



# TumbleBit:

## An Untrusted Bitcoin-Compatible Anonymous Payment Hub

Ethan Heilman, Leen AlShenibr, Foteini Baldimtsi,  
Alessandra Scafuro, Sharon Goldberg



NDSS Symposium 2017

BOSTON  
UNIVERSITY

# Bitcoin faces two technical challenges:



## Scalability:

### Transaction velocity:

Bitcoin takes 10 minutes to confirm a transaction.

### Transaction volume:

Bitcoin can only confirm 3-7 transactions/sec (MAX).  
VISA handles 2000 transactions/sec (AVG).

### Why these scalability limitations?

Bitcoin confirms 1MB of transaction data ~every ten minutes.



## Privacy:

Payments in Bitcoin are not private.

## 10. Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.

*Satoshi Nakamoto, 2008*

**Bitcoin offers privacy—as long as you don't cash out or spend it**

**A Fistful of Bitcoins: Characterizing Payments Among Men with No Names**

**Quantitative Analysis of the Full Bitcoin Transaction Graph**

Dorit Ron and Adi Shamir

Department of Computer Science and Applied Mathematics,  
The Weizmann Institute of Science, Israel  
{dorit.ron, adi.shamir}@weizmann.ac.il

Sarah Meiklejohn   Marjori Pomarole   Grant Jordan  
Levchenko   Damon McCoy<sup>†</sup>   Geoffrey M. Voelker   Stefan Savage  
University of California, San Diego   George Mason University<sup>†</sup>

ademic study that analyzed Bitcoin's blockchain, or the public ledger that records bitcoin transactions. The ledger shows how bitcoins move from one person to another, represented by 34-character alphanumeric addresses.

# Introduction

## TumbleBit is:

1. **Private:** Unlinkable Bitcoin payments and k-anonymous mixing,
2. **Untrusted:** No one including Tumbler can steal or link payments.
3. **Scalable (many payments):** scales transaction velocity and volume.
4. **Compatible:** Works with today's Bitcoin protocol.

## Why is compatibility hard?

Our protocol must work with highly constrained Bitcoin scripts which provide two very limited cryptographic operations. Transactions can check ECDSA sigs and hash preimages.

## Two ways to use TumbleBit:



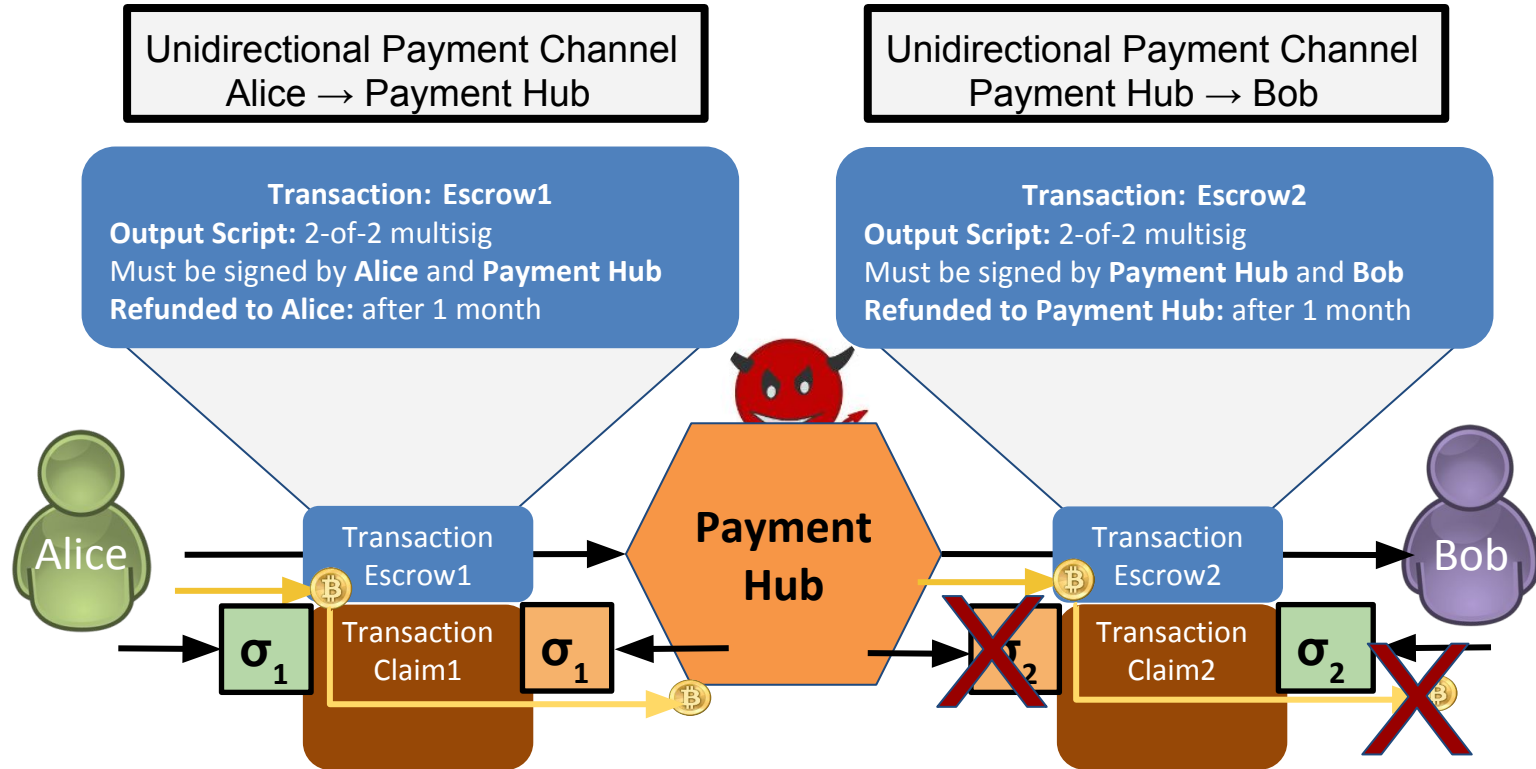
When used to make many payments, TumbleBit helps scale Bitcoin's transaction velocity (faster payments), and transaction volume (more payments).

## When TumbleBit is used to make many payments:

- Unlinkability within the payment phase,
- Payments confirmed in seconds,
- Payments are off-blockchain,  
... don't take up space on the blockchain.

# Background: Payment Hub

**A payment hub:** routes payment channels.



**...But what if the hub is malicious,**

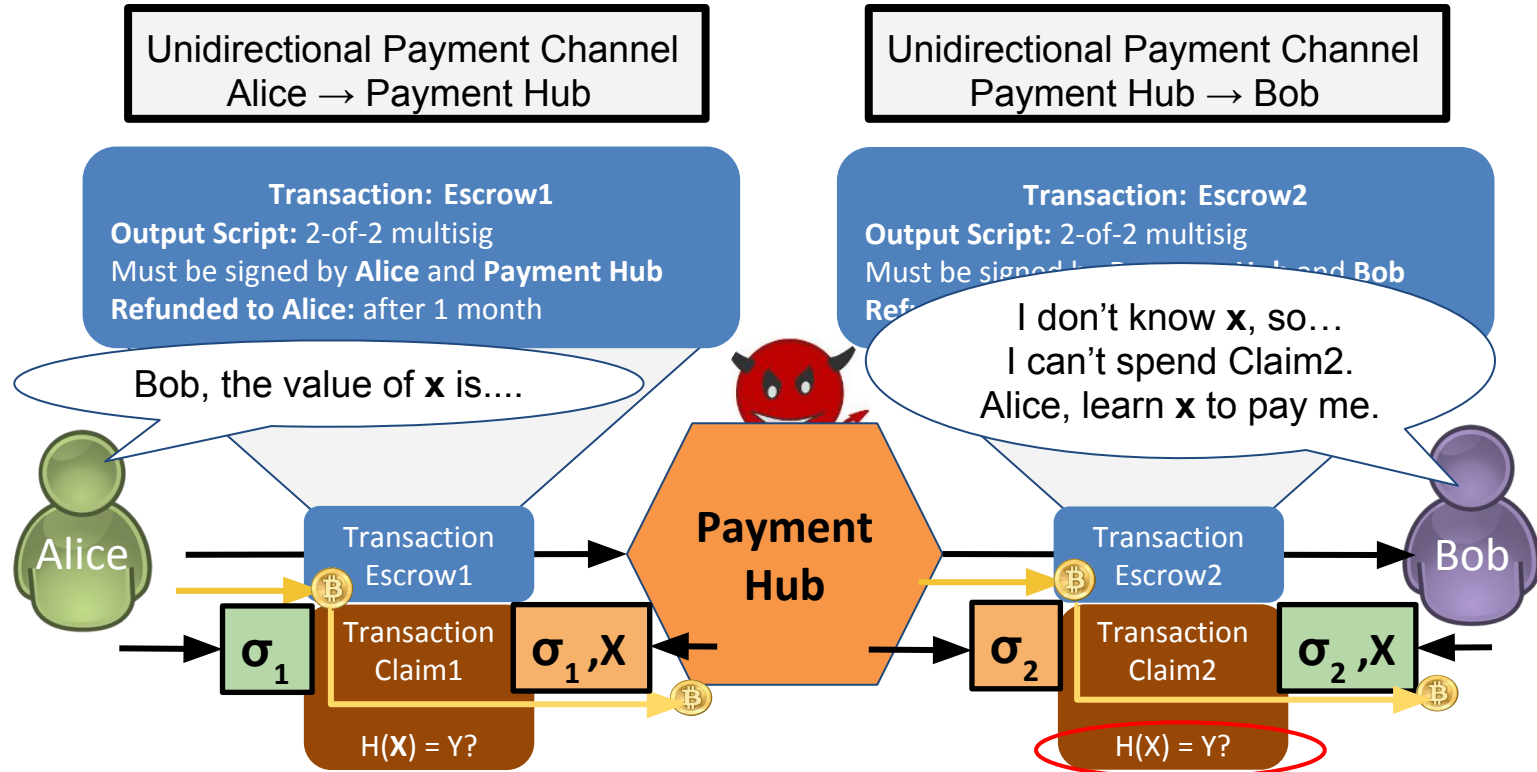
**Atomicity:** If Claim1 and Claim2 happen atomically then theft is prevented.

Hash locks provide this property.



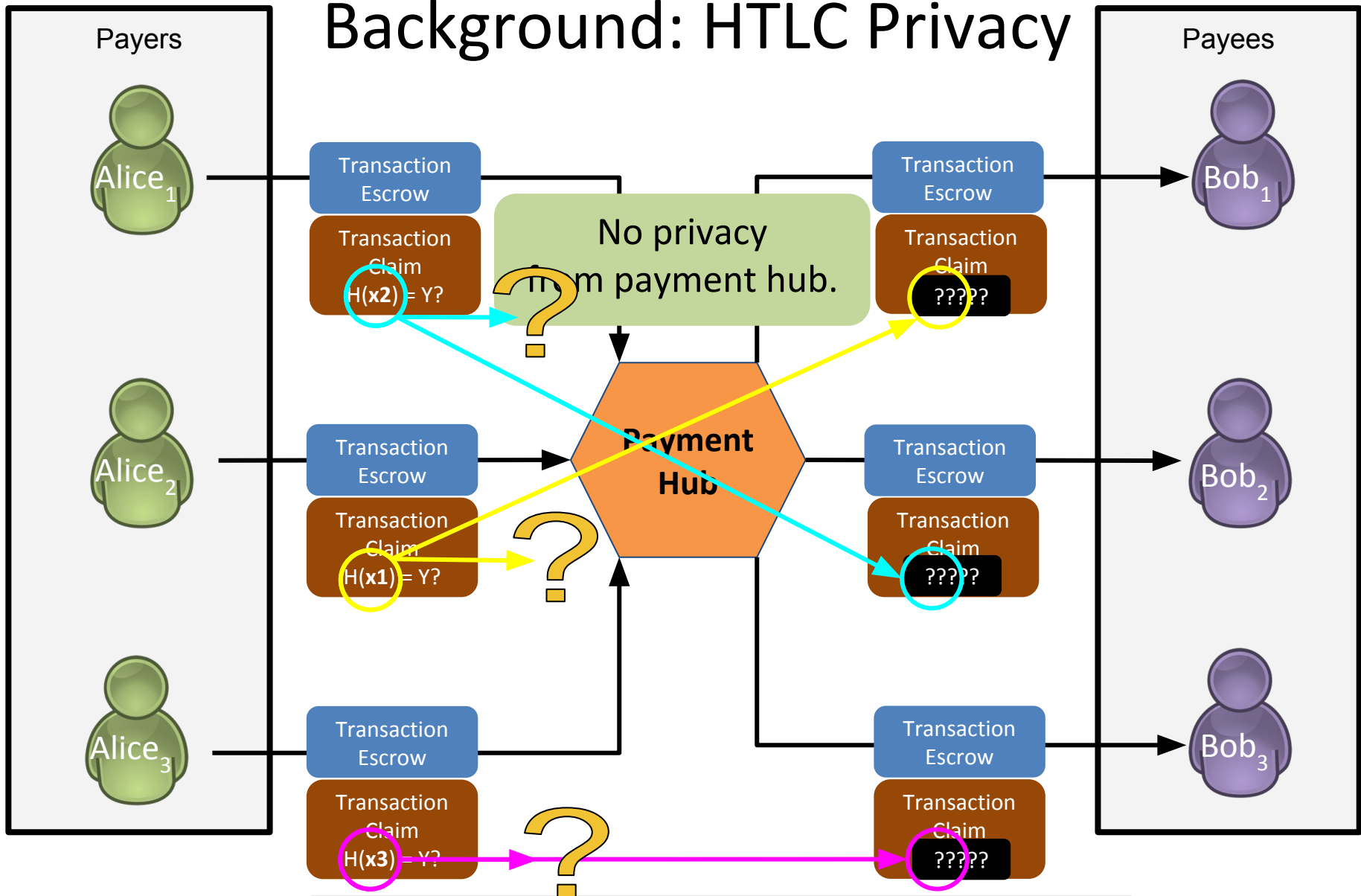
# Background: Payment Hub

**A payment hub:** routes payment channels.



**Thus,** using hash locked transactions or HTLCs a payment hub can prevent theft, however this provides no privacy against the payment hub.

# Background: HTLC Privacy

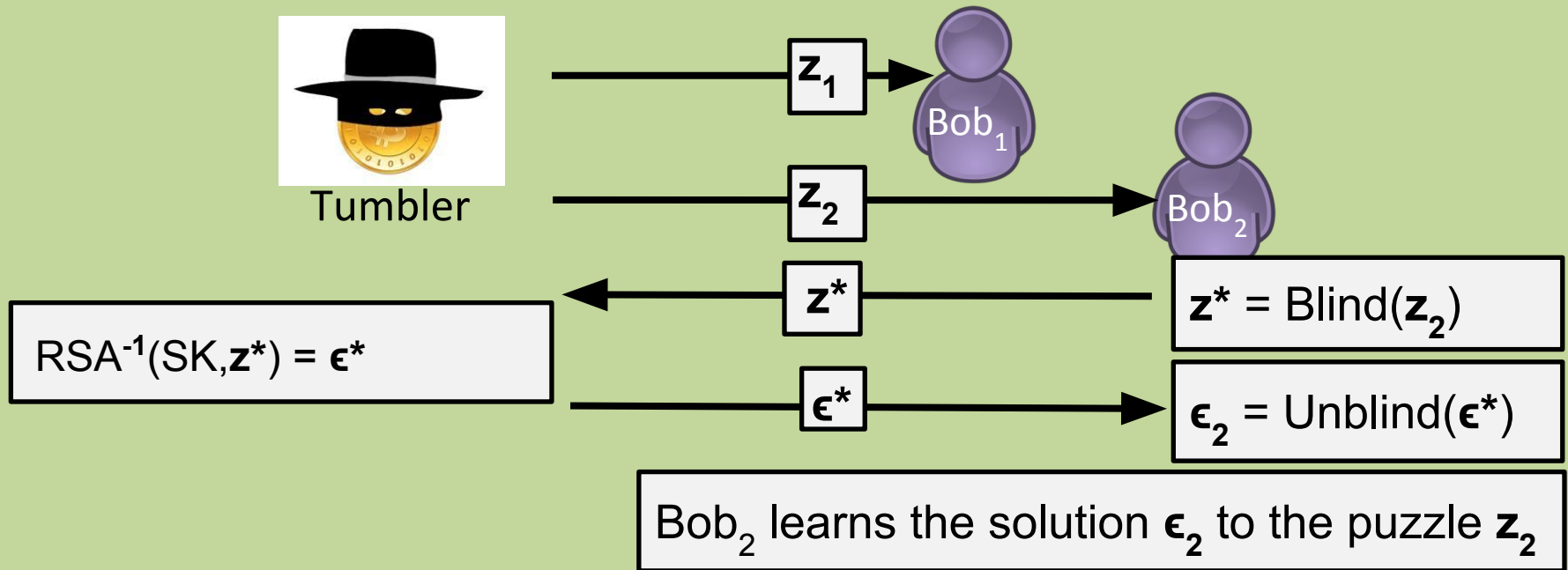


The main idea behind TumbleBit is a protocol which provides **atomicity** but is also **unlinkable** (i.e. private). Think of it like Unlinkable or Private HTLCs.

# Background: RSA Puzzles

- An RSA Puzzle is just a “textbook RSA encryption” of some value  $\epsilon$ :  
$$\text{RSA}(\text{PK}, \epsilon) = z$$
- Only the party that knows SK can solve RSA puzzles:  
$$\text{RSA}^{-1}(\text{SK}, z) = \text{RSA}^{-1}(\text{SK}, \text{RSA}(\text{PK}, \epsilon)) = \epsilon$$

**RSA blinding can be used to blind RSA puzzles**



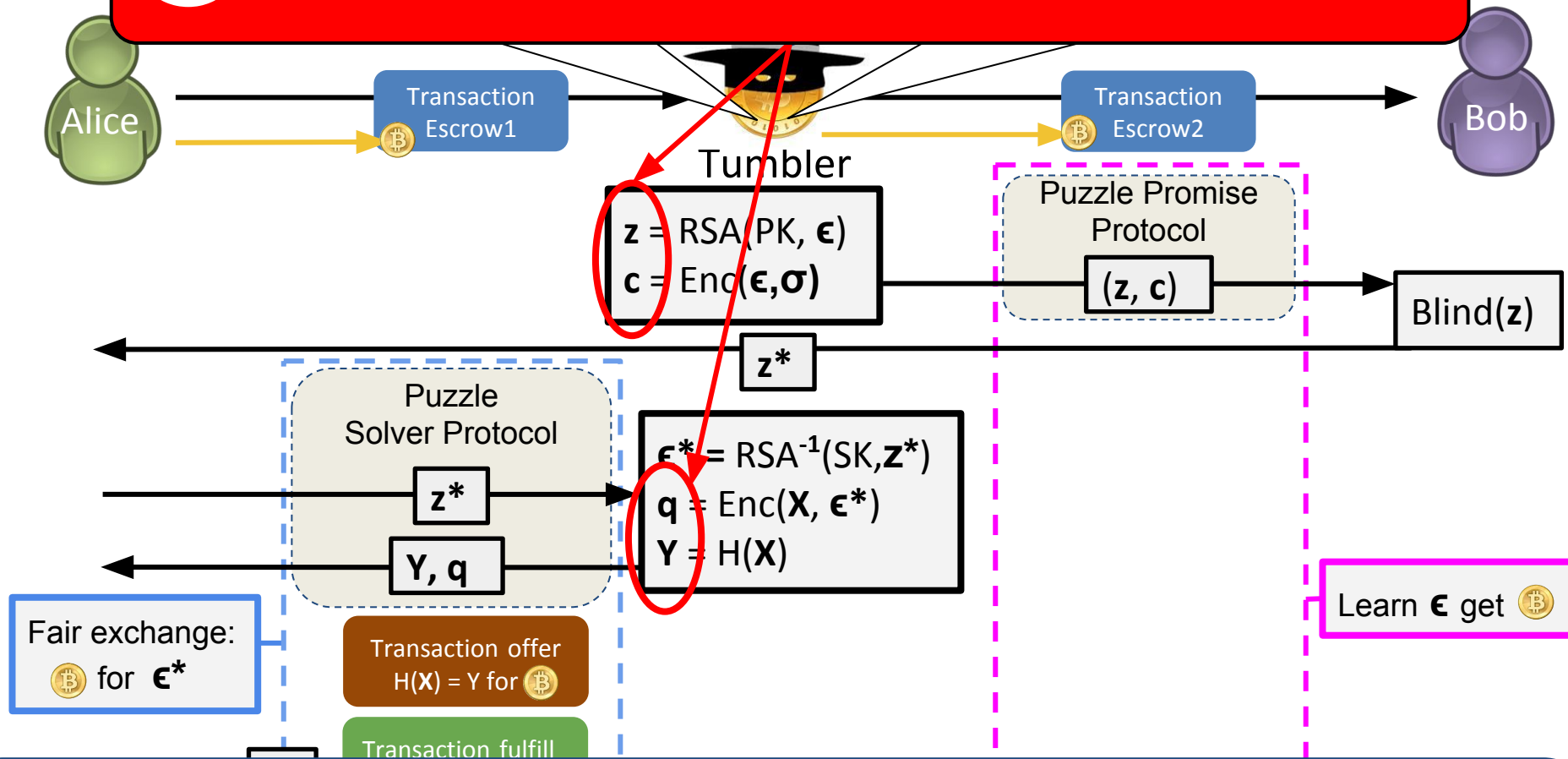
Tumbler can not link the blinded RSA puzzle it solves  $z^*$  to any of the RSA puzzles it issued ( $z_1, z_2$ ).



# TumbleBit: Protocol Overview



If Tumbler corrupts  $z$ ,  $c$ ,  $X$ , or  $q$  it can cheat Alice or Bob!



TumbleBit prevents this via two protocols:

## Puzzle-Solver-Protocol:

Tumbler convinces Alice the preimage  $X$  where  $\text{Hash}(X) = Y$  will allow her to learn  $\epsilon^*$ .

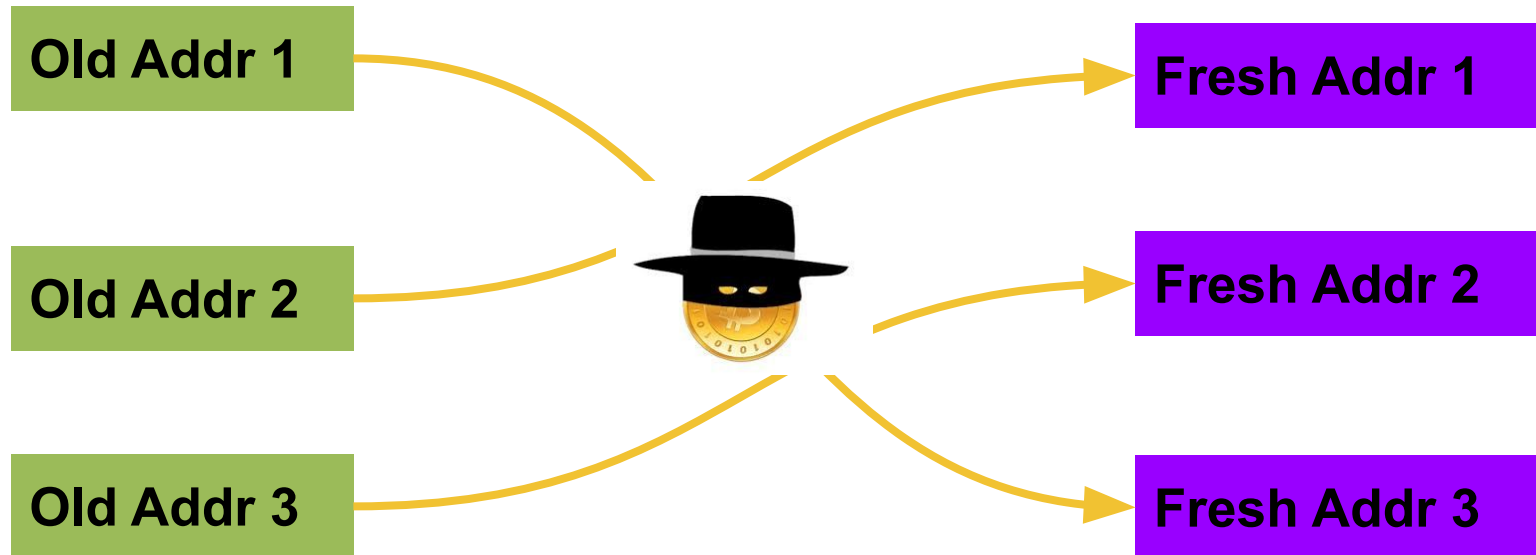
## Puzzle-Promise-Protocol:

Tumbler convinces Bob that the solution to RSA puzzle  $z$  is a value  $\epsilon$  which allows him learn  $\sigma$ .

# TumbleBit: Classic Tumbler

**TumbleBit as a classic tumbler:**

Allows users to privately move bitcoins to an unlinked fresh address.

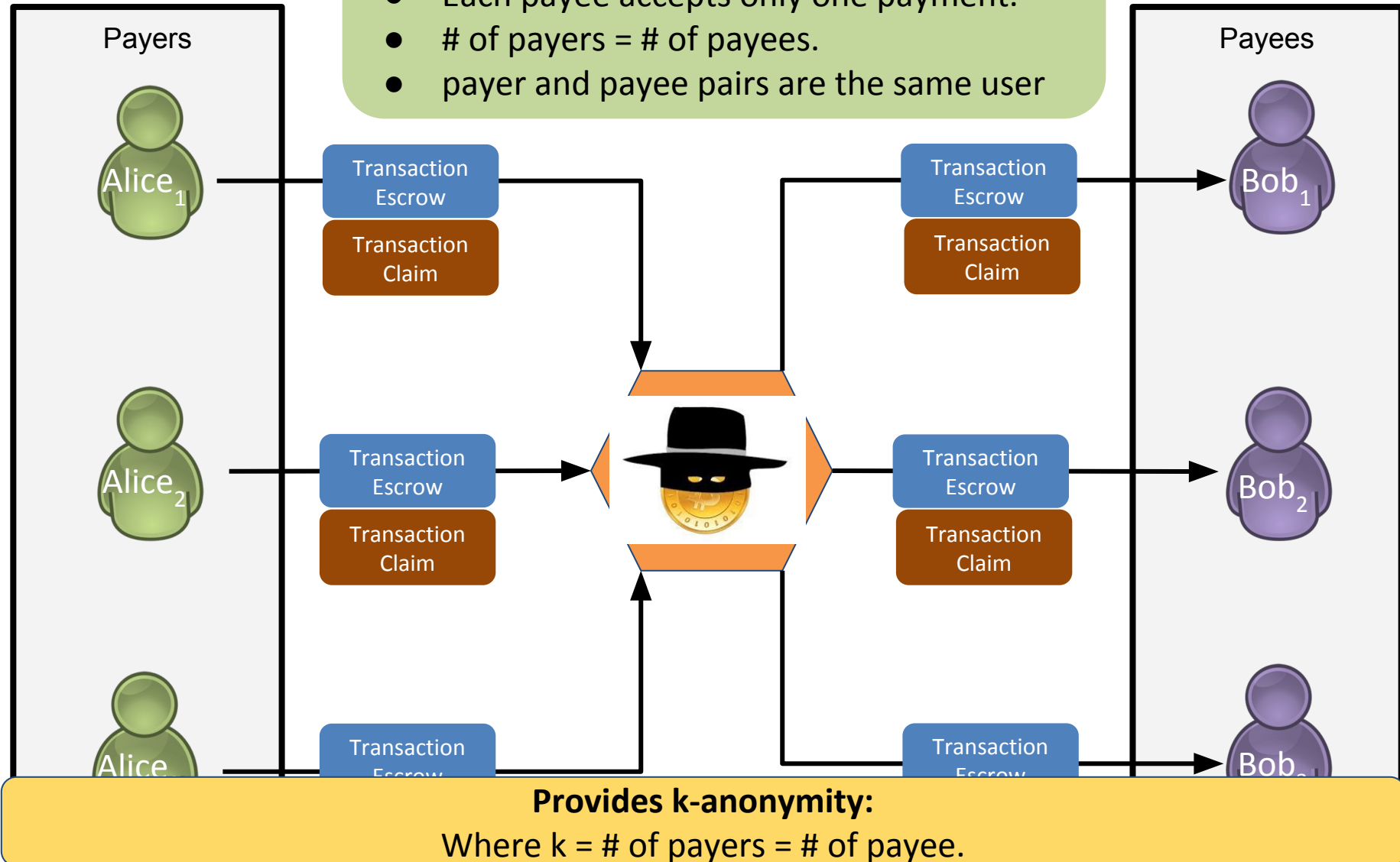


This is also sometimes known as a mixing service or mix.

# Background: Classic Tumbler

## To run TumbleBit as a Classic Bitcoin Tumbler:

- Each payer just makes one payment.
- Each payee accepts only one payment.
- # of payers = # of payees.
- payer and payee pairs are the same user



# TumbleBit: Implementation

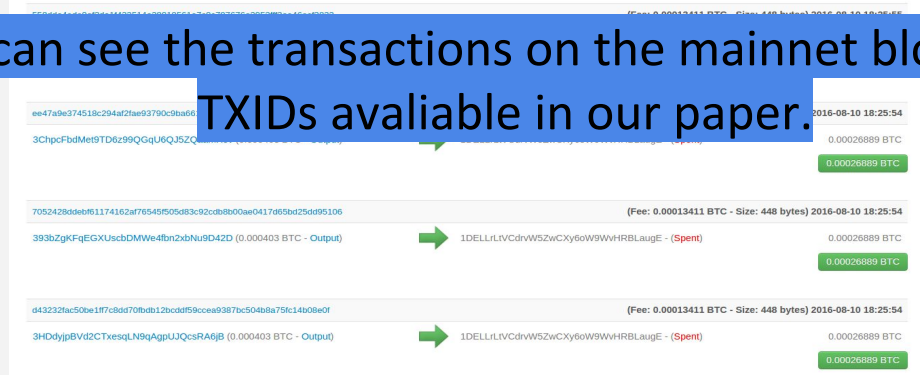
**We wrote a proof-of-concept implementation of the Classic Tumbler:**

- Sourcecode is available on Github.

**We “tumbled” 800 users:**

You can see the transactions on the mainnet blockchain.

TXIDs available in our paper.



**Our implementation is Performant (per TumbleBit payment):**

- 326 KB of Bandwidth,
- Puzzle-Solver takes ~0.4 seconds to compute
- Total time depends on network latency:
  - No latency ~0.6 seconds.
  - Boston to Tokyo ~6 seconds (clear) and ~11 seconds
  - ...(both parties use TOR)

# Conclusion

**TumbleBit provides,  
private untrusted scalable payments via today's Bitcoin**

1. **Private:** Unlinkable or k-anonymous payments
2. **Trustless:** Tumbler can not steal or link payments.
3. **Scalable (payment hub):** scales Bitcoin's transaction velocity and volume.

**We have running code (for TumbleBit classic tumbler):**

- Our code runs on Bitcoin's mainnet blockchain.
- We have published our code on github.
- ...the opensource project nTumbleBit is developing production ready code.



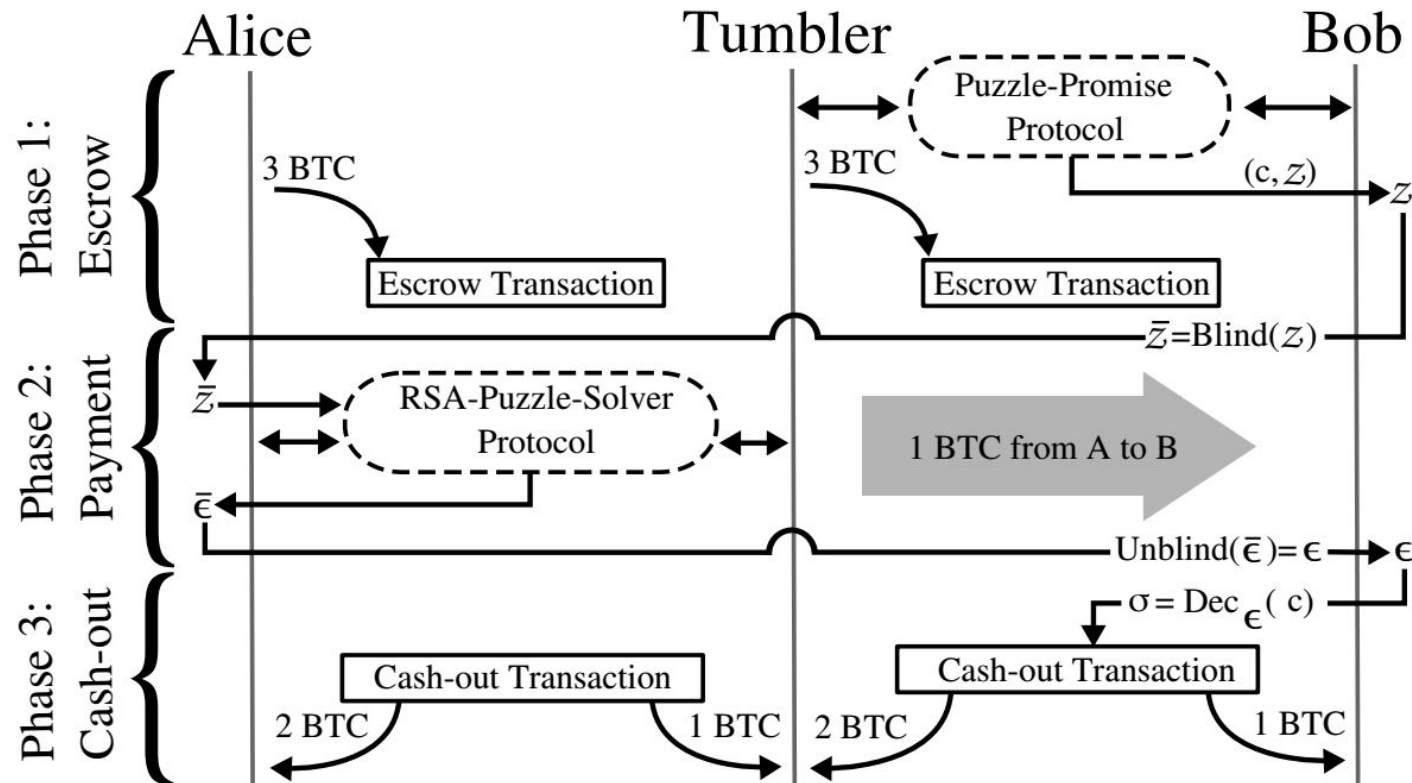
Clone nTumbleBit at  
<https://github.com/nTumbleBit/nTumbleBit>

# Questions?

**Proof-of-concept:** <https://github.com/BUSEC/TumbleBit>

**Opensource project:** <https://github.com/nTumbleBit/nTumbleBit>

**Paper:** <https://eprint.iacr.org/2016/575.pdf>



**Ask questions on twitter:** @Ethan\_Heilman



# TumbleBit: Phases

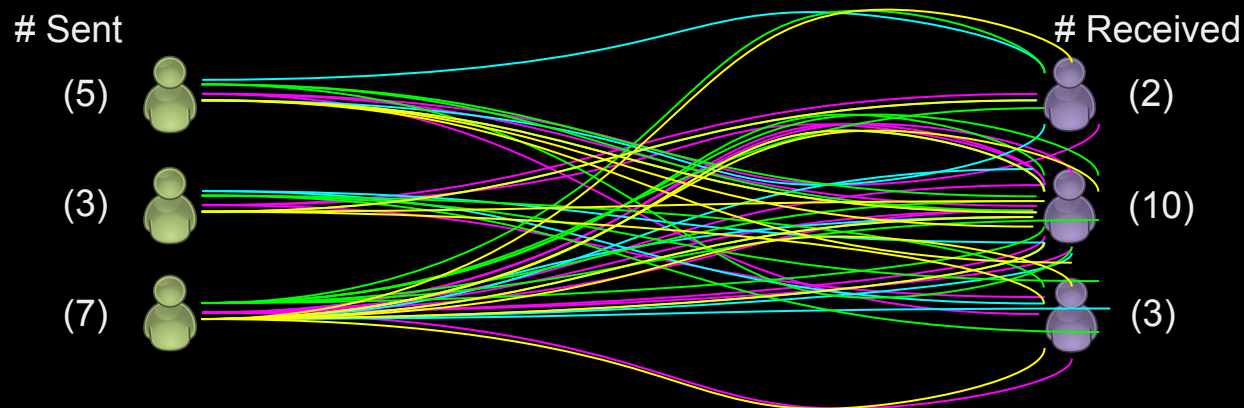
## Privacy offered the TumbleBit Payment Hub

### Tumbler's view:

(1) payer of each payment, (2) # of payments each payee received.

### Unlinkability def:

All interaction graphs compatible with the tumblers view are equally likely.



# TumbleBit: Puzzle-Solver-Protocol

I can't tell which B's are real or fake.



Tumbler

**Fair exchange/contingent payment for an RSA puzzle solution to  $z^*$ .**

1. Alice pays Tumbler if and only if Tumbler solves RSA puzzle  $z^*$
2. Tumbler reveals  $\epsilon^*$  if and only if Alice pays.

$z^*$



Alice

1. Makes  $m$  real puzzles:

for  $i$  in  $m$ :  $D_i = \text{Blind}(z^*, R_i)$

...and  $n$  fake puzzles:

for  $j$  in  $n$ :  $F_j = \text{RSA}(\text{PK}, P_j)$

Shuffle( $D_1, D_2, \dots, D_m, F_1, F_2, \dots, F_n$ )  
 $= (B_1, B_2, B_3, \dots, B_{m+n})$

2. Solves/ encrypts:

for  $i$  in  $m+n$ :

$\epsilon_i = \text{RSA}^{-1}(\text{SK}, B_i)$

$q_i = \text{Enc}(X_i, S_i)$

$Y_i = H(X_i)$

3. Reveal  
by sending

$(a_1, Y_1), (a_2, Y_2), (a_3, Y_3), \dots$

5. Check  
values "i  
correctly computed"


**Probability(Tumbler successfully cheats) =  $(m+n \text{ choose } m) = \sim 1/(2^{80})$**

$m = \# \text{ of real puzzles} = 15$

$n = \# \text{ of fake puzzles} = 285$


6. A proves all real puzzles  
unblind to same puzzle  $z^*$

$(R_1, R_2, \dots, R_m)$

Transaction offer  
 $H(X_1) = Y_1 \text{ AND } H(X_3) \text{ AND } H(X_4) \dots$  for 

7. decrypts  $q$ 's  
learns  $\epsilon^*$

$(X_1, X_3, X_4, \dots)$

Transaction fulfill  
 $X_1, X_3, X_4, \dots$  

If Tumbler computes any  $(q_i, \epsilon_i, Y_i)$  of the real puzzles correctly Alice learns  $\epsilon^*$ ,  
**thus** to cheat Alice, Tumbler must corrupt all the real and none of the fake puzzles.

# TumbleBit: Puzzle-Promise-Protocol

**At the end of this protocol:** Bob should be convinced that for a  $(\mathbf{z}, \mathbf{c})$ :

1. The ciphertext  $\mathbf{c}$  decrypts to  $\sigma$  under a key  $\epsilon$  i.e  $\text{Dec}(\epsilon, \mathbf{c}) = \sigma$
2. **AND** the key  $\epsilon$  is the solution to the RSA-puzzle  $\mathbf{z}$ .

**The protocol should never:** allow Bob to learn a valid  $\sigma$  (without paying).



1. B sends: a mix of hashes of valid and invalid claim transactions.

$\mathbf{B} = H(T1), H(\text{invalid}), H(\text{invalid}), H(T4), H(\text{invalid}), H(T6)$

2. T Signs & Encrypts  $\sigma$ :  
for  $\mathbf{B}_i$  in  $\mathbf{B}$ :

This is why the protocol is hard,  
otherwise Tumbler could convince Bob

$\sigma_i = \text{Sig}(\epsilon_i, \mathbf{B}_i)$   
 $\mathbf{z}_i = \text{RSA}(\mathbf{B}_i)$

**Probability(Tumbler successfully cheats) =  $(m+n \text{ choose } m) = \sim 1/(2^{80})$**

$m = \# \text{ of valid transactions} = 42$

$n = \# \text{ of invalid transactions} = 42$

4. T Reveals:  $\epsilon_i$  for  
invalid transactions.

$\epsilon_2, \epsilon_3, \epsilon_5$

3. B checks: invalid  
transactions  $\sigma_i$  are  
correctly computed.

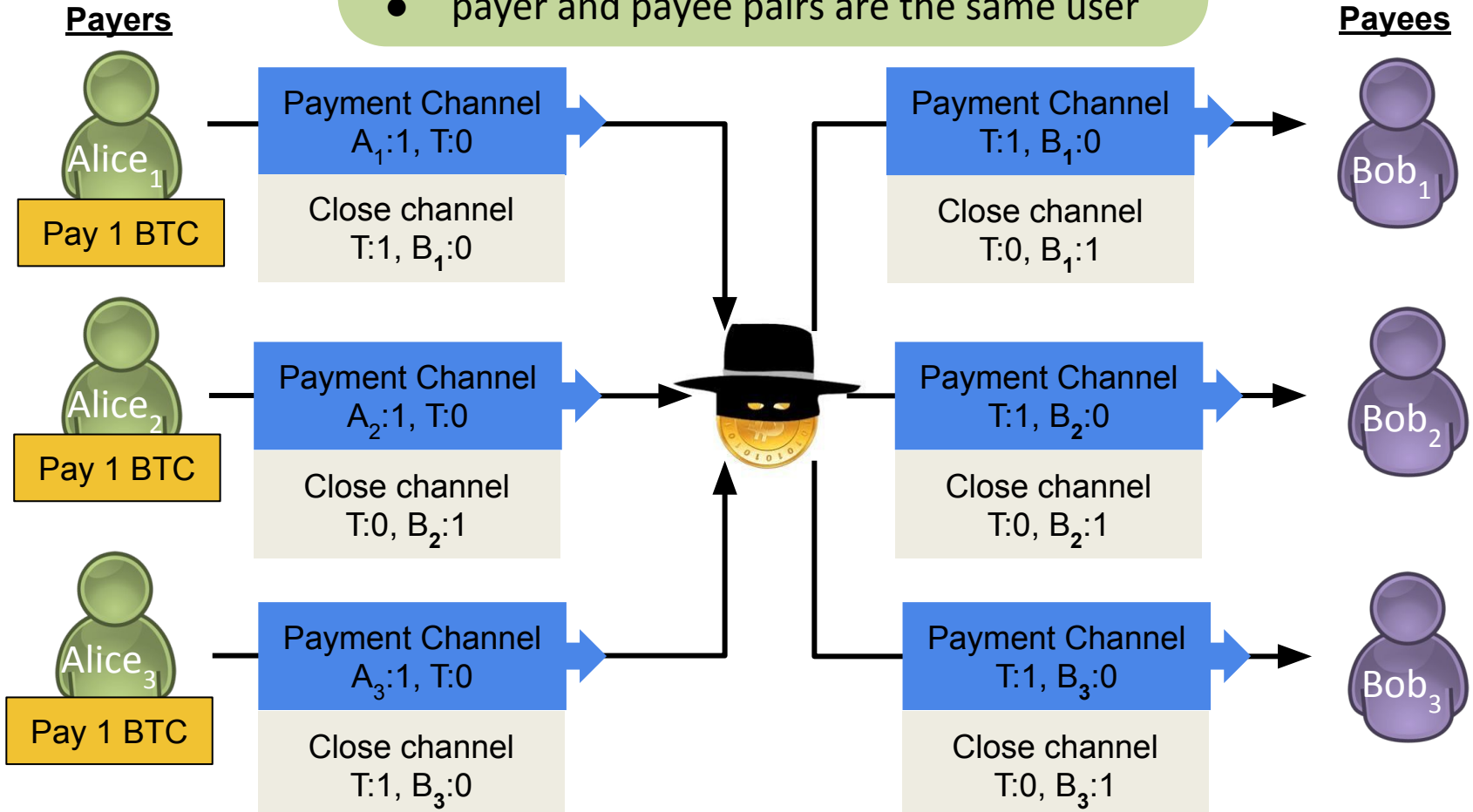
6. Bob and Tumbler run "quotient protocol" ensuring that:  
if Bob learns  $\epsilon_1$ , Bob can use that knowledge to learn  $\epsilon_4, \epsilon_6$ .  
 $(\epsilon_4/\epsilon_1 \bmod N, \epsilon_6/\epsilon_4 \bmod N)$

If Tumbler computes any  $(\epsilon_i, \sigma_i)$  of the valid transactions correctly Bob learns a  $\sigma$ /gets paid,  
**thus** to cheat Bob, Tumbler must corrupt all the valid and none of the invalid transactions.

# TumbleBit: Classic Tumbler

To run TumbleBit as a Classic Bitcoin Tumbler:

- Each payer just makes one payment.
- Each payee accepts only one payment.
- # of payers = # of payees.
- payer and payee pairs are the same user



**Provides k-anonymity:**

Where  $k = \# \text{ of payers} = \# \text{ of payee}$ .

# Compared to other Tumblers

**Vulnerable to DoS &  
Sybil Attacks**



**Limited Anonymity**



**TumbleBit**

**Mixing takes  
hours  
Xim**

**Vulnerable to bitcoin theft**



**True Anonymous Cryptocurrency**

**Blindcoin:**



**Intermediary  
breaks  
anonymity**