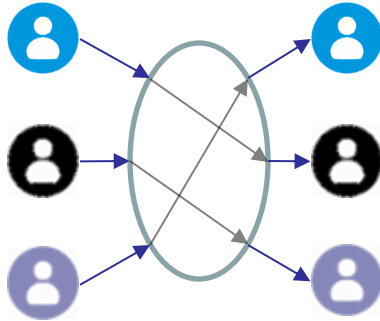
Centralized mixer



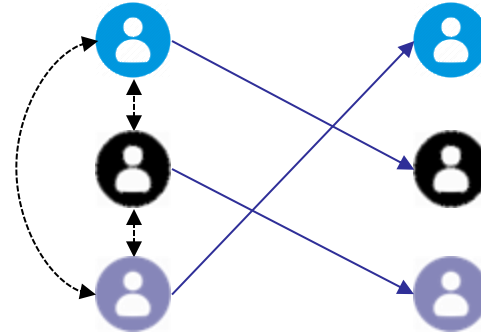
De-centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])

# Two categories of existing mixers



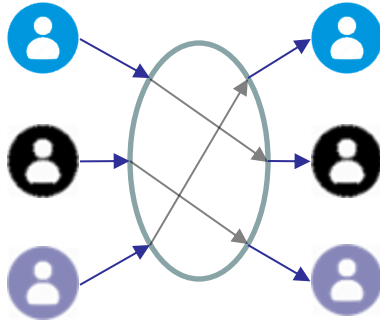
Centralized mixer



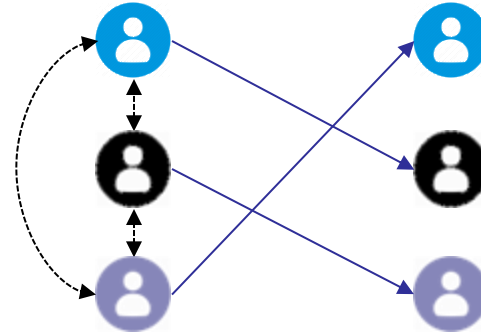
De-centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])
- Not rely on any centralized party

# Two categories of existing mixers



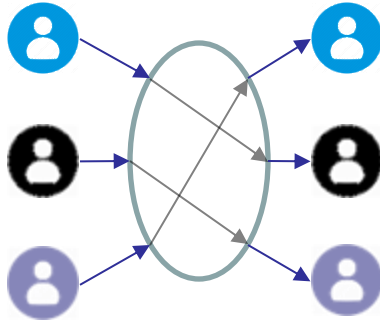
Centralized mixer



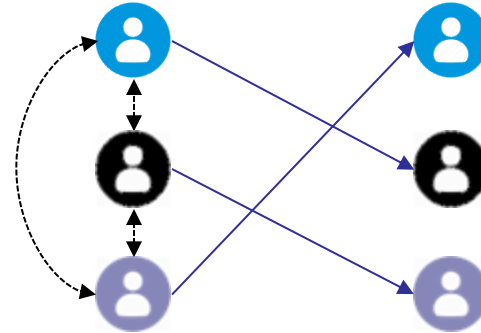
De-centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])
- Not rely on any centralized party
- ***High*** communication overhead

# Two categories of existing mixers



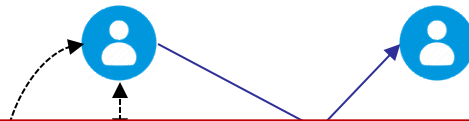
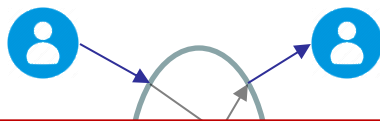
Centralized mixer



De-centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])
- Not rely on any centralized party
- ***High*** communication overhead
  - ✓ E.g., 100 users may take ***21s to mins (or even longer)*** to mix using CoinShuffle++ [NDSS'17]

# Two categories of existing mixers



Existing Bitcoin mixers are either  
***insecure*** or ***inefficient!***

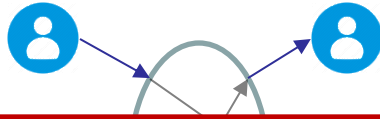
## Centralized mixer

- Simple architecture, large anonymity set size (e.g., thousands)
- Mixer operator may ***misbehave***
  - ✓ ***Steal*** coin or ***leak*** permutation
  - ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])

## De-centralized mixer

- Not rely on any centralized party
- ***High*** communication overhead
  - ✓ E.g., 100 users may take ***21s to mins (or even longer)*** to mix using CoinShuffle++ [NDSS'17]

# Two categories of existing mixers



Existing Bitcoin mixers are either  
***insecure*** or ***inefficient!***

Centralized mixer

De-centralized mixer

Can we have a Bitcoin mixer that is both  
***secure*** and ***efficient?***

- ✓ Partial solutions exist (e.g., TumbleBit [NDSS'17])

... (or even longer) to mix using CoinShuffle++ [NDSS'17]

# Obscuro: centralized mixer protected by *Trusted Execution Environments*



# Obscuro: centralized mixer protected by *Trusted Execution Environments*



- Trusted Execution Environments (TEEs)



# Obscuro: centralized mixer protected by *Trusted Execution Environments*



Verifier

- Trusted Execution Environments (TEEs)

# Obscuro: centralized mixer protected by *Trusted Execution Environments*



Verifier

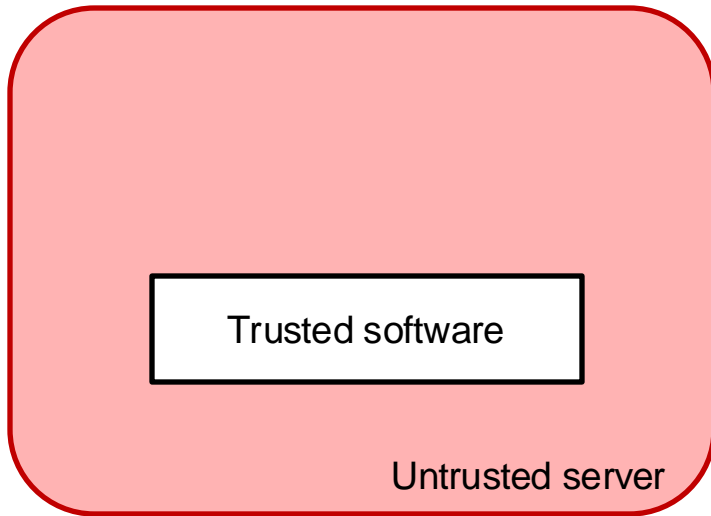
- Trusted Execution Environments (TEEs)

Trusted software

# Obscuro: centralized mixer protected by *Trusted Execution Environments*



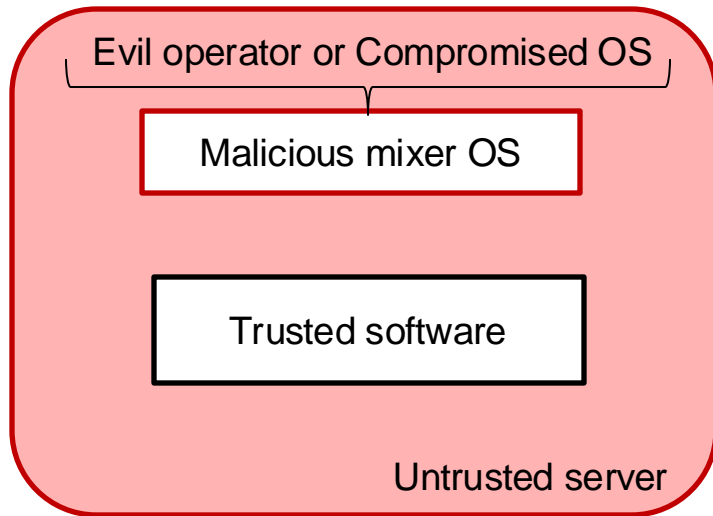
- Trusted Execution Environments (TEEs)



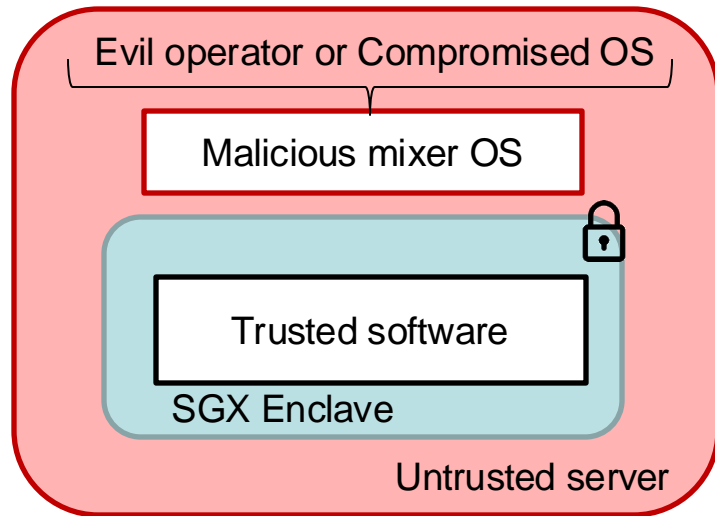
# Obscuro: centralized mixer protected by *Trusted Execution Environments*



- Trusted Execution Environments (TEEs)

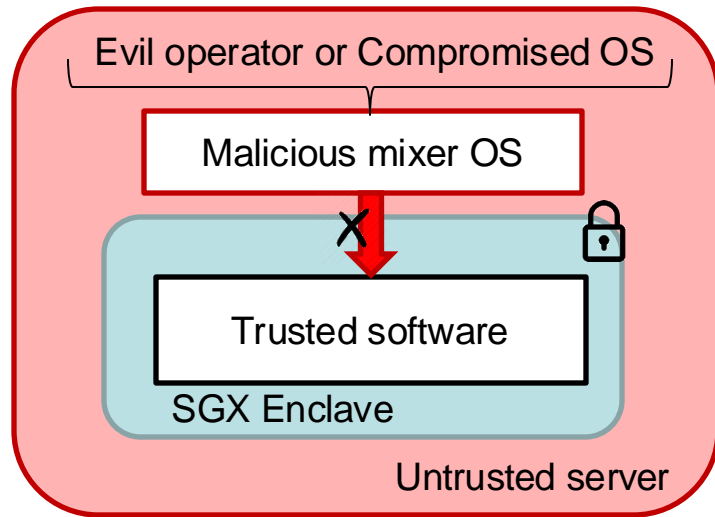


# Obscuro: centralized mixer protected by *Trusted Execution Environments*



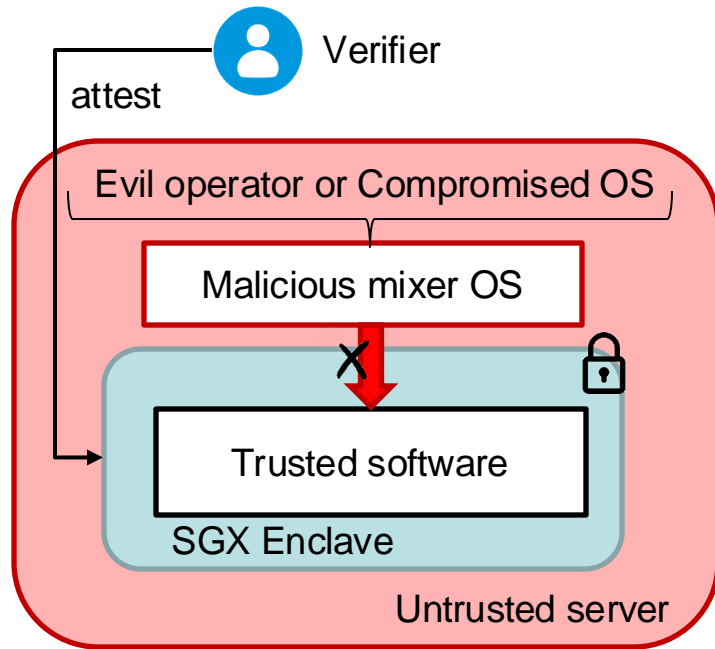
- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**

# Obscuro: centralized mixer protected by *Trusted Execution Environments*



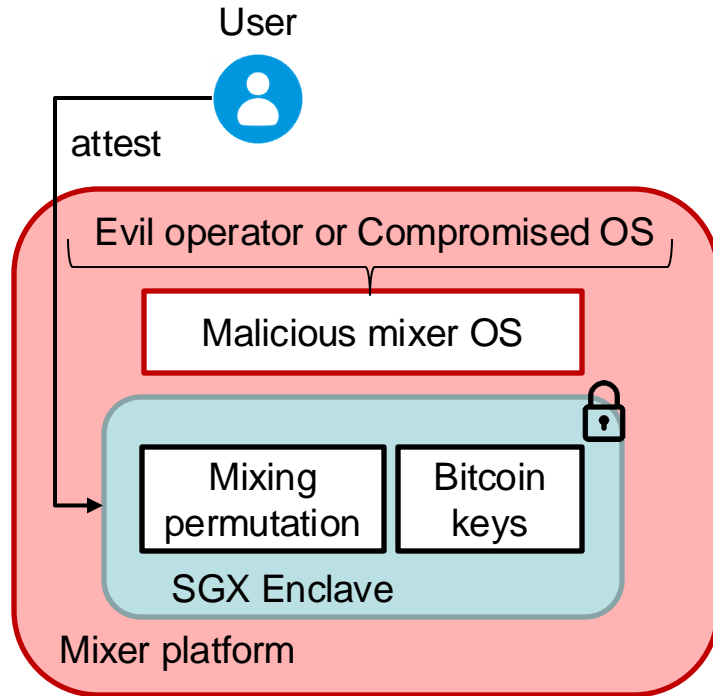
- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**

# Obscuro: centralized mixer protected by *Trusted Execution Environments*



- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**
  - ✓ verifiable via **remote attestation**

# Obscuro: centralized mixer protected by *Trusted Execution Environments*

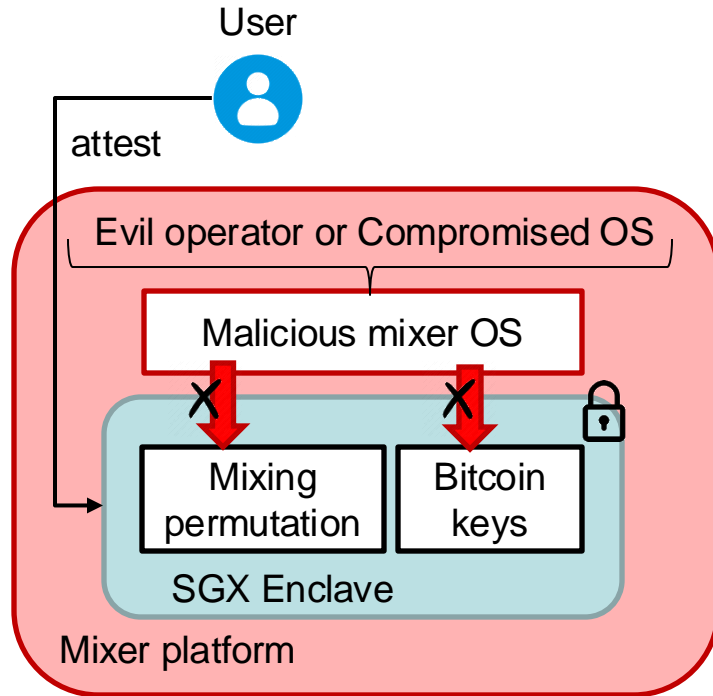


**Naïve Obscuro Model**

- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**
  - ✓ verifiable via **remote attestation**
- **TEE-based mixer** addresses **simple** attacks



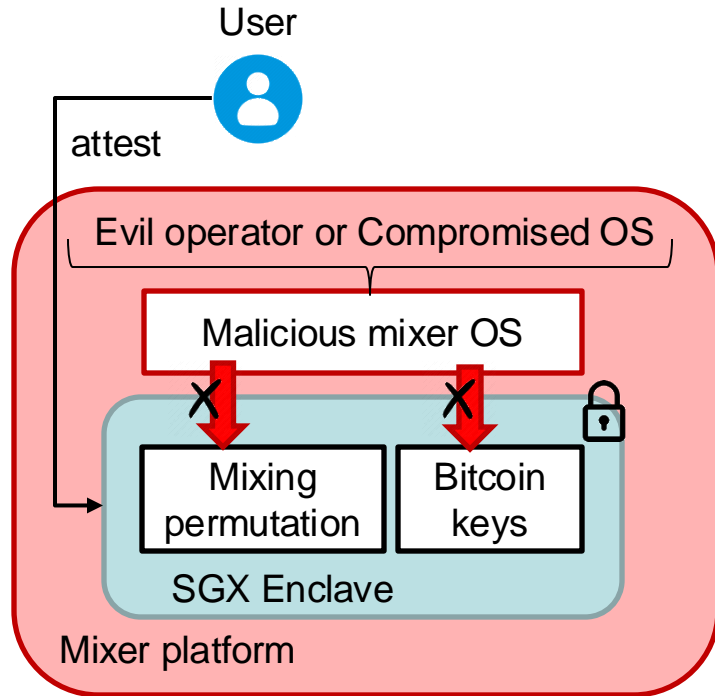
# Obscuro: centralized mixer protected by *Trusted Execution Environments*



**Naïve Obscuro Model**

- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**
  - ✓ verifiable via **remote attestation**
- **TEE-based mixer** addresses **simple** attacks
  - ✓ malicious mixer OS cannot **steal coins** or **leak** the mixing permutation

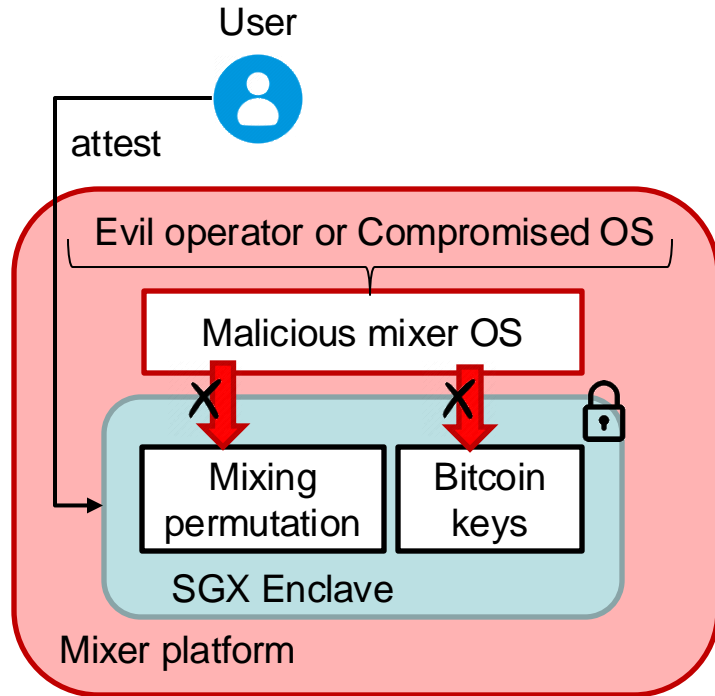
# Obscuro: centralized mixer protected by *Trusted Execution Environments*



**Naïve Obscuro Model**

- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data in a *protected memory region*, called an **enclave**
  - ✓ verifiable via **remote attestation**
- **TEE-based mixer** addresses **simple** attacks
  - ✓ malicious mixer OS cannot **steal coins** or **leak** the mixing permutation
  - ✓ users verify Obscuro's execution **before** they join

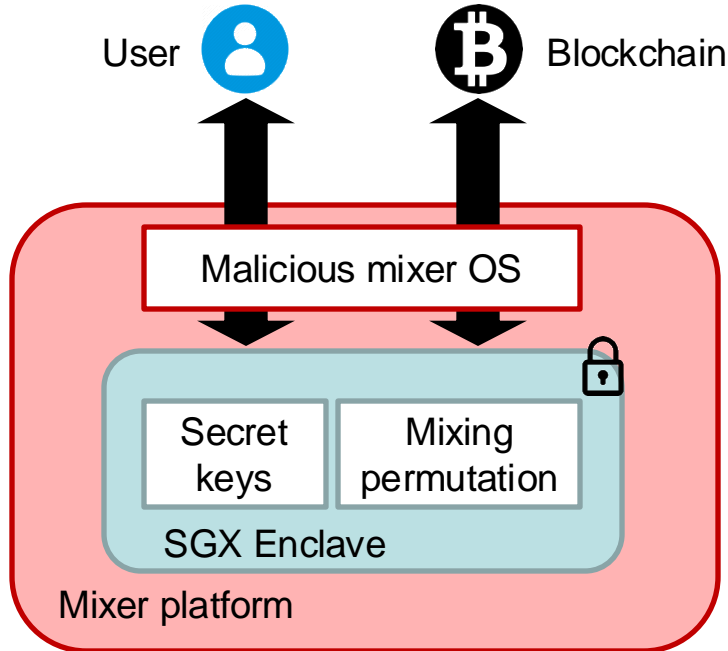
# Obscuro: centralized mixer protected by *Trusted Execution Environments*



**Naïve Obscuro Model**

- Trusted Execution Environments (TEEs)
  - ✓ **confidentiality** and **integrity** of code and data
- **Is this Naïve TEE-based mixer design *sufficient*?**
- **TEE-based mixer** addresses **simple** attacks
  - ✓ malicious mixer OS cannot **steal coins** or **leak** the mixing permutation
  - ✓ users verify Obscuro's execution **before** they join

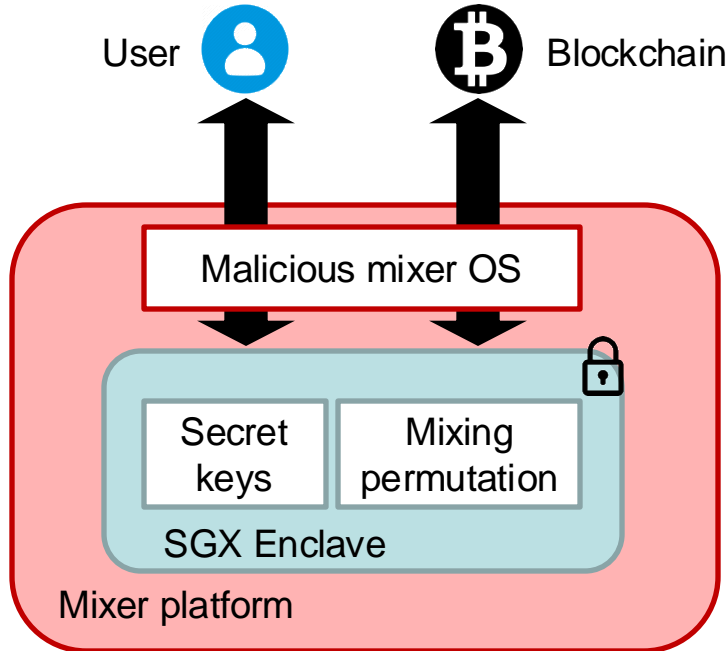
# *Sophisticated* attacks against *naïve* Obscuro design



**Naïve Obscuro Model**

# *Sophisticated* attacks against *naïve* Obscuro design

## Adversary model

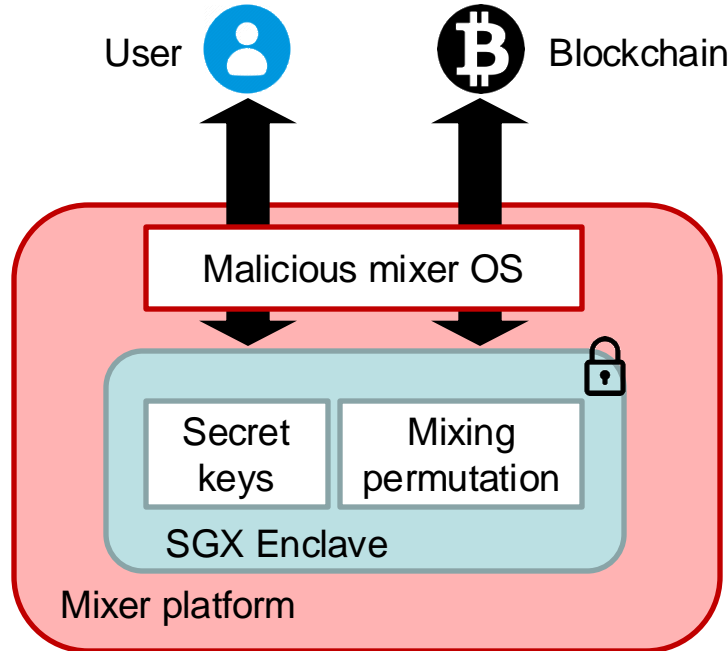


**Naïve Obscuro Model**

# *Sophisticated* attacks against *naïve* Obscuro design

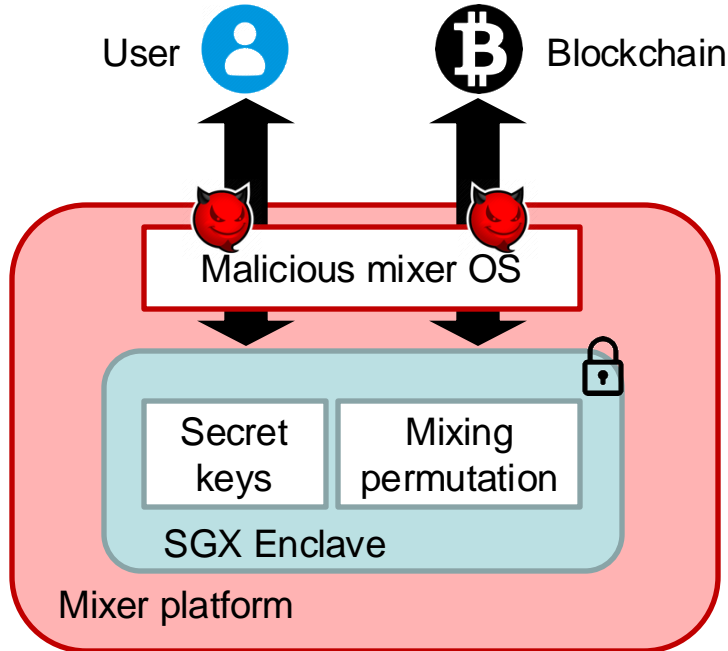
## Adversary model

- **Capabilities:**



**Naïve Obscuro Model**

# *Sophisticated* attacks against *naïve* Obscuro design

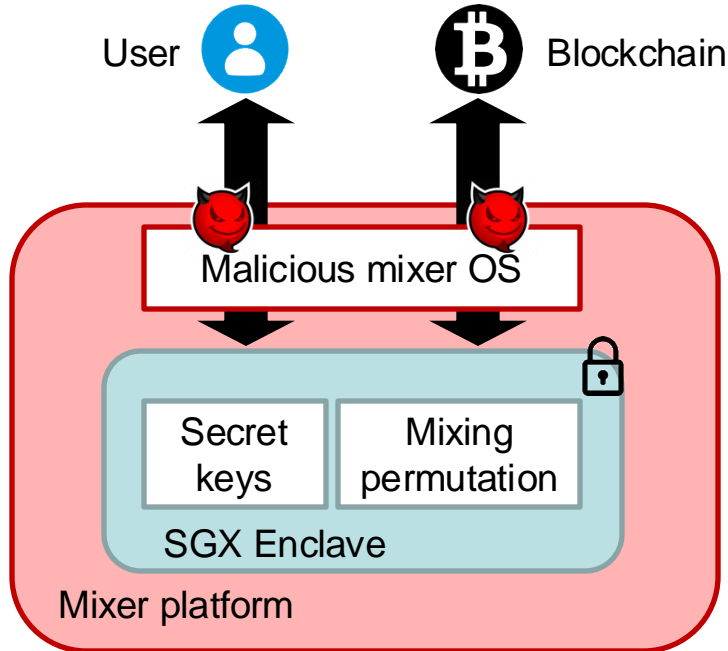


**Naïve Obscuro Model**

## Adversary model

- **Capabilities:**
  - ✓ Mixer OS can read/write all messages

# *Sophisticated* attacks against *naïve* Obscuro design



**Naïve Obscuro Model**

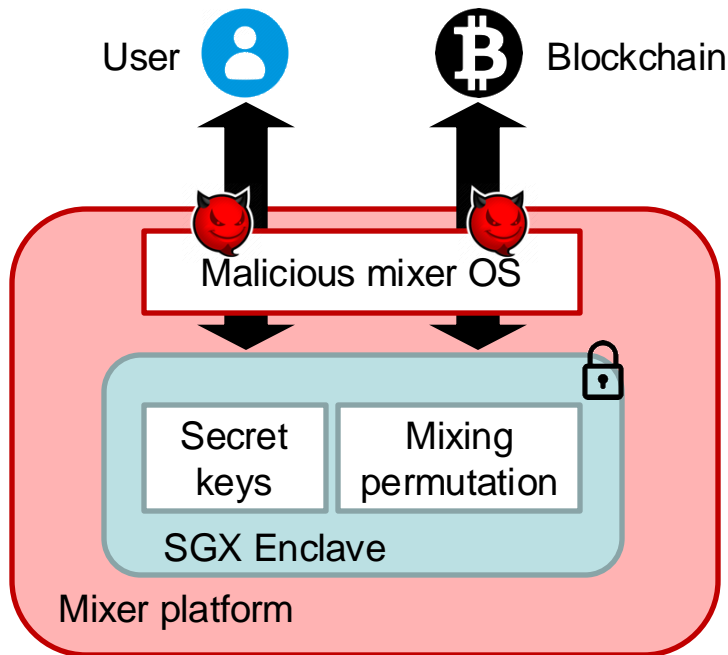
## Adversary model

- **Capabilities:**

- ✓ Mixer OS can read/write all messages
- ✓ Mixer OS can create Bitcoin blocks and transactions



# *Sophisticated* attacks against *naïve* Obscuro design

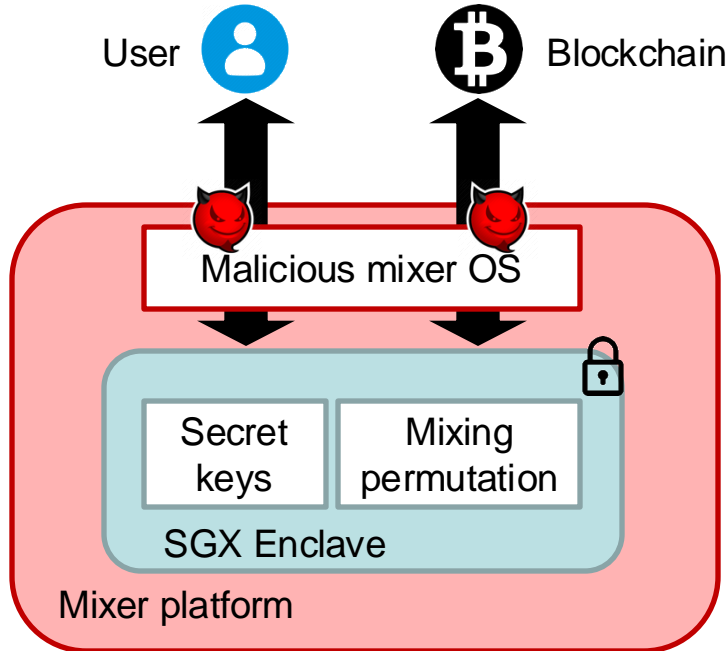


**Naïve Obscuro Model**

## Adversary model

- **Capabilities:**
  - ✓ Mixer OS can read/write all messages
  - ✓ Mixer OS can create Bitcoin blocks and transactions
- **Goal: *reduce anonymity set***

# *Sophisticated* attacks against *naïve* Obscuro design



**Naïve Obscuro Model**

## Adversary model

- **Capabilities:**

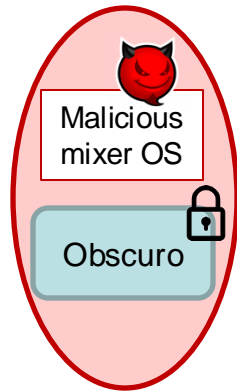
- ✓ Mixer OS can read/write all messages
- ✓ Mixer OS can create Bitcoin blocks and transactions

- **Goal: *reduce anonymity set***

Two specific attacks:

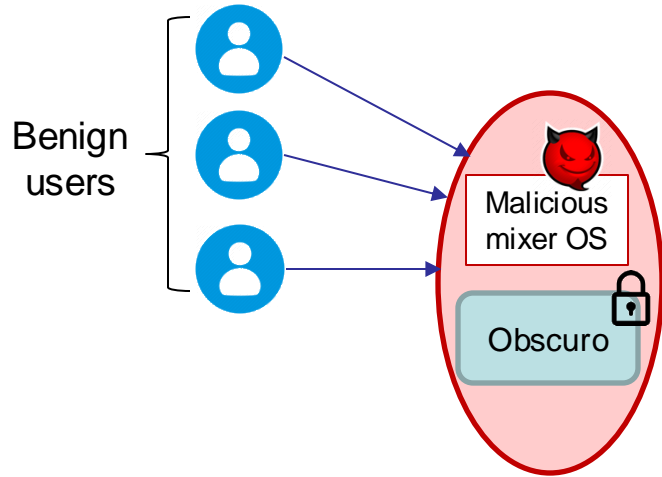
- (1) Participation rejection
- (2) Blockchain forking

# Attack 1: *Participation Rejection*



**Naïve Obscuro**

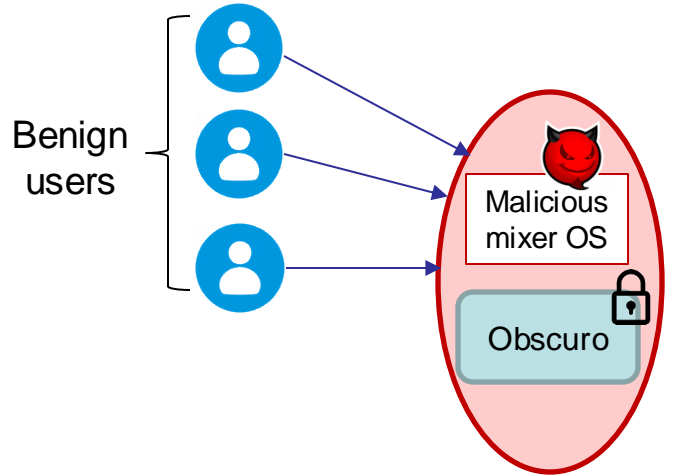
# Attack 1: *Participation Rejection*



**Naïve Obscuro**

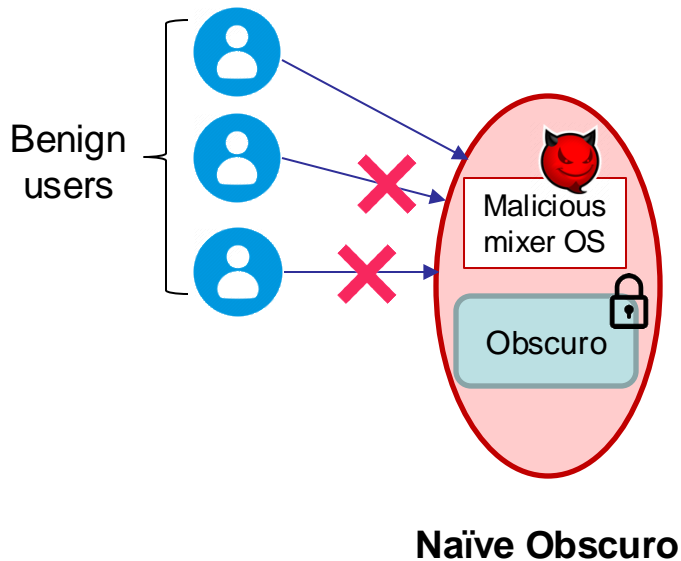
# Attack 1: *Participation Rejection*

- Malicious mixer OS *rejects* some benign users:



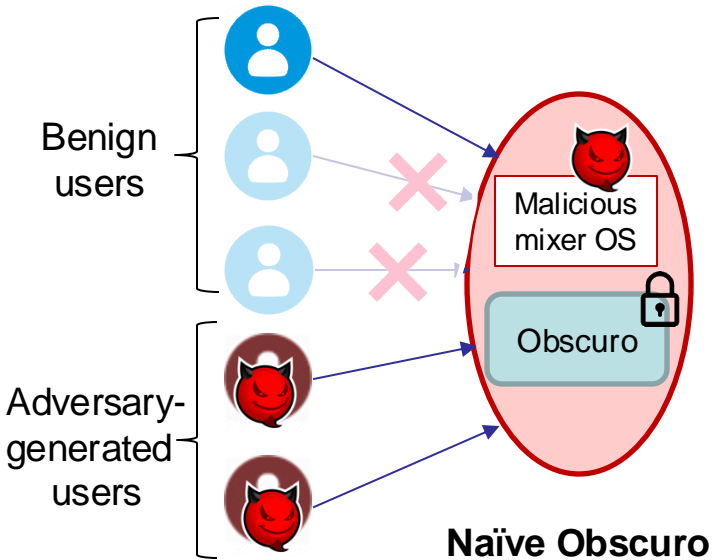
Naïve Obscuro

# Attack 1: *Participation Rejection*



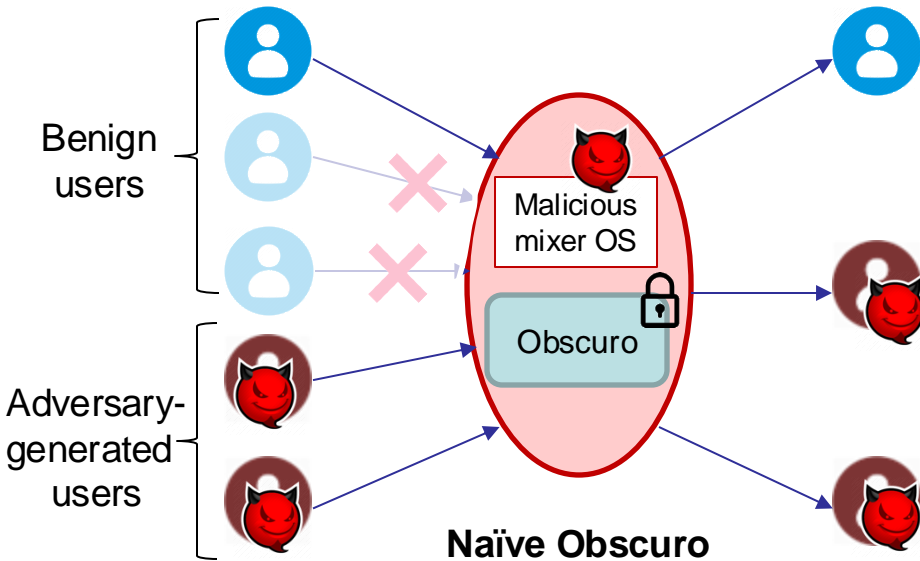
- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients

# Attack 1: *Participation Rejection*



- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients
- Instead, adversary inserts her own participants

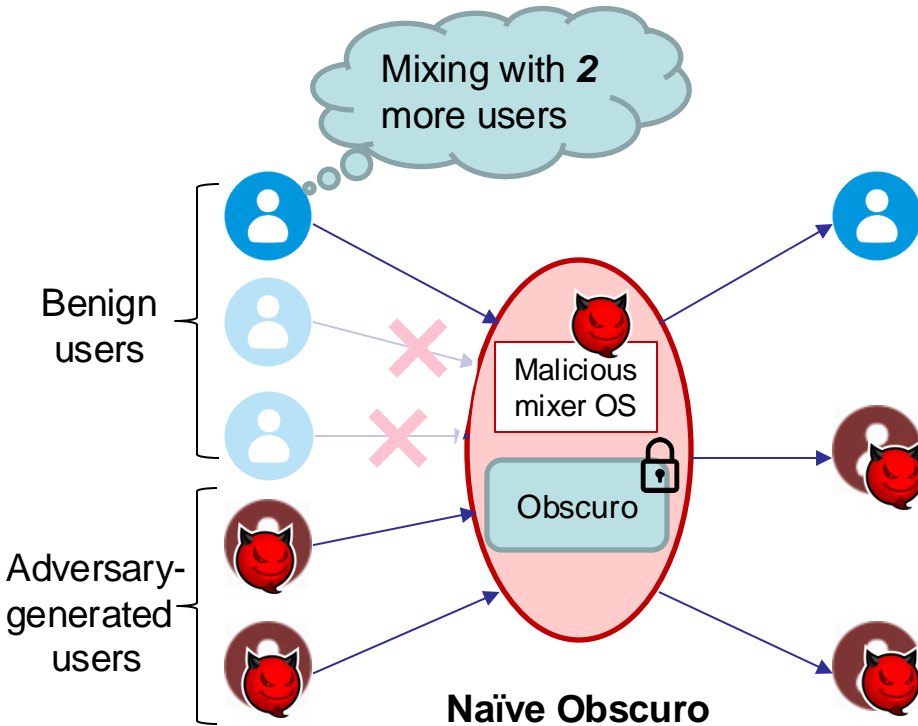
# Attack 1: *Participation Rejection*



- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients
- Instead, adversary inserts her own participants

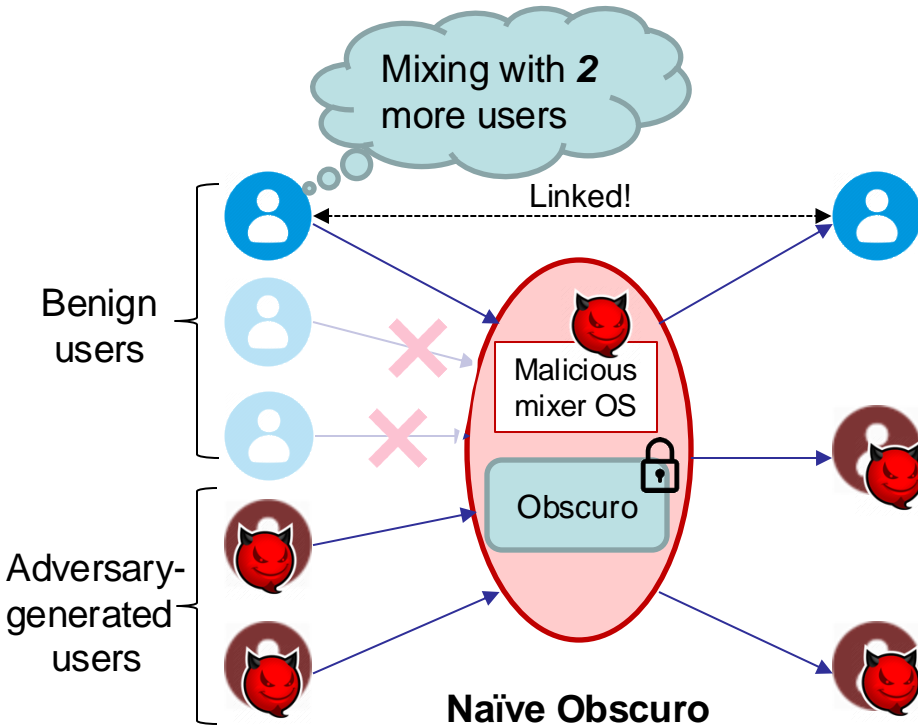


# Attack 1: *Participation Rejection*



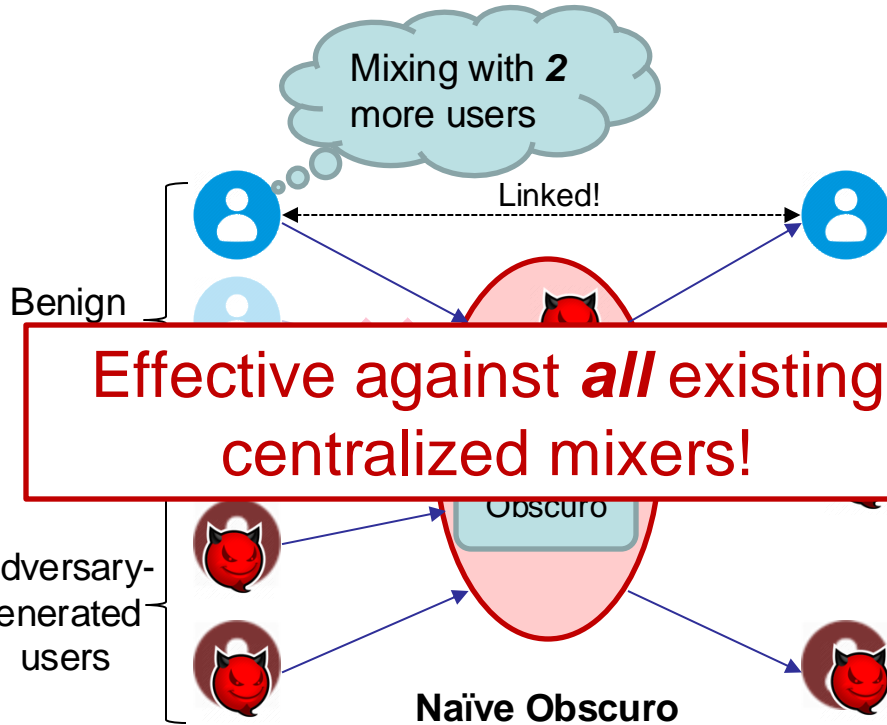
- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients
- Instead, adversary inserts her own participants
- Benign users may see a large anonymity set but small anonymity set in practice

# Attack 1: *Participation Rejection*



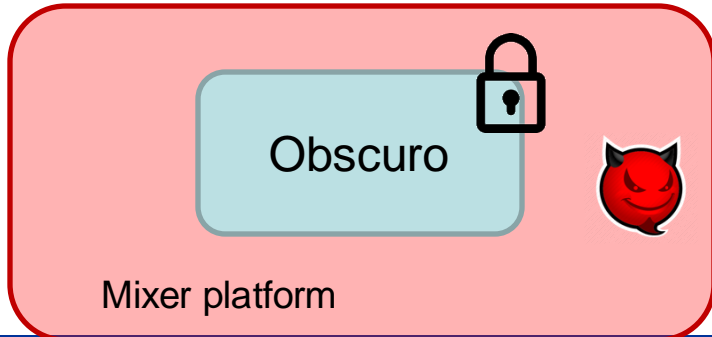
- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients
- Instead, adversary inserts her own participants
- Benign users may see a large anonymity set but small anonymity set in practice

# Attack 1: *Participation Rejection*



- Malicious mixer OS *rejects* some benign users:
  - ✓ Users cannot inform Obscuro their recipients
- Instead, adversary inserts her own participants
- Benign users may see a large anonymity set but small anonymity set in practice

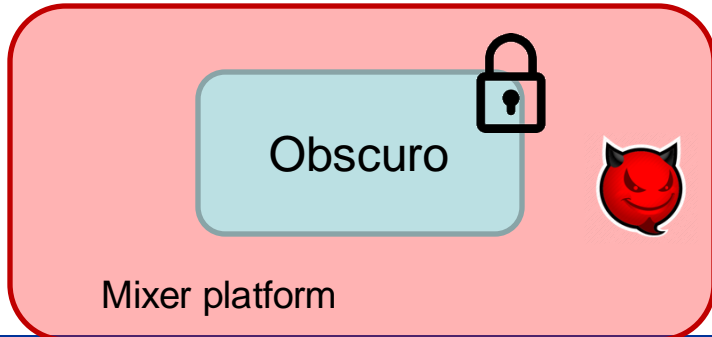
# *Indirect Participation* against Attack 1



# Indirect Participation against Attack 1



- Users *never* contact the mixer

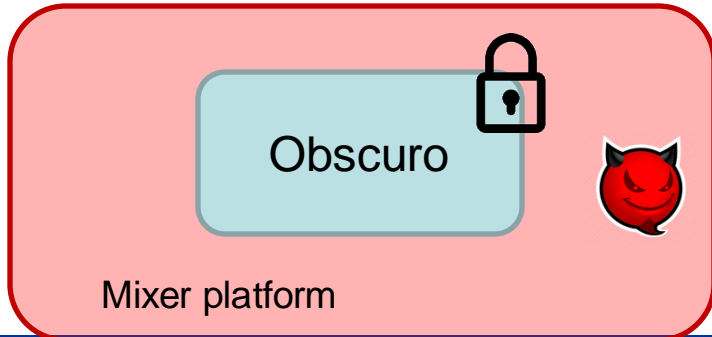


# Indirect Participation against Attack 1



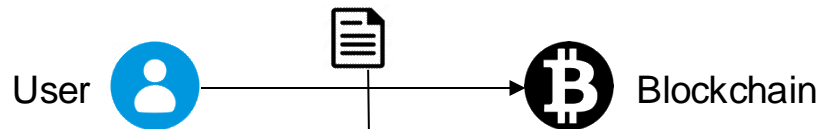
- Users *never* contact the mixer
- But submit transactions to the blockchain to participate mixing:

```
PayTo = mixer_address  
Amount = 1 BTC  
Data = Enc(recipient_address, )  
RefundTo = refund_address
```




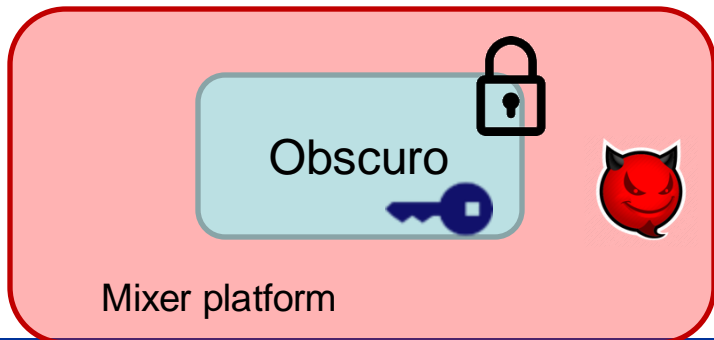
A red rounded rectangle containing a light blue rounded rectangle labeled "Obscuro" with a padlock icon above it and a red devil icon to its right. Below the rectangle is the text "Mixer platform".

# Indirect Participation against Attack 1



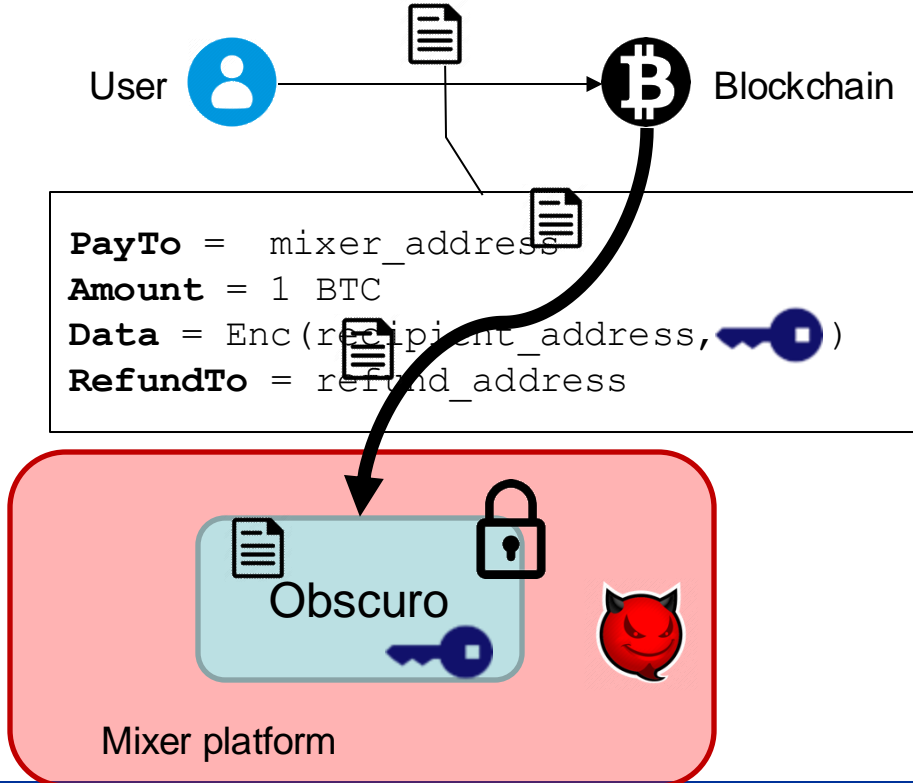
- Users **never** contact the mixer
- But submit transactions to the blockchain to participate mixing:
- Recipient address is **encrypted** via ECIES\* (a **69-byte** encryption)

```
PayTo = mixer_address  
Amount = 1 BTC  
Data = Enc(recipient_address, )  
RefundTo = refund_address
```



\*: Elliptic Curve Integrated Encryption Scheme

# Indirect Participation against Attack 1

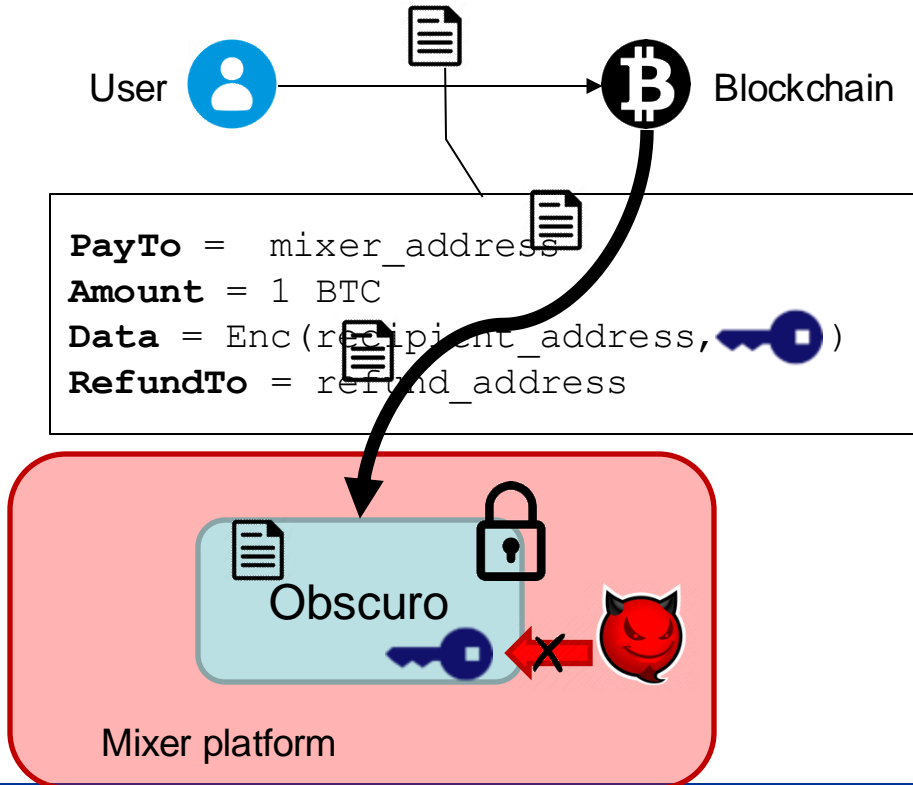


- Users **never** contact the mixer
- But submit transactions to the blockchain to participate mixing:
  - Recipient address is **encrypted** via ECIES\* (a **69-byte** encryption)
- Obscuro searches user deposits from the blockchain

\*: Elliptic Curve Integrated Encryption Scheme



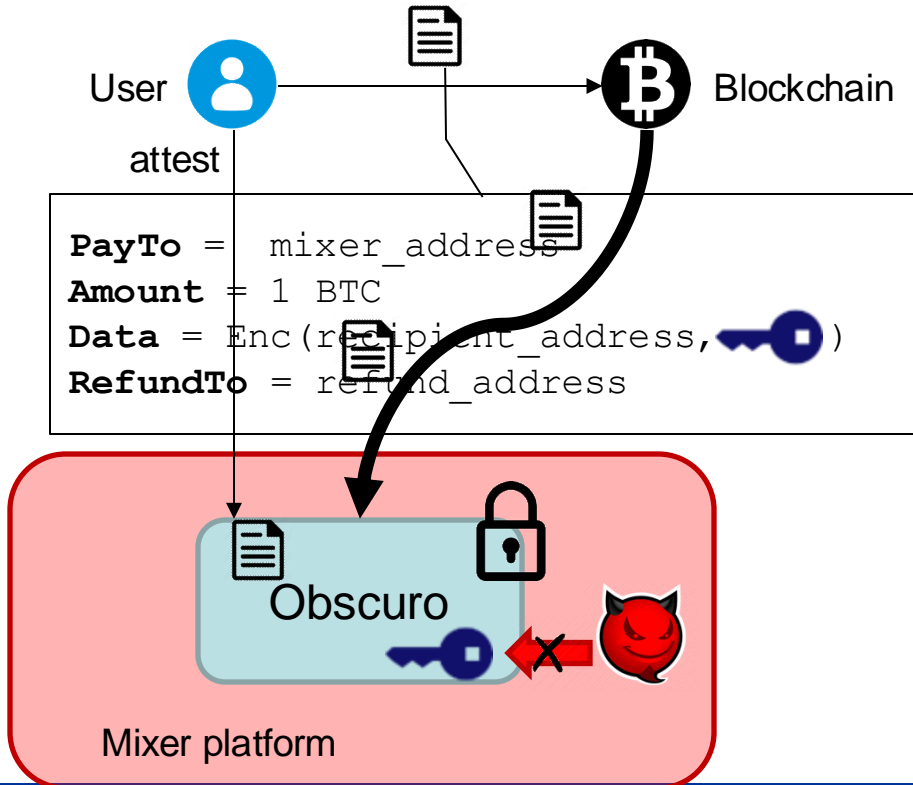
# Indirect Participation against Attack 1



- Users **never** contact the mixer
- But submit transactions to the blockchain to participate mixing:
  - Recipient address is **encrypted** via ECIES\* (a **69-byte** encryption)
- Obscuro searches user deposits from the blockchain

\*: Elliptic Curve Integrated Encryption Scheme

# Indirect Participation against Attack 1



- Users **never** contact the mixer
- But submit transactions to the blockchain to participate mixing:
  - Recipient address is **encrypted** via ECIES\* (a **69-byte** encryption)
- Obscuro searches user deposits from the blockchain

\*: Elliptic Curve Integrated Encryption Scheme

# Attack 2: *Blockchain Forking*



# Attack 2: *Blockchain Forking*

*Reducing anonymity set* even when Obscuro deploys *indirect participation*



# Attack 2: *Blockchain Forking*

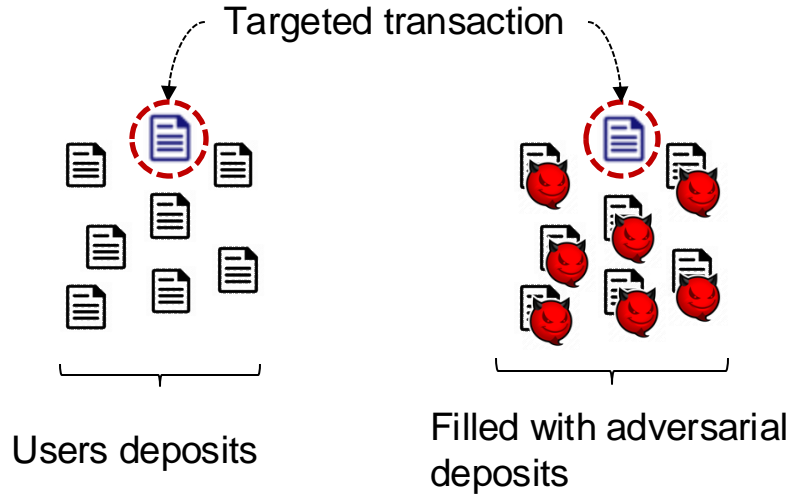
*Reducing anonymity set* even when Obscuro deploys *indirect participation*



Users deposits

# Attack 2: *Blockchain Forking*

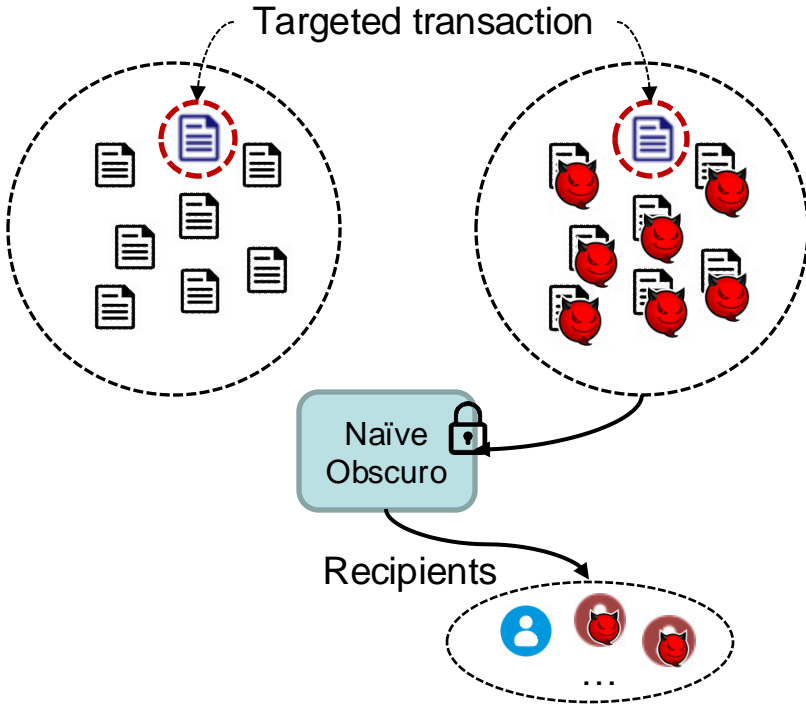
*Reducing anonymity set* even when Obscuro deploys *indirect participation*



- Mixer OS creates *another mixing set*
  - ✓ including some targeted transactions and her transactions

# Attack 2: *Blockchain Forking*

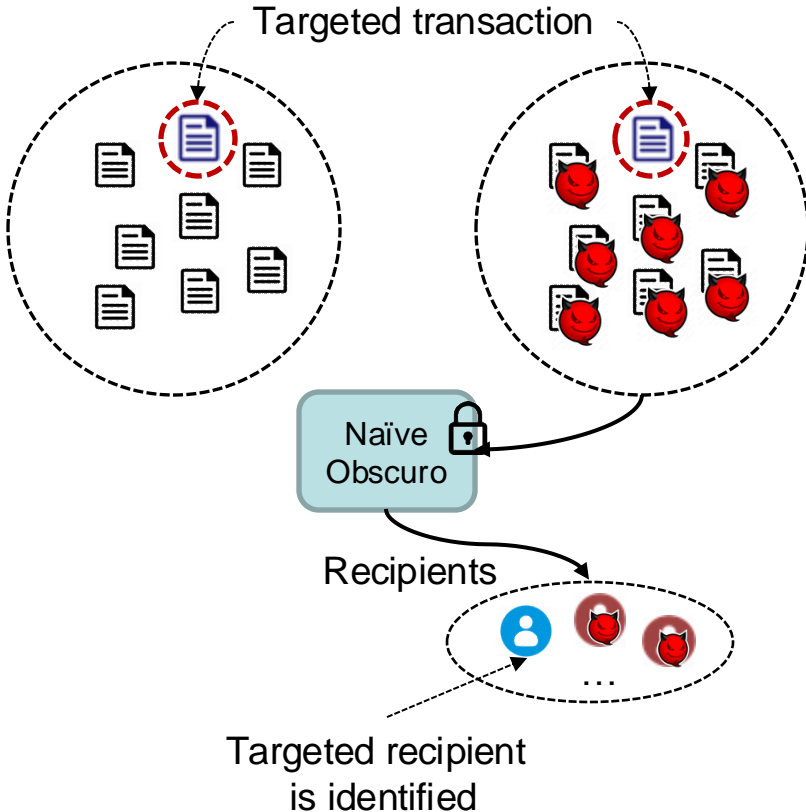
*Reducing anonymity set* even when Obscuro deploys *indirect participation*



- Mixer OS creates *another mixing set*
  - ✓ including some targeted transactions and her transactions
- Adversarial set is fed to Obscuro

# Attack 2: *Blockchain Forking*

*Reducing anonymity set* even when Obscuro deploys *indirect participation*

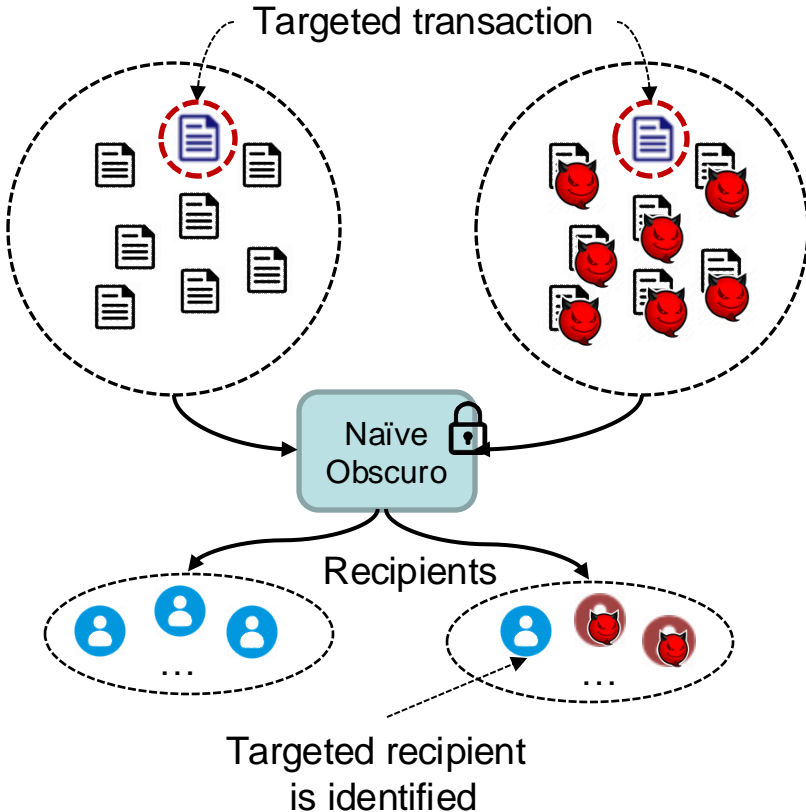


- Mixer OS creates *another mixing set*
  - ✓ including some targeted transactions and her transactions
- Adversarial set is fed to Obscuro
  - ✓ the targeted recipient is *identified*



# Attack 2: *Blockchain Forking*

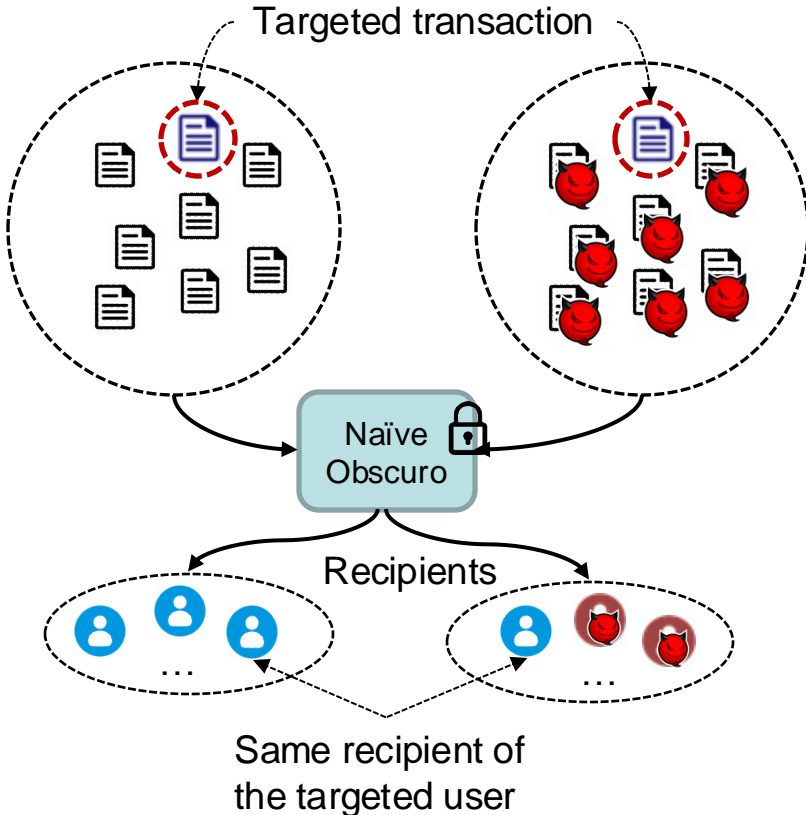
*Reducing anonymity set* even when Obscuro deploys *indirect participation*



- Mixer OS creates *another mixing set*
  - ✓ including some targeted transactions and her transactions
- Adversarial set is fed to Obscuro
  - ✓ the targeted recipient is *identified*
- Valid users set is fed to Obscuro

# Attack 2: *Blockchain Forking*

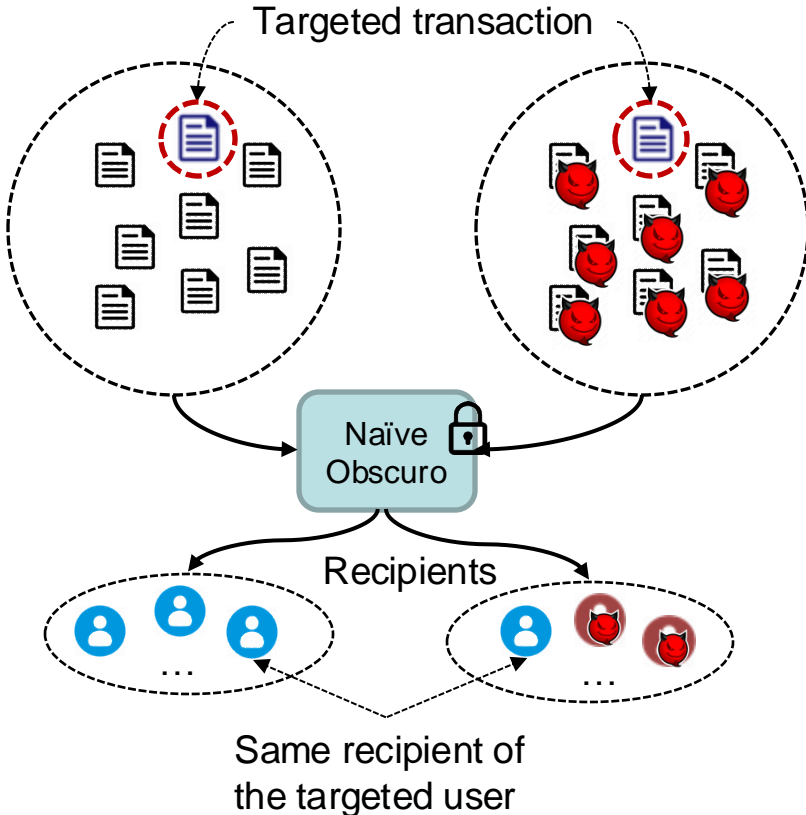
Reducing anonymity set even when Obscuro deploys *indirect participation*



- Mixer OS creates *another mixing set*
  - ✓ including some targeted transactions and her transactions
- Adversarial set is fed to Obscuro
  - ✓ the targeted recipient is *identified*
- Valid users set is fed to Obscuro
  - ✓ the targeted recipient is *already* identified

# Attack 2: *Blockchain Forking*

Reducing anonymity set even when Obscuro deploys *indirect participation*



- Mixer OS creates **another mixing set**
  - ✓ including some targeted transactions and her transactions
- **How to enforce naïve Obscuro to mix a transaction twice?**
- Valid users set is fed to Obscuro
  - ✓ the targeted recipient is **already** identified

# How to *fork* blockchain

Malicious mixer OS

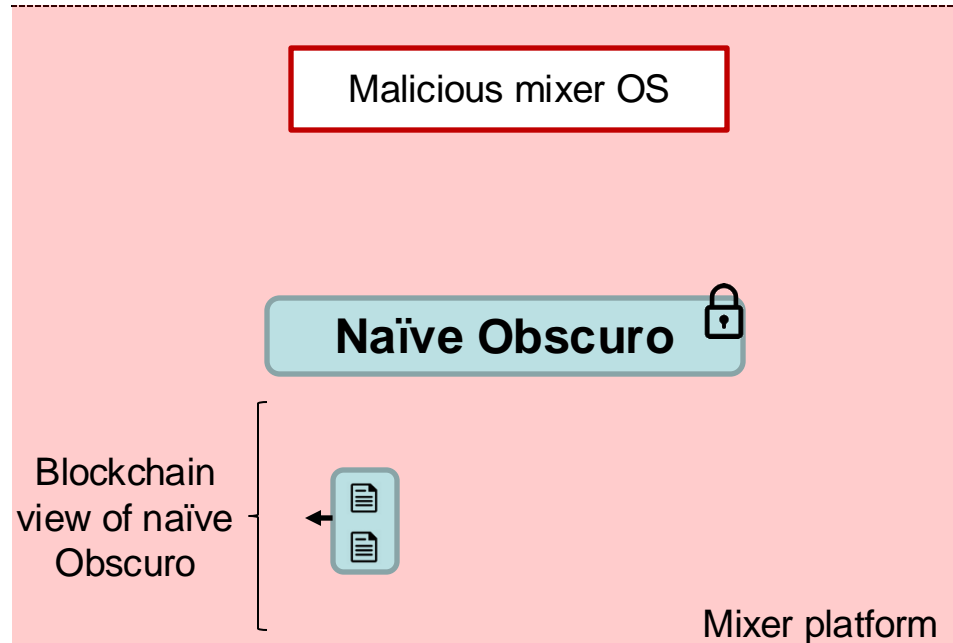
Naïve Obscuro 

Mixer platform

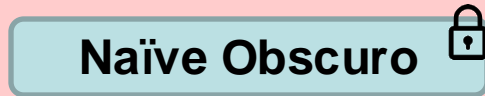
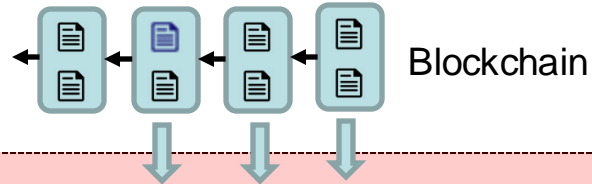
# How to *fork* blockchain



- Malicious OS *manipulates* the *blockchain view* of the naïve Obscuro:



# How to *fork* blockchain

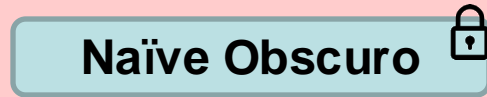
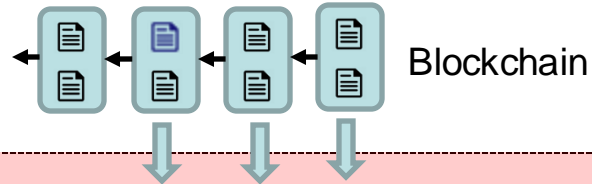


Blockchain view of naïve Obscuro

Mixer platform

- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:

# How to *fork* blockchain



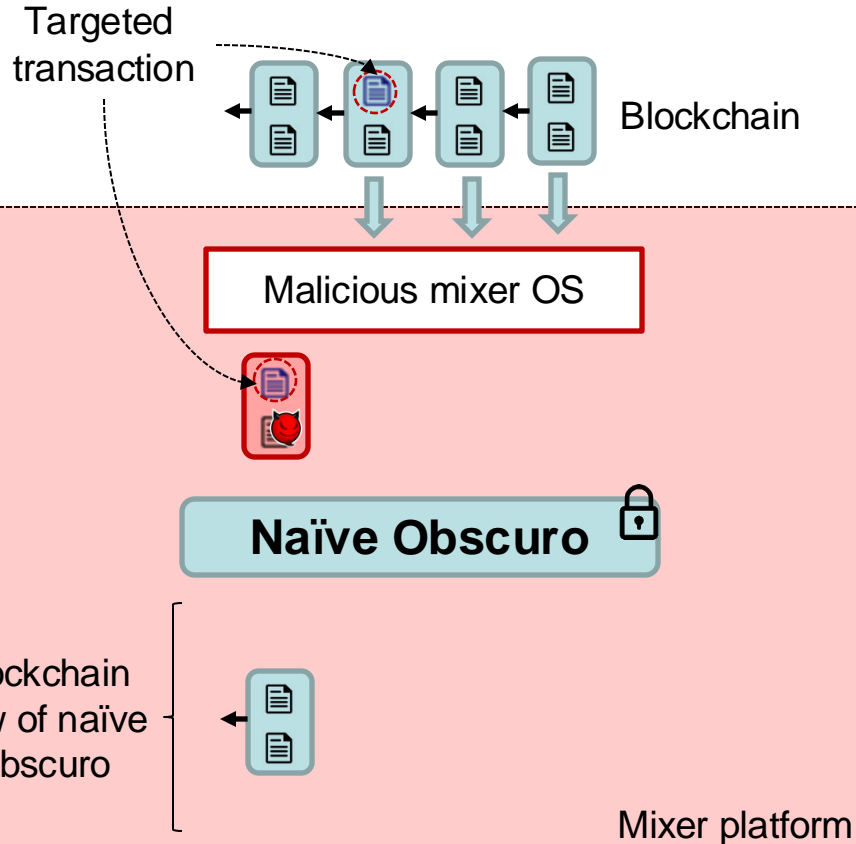
Blockchain  
view of naïve  
Obscuro



Mixer platform

- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions

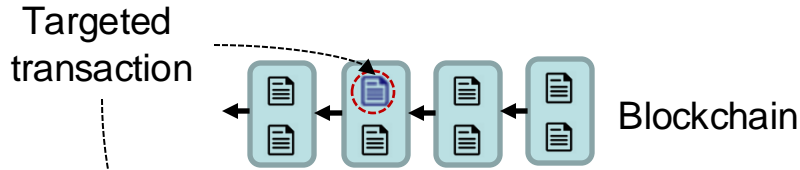
# How to *fork* blockchain



- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions




# How to *fork* blockchain



Malicious mixer OS

Adversarial transaction



Naïve Obscuro 

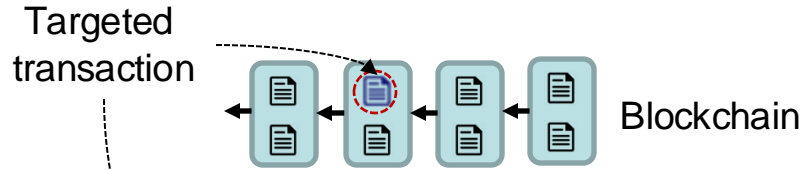
Blockchain view of naïve Obscuro



Mixer platform

- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions

# How to *fork* blockchain

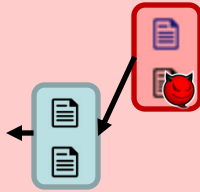


Adversarial transaction



Naïve Obscuro

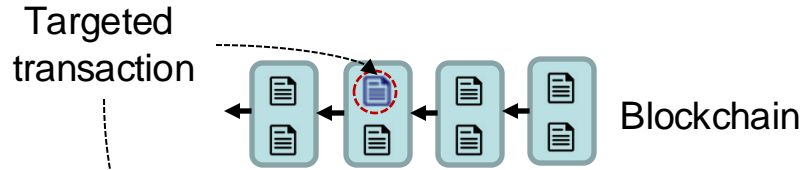
Blockchain view of naïve Obscuro



Mixer platform

- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions

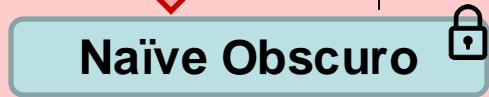
# How to *fork* blockchain



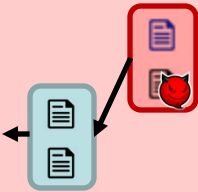
Adversarial transaction



Mixed transaction



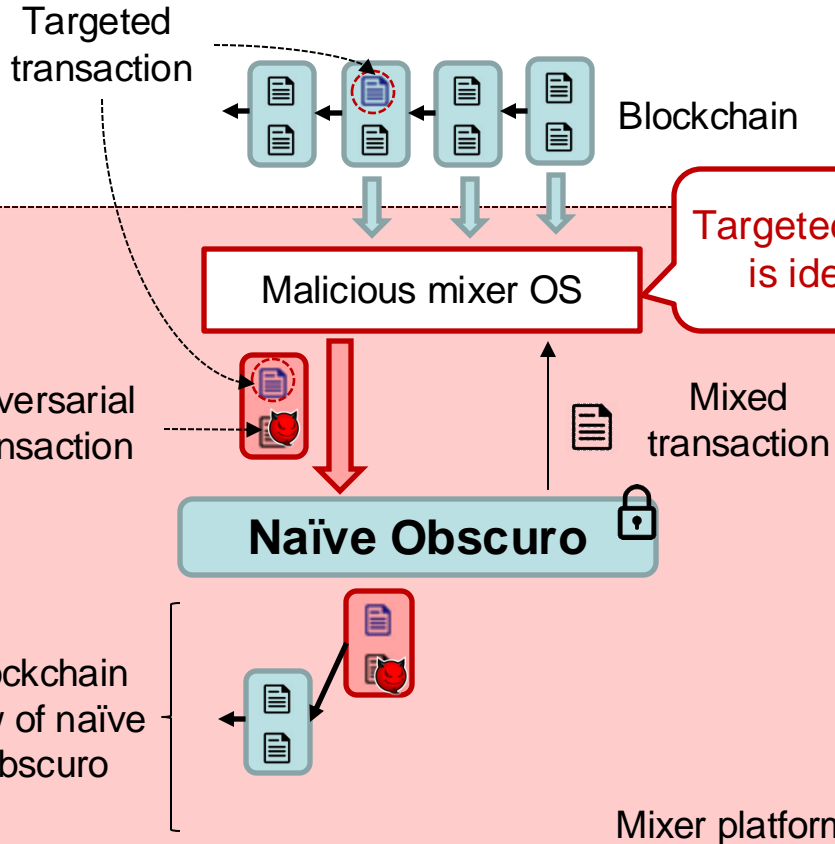
Blockchain view of naïve Obscuro



Mixer platform

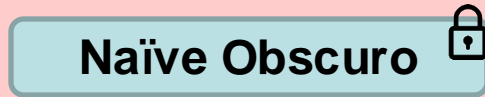
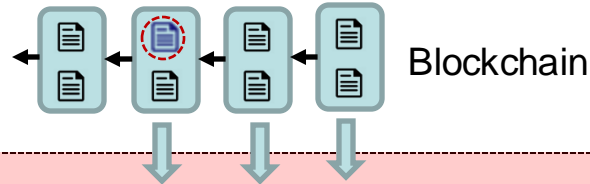
- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions

# How to *fork* blockchain

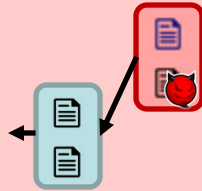


- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - adversary first feeds a block
  - includes targeted transactions and her transactions

# How to *fork* blockchain



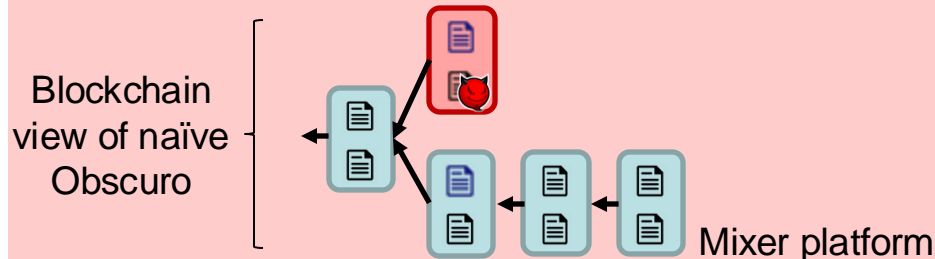
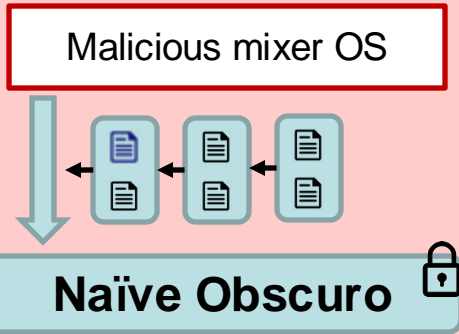
Blockchain view of naïve Obscuro



Mixer platform

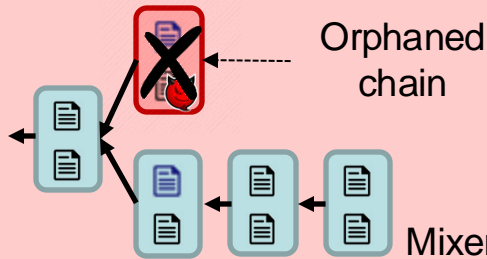
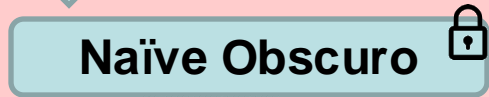
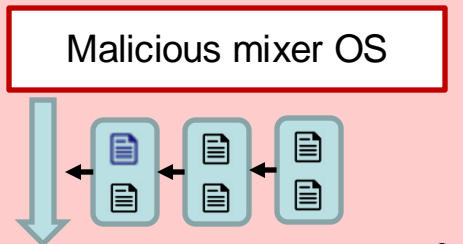
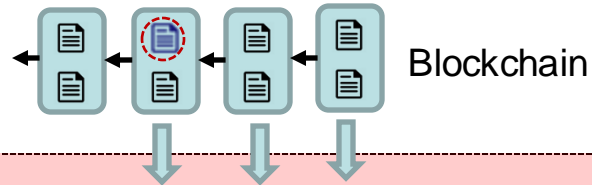
- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions

# How to *fork* blockchain



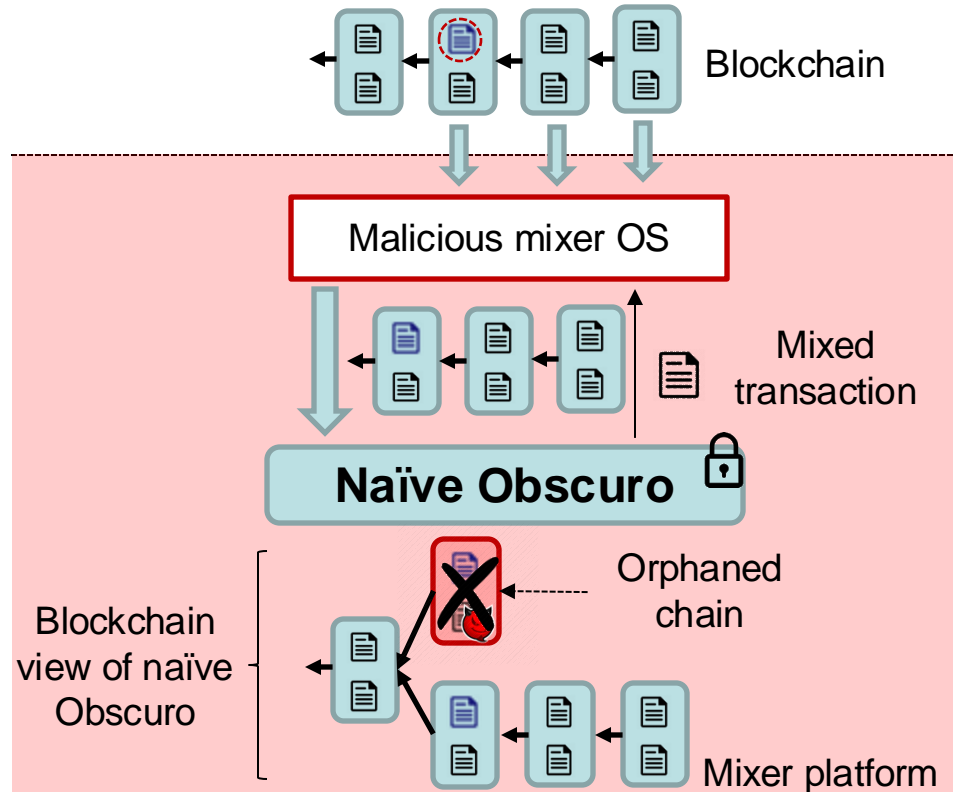
- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions
  - ✓ adversary then feeds the valid blockchain

# How to *fork* blockchain



- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions
  - ✓ adversary then feeds the valid blockchain
  - ✓ adversarial chain becomes **orphaned** and is **abandoned**

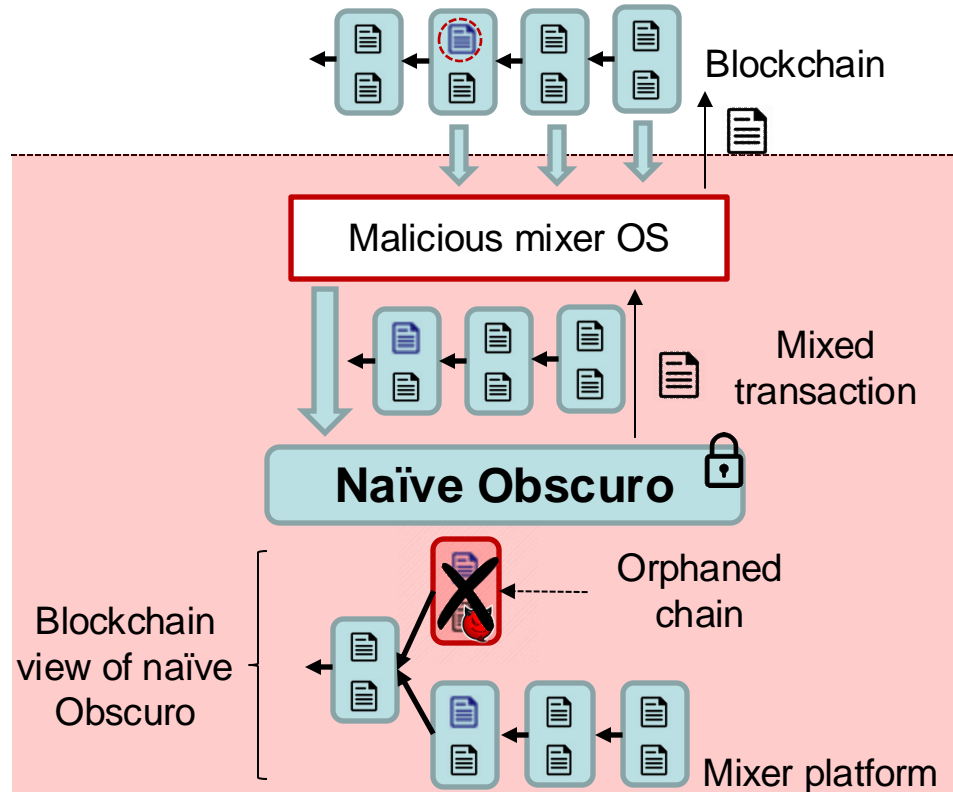
# How to *fork* blockchain



- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions
  - ✓ adversary then feeds the valid blockchain
  - ✓ adversarial chain becomes **orphaned** and is **abandoned**

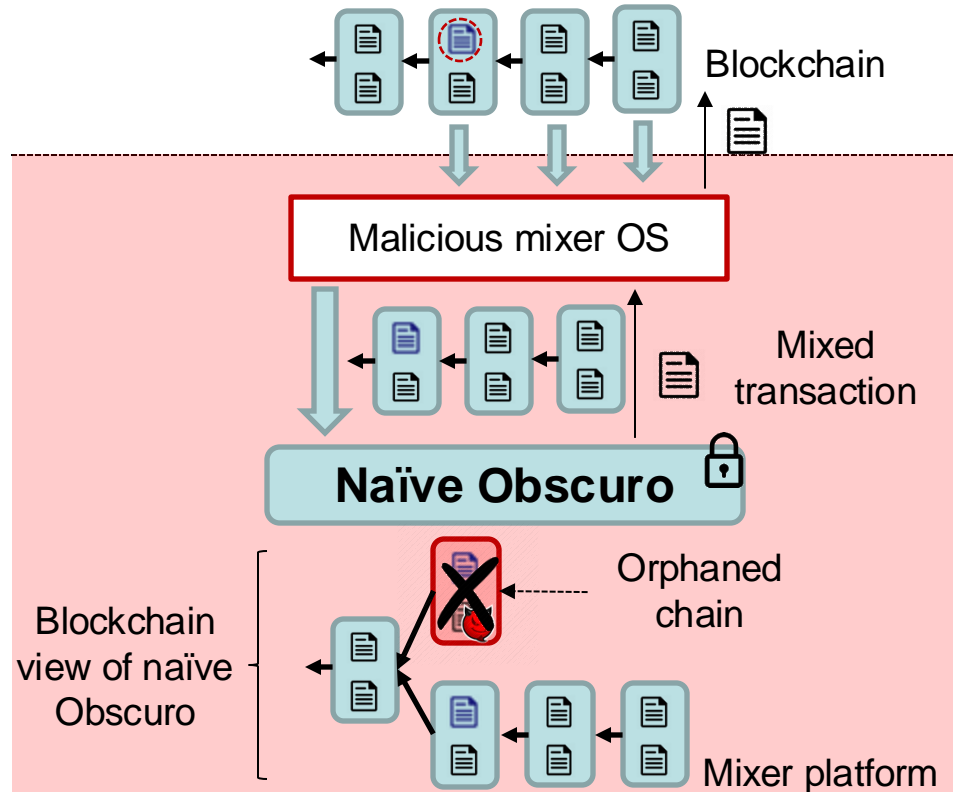


# How to *fork* blockchain



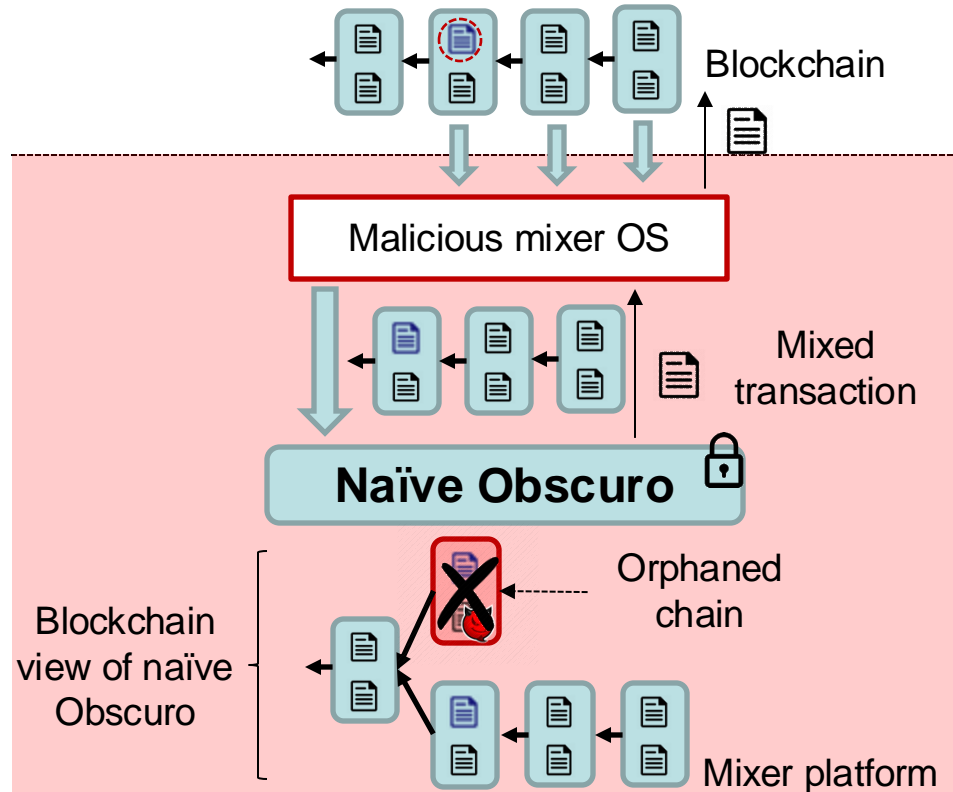
- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions
  - ✓ adversary then feeds the valid blockchain
  - ✓ adversarial chain becomes **orphaned** and is **abandoned**

# How to *fork* blockchain



- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
  - ✓ adversary first feeds a block includes targeted transactions and her transactions
  - ✓ adversary then feeds the valid blockchain
  - ✓ adversarial chain becomes **orphaned** and is **abandoned**
- Attack is **invisible**

# How to *fork* blockchain



- Malicious OS *manipulates* the **blockchain view** of the naïve Obscuro:
    - ✓ adversary first feeds a block includes targeted transactions and
- Effective against **all** existing centralized mixers!
- ✓ adversarial chain becomes **orphaned** and is **abandoned**
  - Attack is **invisible**

# *Mitigations* against Attack 2



## *Mitigations* against Attack 2

- Obscuro ***detects*** if blockchain feed contains two chains and a transaction is already mixed

## *Mitigations* against Attack 2

- Obscuro ***detects*** if blockchain feed contains two chains and a transaction is already mixed
- Obscuro ***excludes*** the repeated transaction and guarantees each transaction can be mixed at most once

## *Mitigations* against Attack 2

- Obscuro **detects** if blockchain feed contains two chains and a transaction is already mixed
- Obscuro **excludes** the repeated transaction and guarantees each transaction can be mixed at most once
  - ✓ Excluded transactions are refunded

# (Informal) Security Analysis



# (Informal) Security Analysis

Anonymity set size reduction attacks (Attack 1 & 2)

# (Informal) Security Analysis

## Anonymity set size reduction attacks (Attack 1 & 2)

- Require attack **capabilities**:

# (Informal) Security Analysis

## Anonymity set size reduction attacks (Attack 1 & 2)

- Require attack **capabilities**:
  - ✓ Rejecting user participations
  - ✓ Mixing twice with different anonymity set size

# (Informal) Security Analysis

## Anonymity set size reduction attacks (Attack 1 & 2)

- Require attack **capabilities:**

~~✓ Rejecting user participations~~ *removed*

*No direct deposits (via indirect participation)*

- ✓ Mixing twice with different anonymity set size

# (Informal) Security Analysis

## Anonymity set size reduction attacks (Attack 1 & 2)

- Require attack **capabilities:**

~~✓ Rejecting user participations~~ *removed*

*No direct deposits (via indirect participation)*

~~✓ Mixing twice with different anonymity set size~~ *removed*

*No malicious forking*

# (Informal) Security Analysis

## Anonymity set size reduction attacks (Attack 1 & 2)

- Require attack **capabilities:**

~~✓ Rejecting user participations~~ *removed*

*No direct deposits (via indirect participation)*

~~✓ Mixing twice with different anonymity set size~~ *removed*

*No malicious forking*

*More (informal) security analysis can be found in our paper*

# Implementation

# Implementation

- Obscuro's existing Trusted Computing Base (TCB)



# Implementation

- Obscuro's existing Trusted Computing Base (TCB)
  - ✓ Bitcoin core v0.13.1

# Implementation

- Obscuro's existing Trusted Computing Base (TCB)
  - ✓ Bitcoin core v0.13.1
  - ✓ OpenSSL, libsecp256k1

# Implementation

- Obscuro's existing Trusted Computing Base (TCB)
  - ✓ Bitcoin core v0.13.1
  - ✓ OpenSSL, libsecp256k1
  - ✓ Panoply [NDSS' 17] abstraction framework

# Implementation

- Obscuro's existing Trusted Computing Base (TCB)
  - ✓ Bitcoin core v0.13.1
  - ✓ OpenSSL, libsecp256k1
  - ✓ Panoply [NDSS' 17] abstraction framework
- Obscuro contributes only 2,442 SLoC in addition to TCB

# Implementation

- Obscuro's existing Trusted Computing Base (TCB)
  - ✓ Bitcoin core v0.13.1
  - ✓ OpenSSL, libsecp256k1
  - ✓ Panoply [NDSS' 17] abstraction framework
- Obscuro contributes only 2,442 SLoC in addition to TCB

<b>Obscuro's functions</b>	<b>1,150 SLoC</b>	} Additional TCB
<b>Bitcoin core modification</b>	<b>1,292 SLoC</b>	

# Performance Evaluation



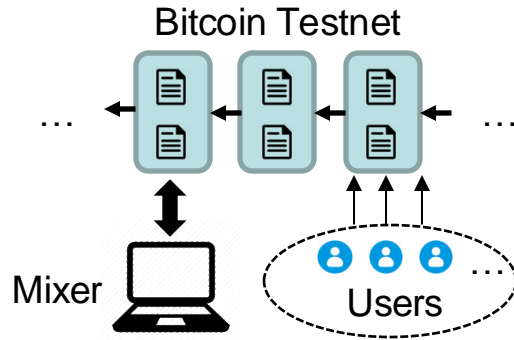
# Performance Evaluation

## Experiment setup



# Performance Evaluation

## Experiment setup



CPU: Intel Core i7-6820HQ

RAM: 8GB

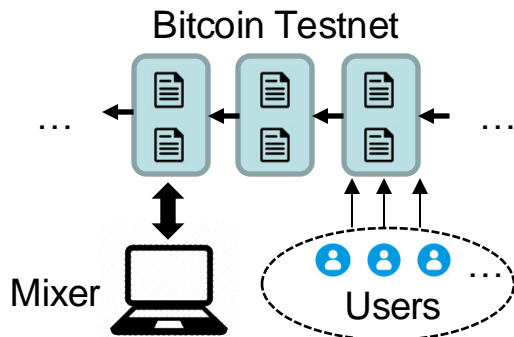
Ubuntu 16.04 LTS

Intel SGX SDK for Linux



# Performance Evaluation

## Experiment setup



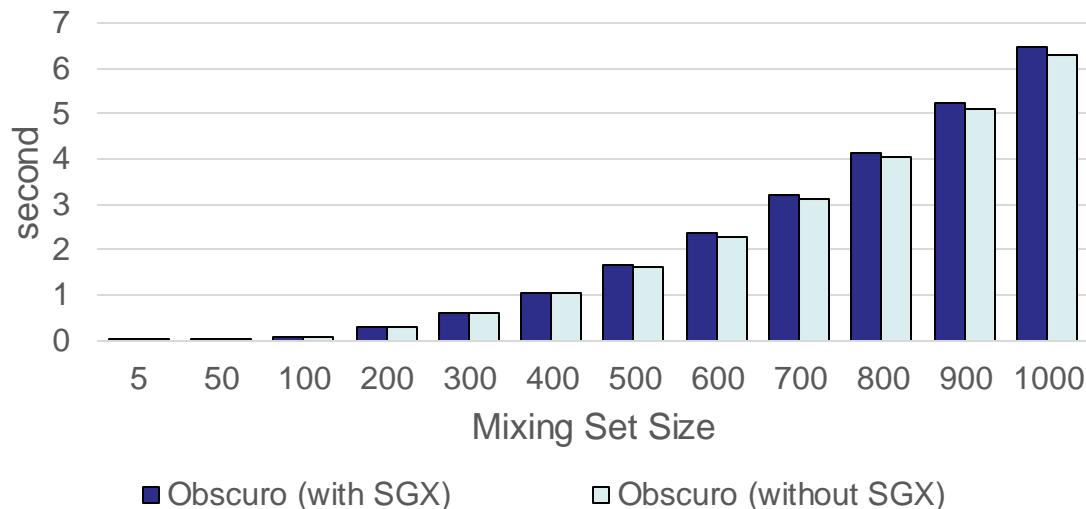
CPU: Intel Core i7-6820HQ

RAM: 8GB

Ubuntu 16.04 LTS

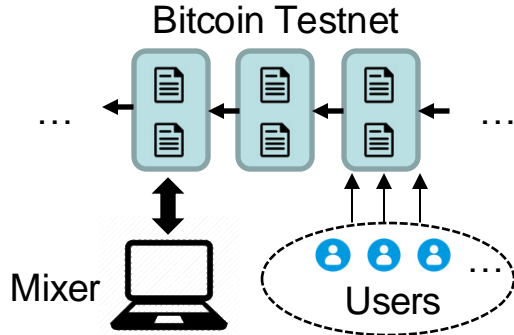
Intel SGX SDK for Linux

Computation time (mixing, signing) of Obscuro



# Performance Evaluation

## Experiment setup



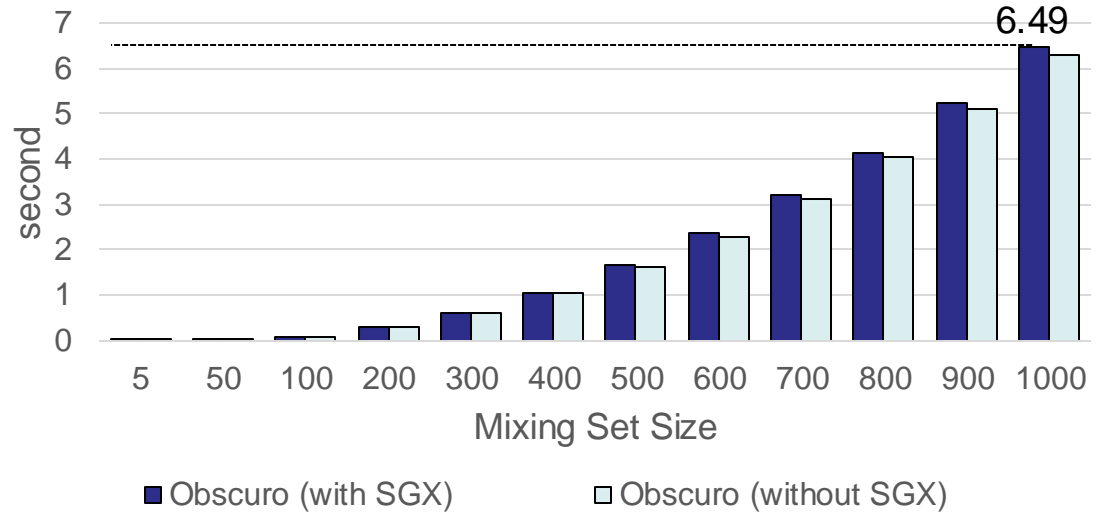
CPU: Intel Core i7-6820HQ

RAM: 8GB

Ubuntu 16.04 LTS

Intel SGX SDK for Linux

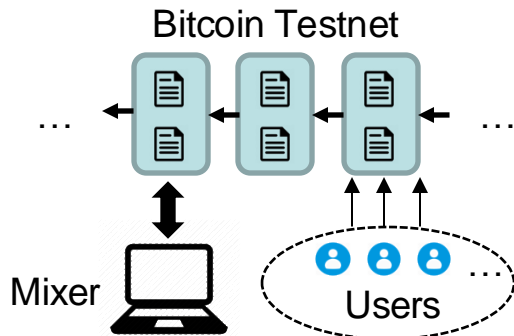
Computation time (mixing, signing) of Obscuro



**Obscuro mixed *1,000 users* on Bitcoin Testnet in *6s***

# Performance Evaluation

## Experiment setup



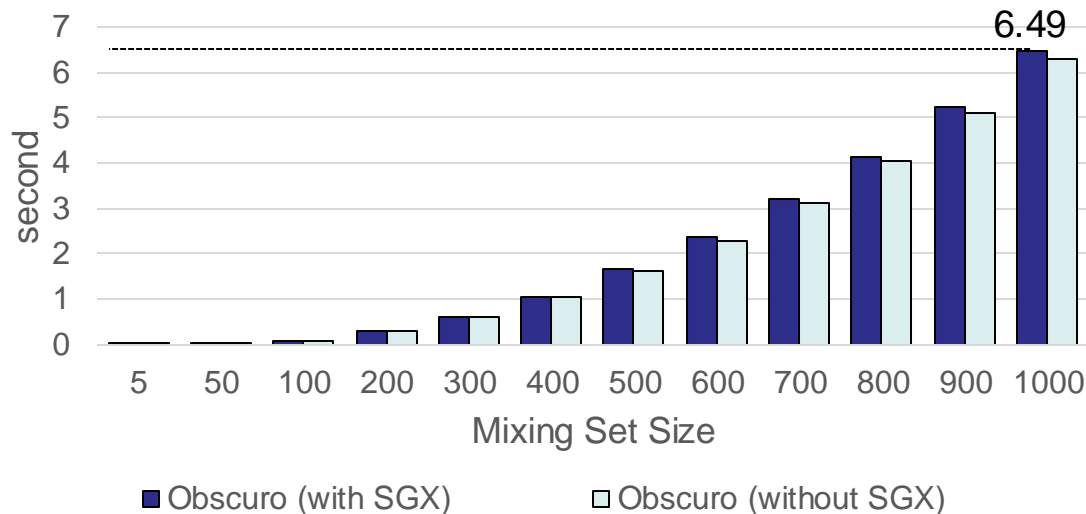
CPU: Intel Core i7-6820HQ

RAM: 8GB

Ubuntu 16.04 LTS

Intel SGX SDK for Linux

Computation time (mixing, signing) of Obscuro

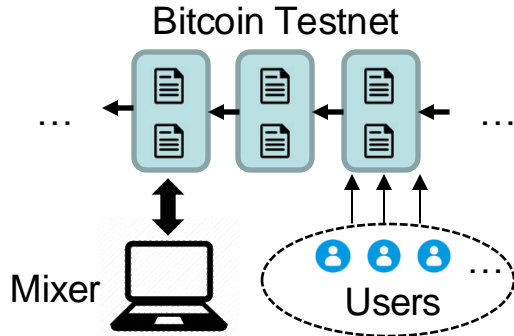


**Obscuro mixed *1,000 users* on Bitcoin Testnet in *6s***

E.g., TumbleBit [NDSS' 17] computation time for **one** pair of sender-recipient is **0.6s**

# Performance Evaluation

## Experiment setup



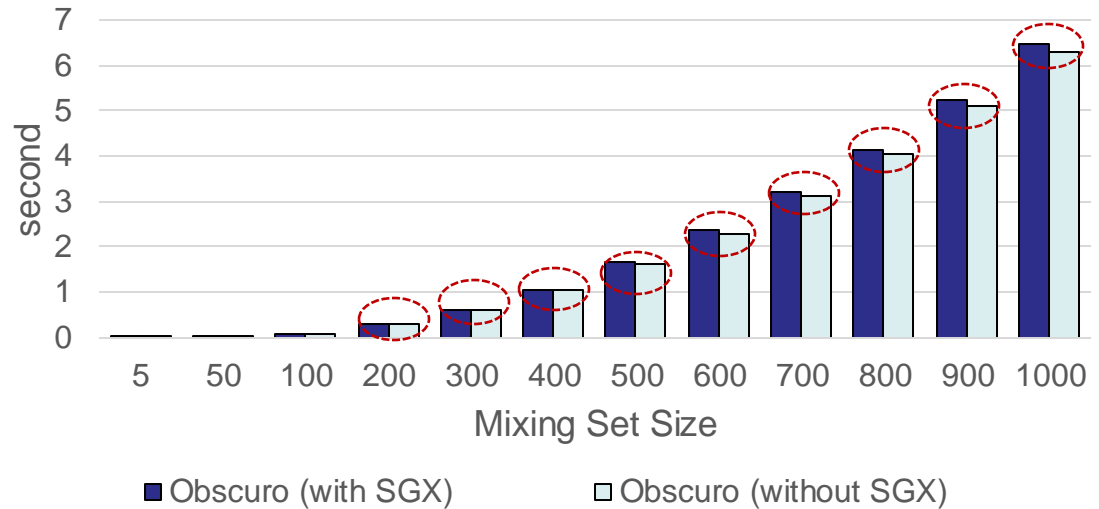
CPU: Intel Core i7-6820HQ

RAM: 8GB

Ubuntu 16.04 LTS

Intel SGX SDK for Linux

Computation time (mixing, signing) of Obscuro



Obscuro mixed **1,000 users** on Bitcoin Testnet in **6s**

SGX overhead is **negligible!**



# Comparison with Existing Mixers

Bitcoin Mixers	
Decentralized	Coinshuffle++ [NDSS'17]
	CoinParty [CODASPY'15]
	Xim [WPES'13]
	<b>Obscuro</b>

# Comparison with Existing Mixers

Bitcoin Mixers	
Decentralized	Coinshuffle++ [NDSS'17]
	CoinParty [CODASPY'15]
	Xim [WPES'13]
Centralized	Blindcoin [FC'15]
	TumbleBit [NDSS'17]
	<b>Obscuro</b>

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention
Decentralized	Coinshuffle++ [NDSS'17]	
	CoinParty [CODASPY'15]	
	Xim [WPES'13]	
Centralized	Blindcoin [FC'15]	
	TumbleBit [NDSS'17]	
	<b>Obscuro</b>	



# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee
Decentralized	Coinshuffle++ [NDSS'17]		
	CoinParty [CODASPY'15]		
	Xim [WPES'13]		
Centralized	Blindcoin [FC'15]		
	TumbleBit [NDSS'17]		
	<b>Obscuro</b>		

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓
	CoinParty [CODASPY'15]	✓	✓
	Xim [WPES'13]	✓	✓
Centralized	Blindcoin [FC'15]		
	TumbleBit [NDSS'17]	✓	
	<b>Obscuro</b>	✓	✓

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓	
	CoinParty [CODASPY'15]	✓	✓	
	Xim [WPES'13]	✓	✓	
Centralized	Blindcoin [FC'15]			
	TumbleBit [NDSS'17]	✓		
	<b>Obscuro</b>	✓	✓	

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓		
	CoinParty [CODASPY'15]	✓	✓		
	Xim [WPES'13]	✓	✓		
Centralized	Blindcoin [FC'15]				
	TumbleBit [NDSS'17]	✓			
	<b>Obscuro</b>	✓	✓		

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓		
	CoinParty [CODASPY'15]	✓	✓	✓	
	Xim [WPES'13]	✓	✓		✓
Centralized	Blindcoin [FC'15]				✓
	TumbleBit [NDSS'17]	✓			✓
	<b>Obscuro</b>	✓	✓	✓	✓

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓		
	CoinParty [CODASPY'15]	✓	✓	✓	
	Xim [WPES'13]	✓	✓		✓
Centralized	Blindcoin [FC'15]				✓
	TumbleBit [NDSS'17]	✓			✓
	<b>Obscuro</b>	✓	✓	✓	✓

## Obscuro's limitations:

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓		
	CoinParty [CODASPY'15]	✓	✓	✓	
	Xim [WPES'13]	✓	✓		✓
Centralized	Blindcoin [FC'15]				✓
	TumbleBit [NDSS'17]	✓			✓
	<b>Obscuro</b>	✓	✓	✓	✓

## Obscuro's limitations:

- Reliance on Intel SGX → deployed with other trusted-hardware (e.g., ARM TrustZone, OP-TEE)

# Comparison with Existing Mixers

	Bitcoin Mixers	Theft prevention	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance
Decentralized	Coinshuffle++ [NDSS'17]	✓	✓		
	CoinParty [CODASPY'15]	✓	✓	✓	
	Xim [WPES'13]	✓	✓		✓
Centralized	Blindcoin [FC'15]				✓
	TumbleBit [NDSS'17]	✓			✓
	<b>Obscuro</b>	✓	✓	✓	✓

## Obscuro's limitations:

- Reliance on Intel SGX → deployed with other trusted-hardware (e.g., ARM TrustZone, OP-TEE)
- Vulnerable to side-channel attacks → defenses are orthogonal with Obscuro



# Conclusion



# Conclusion

- Obscuro Bitcoin mixer is ***efficient***
  - ✓ maintains the simple centralized architecture to achieve high performance (e.g., mix thousand in seconds).

# Conclusion

- Obscuro Bitcoin mixer is **efficient**
  - ✓ maintains the simple centralized architecture to achieve high performance (e.g., mix thousand in seconds).
- Obscuro Bitcoin mixer is **secure**
  - ✓ prevents a malicious operator from misbehaving using TEEs.
  - ✓ guarantees any user can join the mix, and get refunded if the mix is unsuccessful.

# Conclusion

- Obscuro Bitcoin mixer is **efficient**
  - ✓ maintains the simple centralized architecture to achieve high performance (e.g., mix thousand in seconds).
- Obscuro Bitcoin mixer is **secure**
  - ✓ prevents a malicious operator from misbehaving using TEEs.
  - ✓ guarantees any user can join the mix, and get refunded if the mix is unsuccessful.
- Obscuro is **available today**
  - ✓ Obscuro is implemented with Intel SGX and available at:  
<https://github.com/BitObscuro/Obscuro>

# Question?

Muoi Tran

[muoitran@comp.nus.edu.sg](mailto:muoitran@comp.nus.edu.sg)