TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

S&P SECURITY & PRIVACY GROUP

PURDUE UNIVERSITY

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Security, Privacy and Interoperability in Payment-Channel Networks

Pedro Moreno-Sanchez (@pedrorechez)

Joint work with
Giulio Malavolta, Clara Schneidewind,
Aniket Kate, Matteo Maffei
@aniketpkate @matteo_maffei

Permissionless Blockchains Scalability Issue

- Low transaction rate (~10 transactions per second)
- Fast growth of the Bitcoin transactions
- Scalability approaches:
  - On-chain (layer 1) sharding
  - Off-chain (layer 2) payment channels [The focus of our work]

Payment Channels

Alice

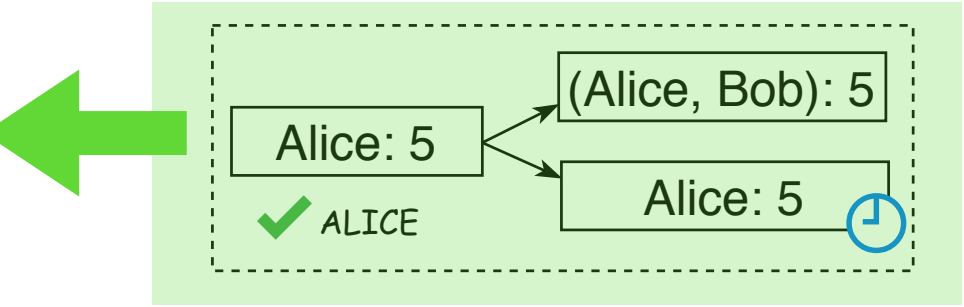
Bob

Payment Channels: Open

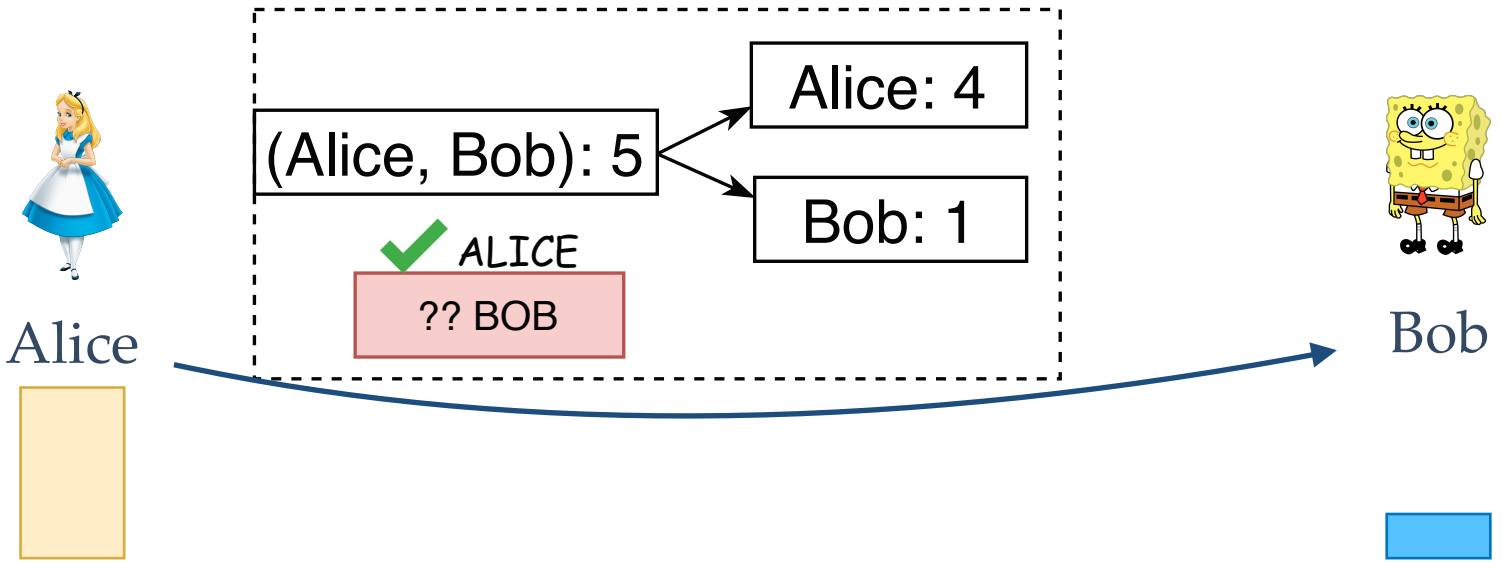
Alice

Bob

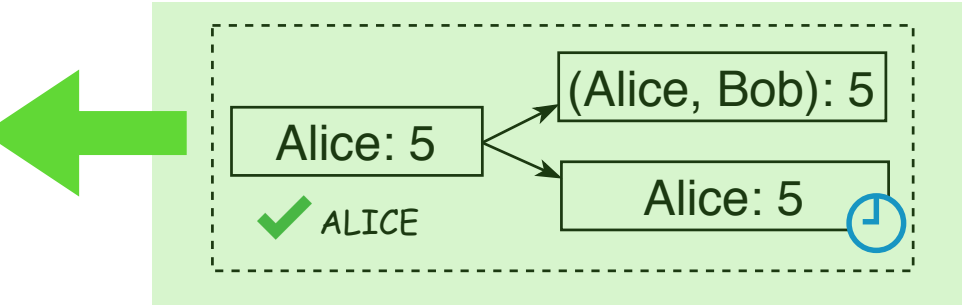
Blockchain



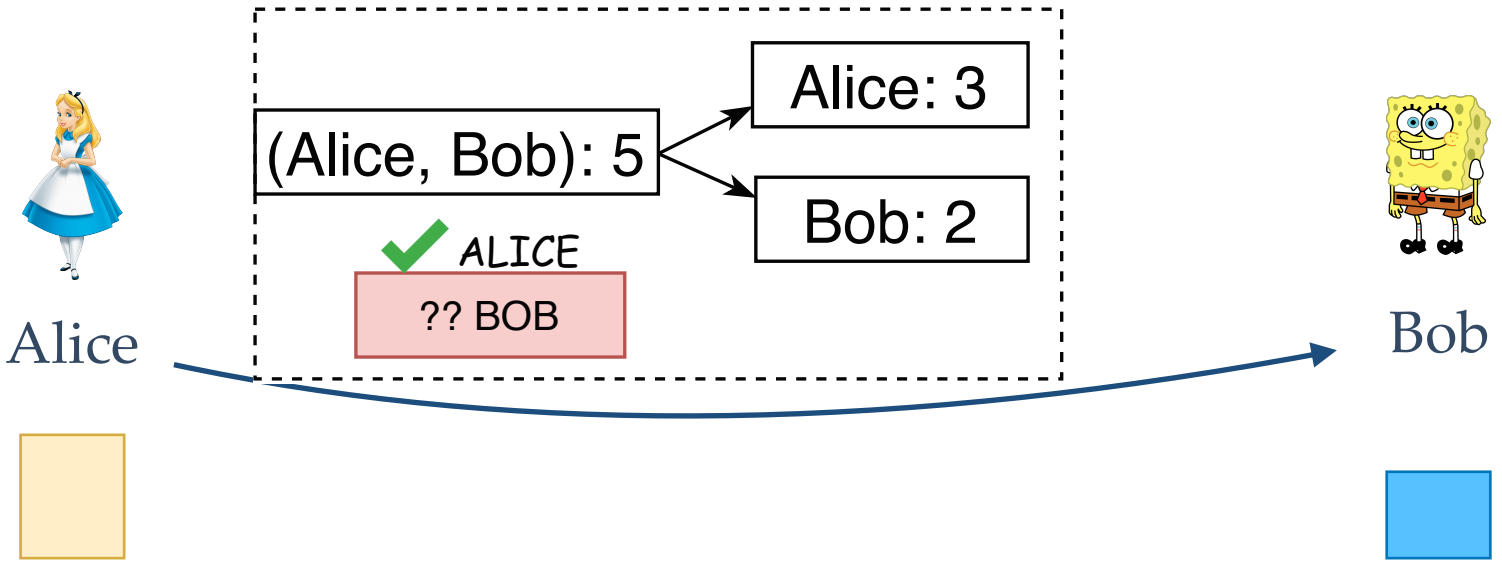
Payment Channels: Pay



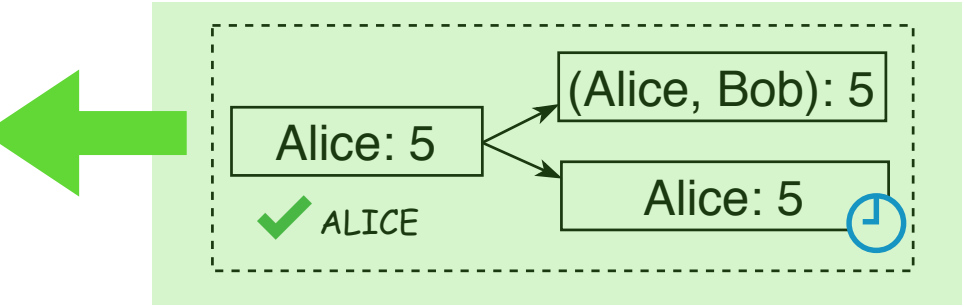
Blockchain



Payment Channels: Pay



Blockchain

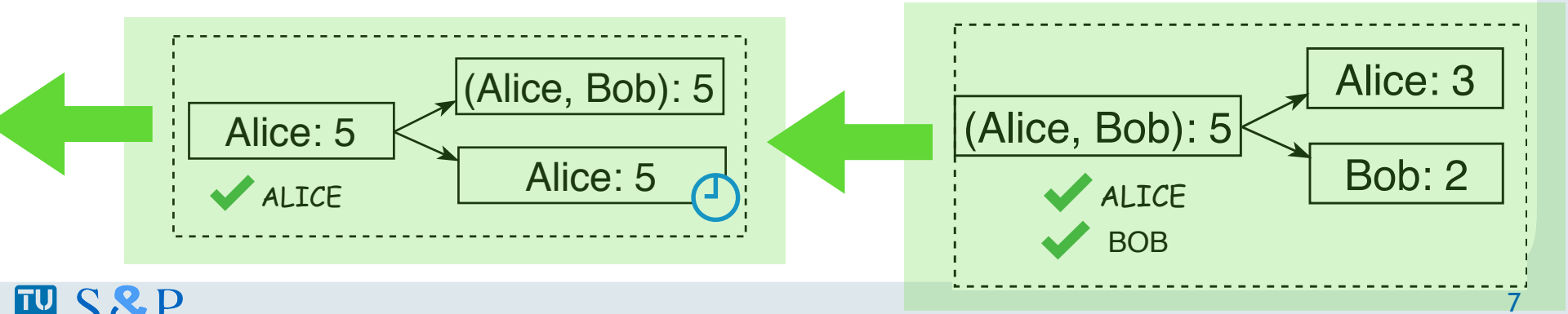


Payment Channels: Close

Alice

Bob

Blockchain

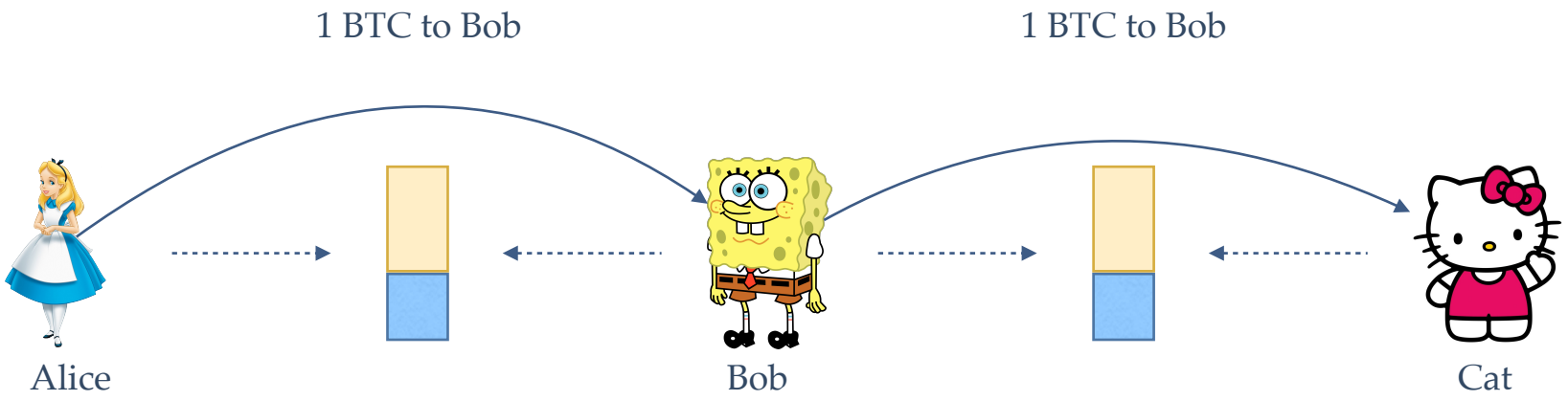


Payment-Channel Networks (PCN)

- Each payment channel requires to lock coins in the deposit
  - Impractical to open a channel with each other
- Open a few channels
  - Rely on other channels to reach the intended receiver

Payment-Channel Networks (PCN)

- Each payment channel requires to lock coins in the deposit
  - Impractical to open a channel with each other
- Open a few channels
  - Rely on other channels to reach the intended receiver



Current PCN (Proposals)

- Bitcoin and Altcoins:
  - Lightning network, c-lighntning, Eclair
- Ethereum:
  - Raiden Network
- Eventually, every blockchain might need a scalability solution

What is our group’s research about in PCN?

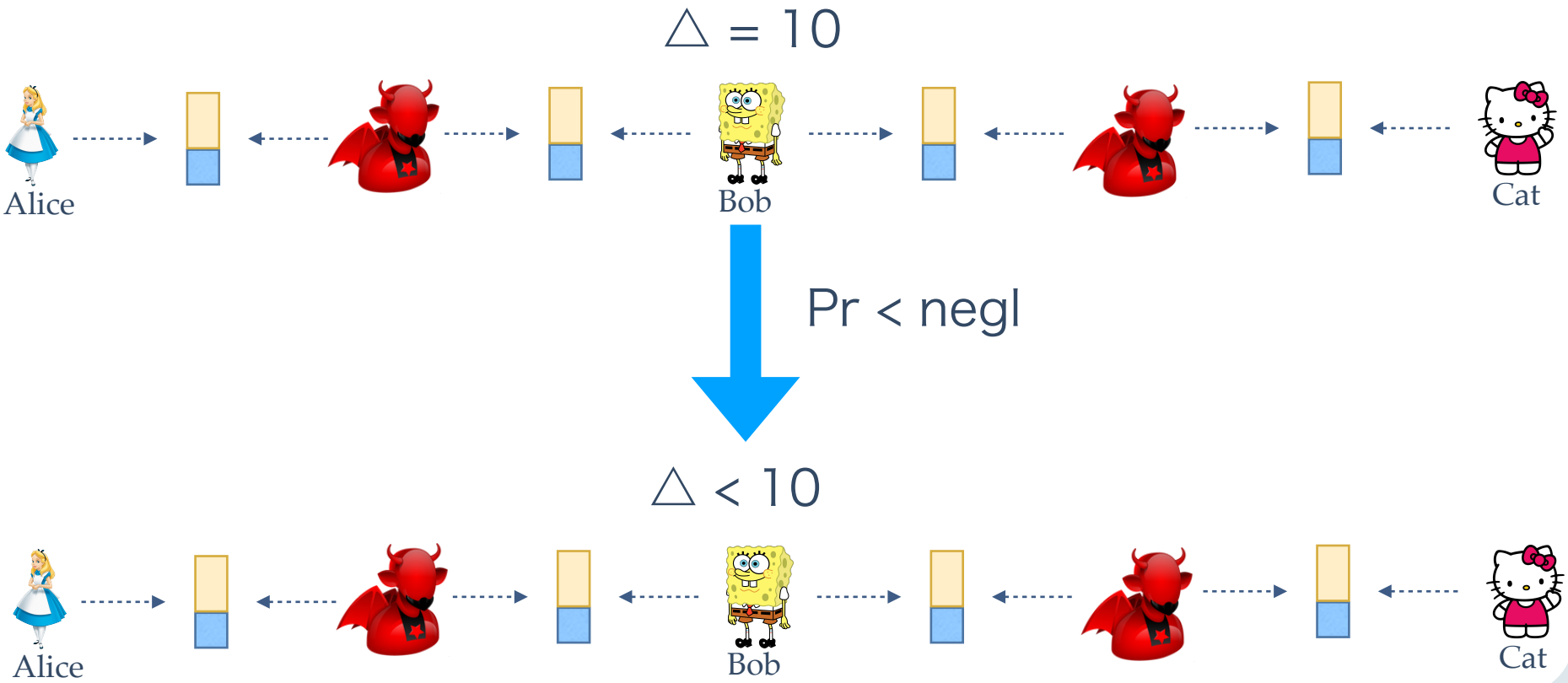
Our Research

- Formally describe notions of interest for PCNs in the Universal Composability framework:
  - Security, privacy, concurrency
- Analyze whether current PCNs achieve them
  - e.g., we showed an inherent tradeoff privacy vs concurrency
- Provide cryptographic constructions with formal security and privacy guarantees

Security in PCNs

Security Notion

- Balance security: Honest users do not lose coins in a multi-hop payment

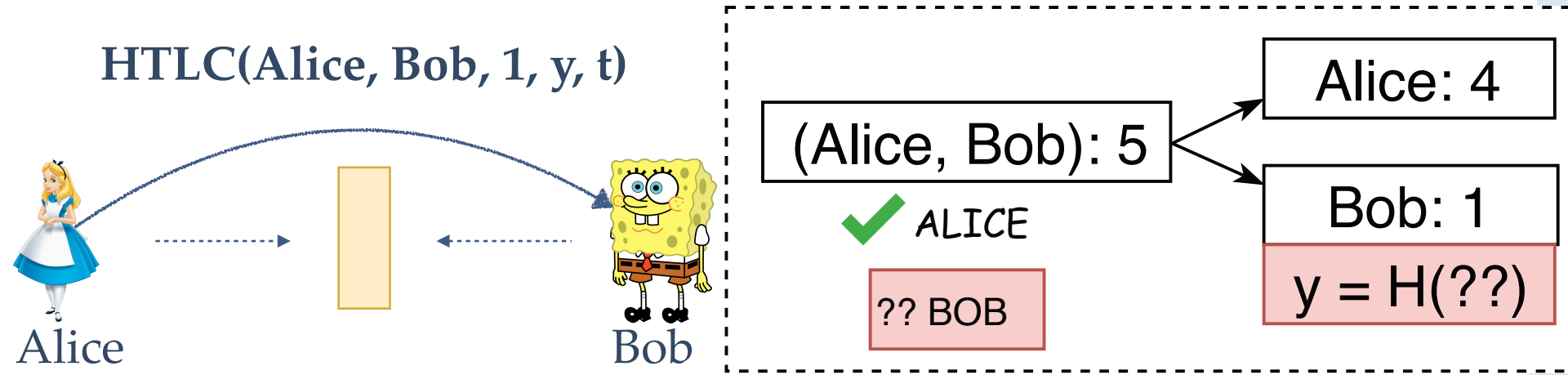


Security and HTLC

- Balance security: Honest users do not lose coins in a payment
- Security tool: Hash-Time Lock Contract (HTLC): Payment conditioned on revealing the pre-image of a hash function

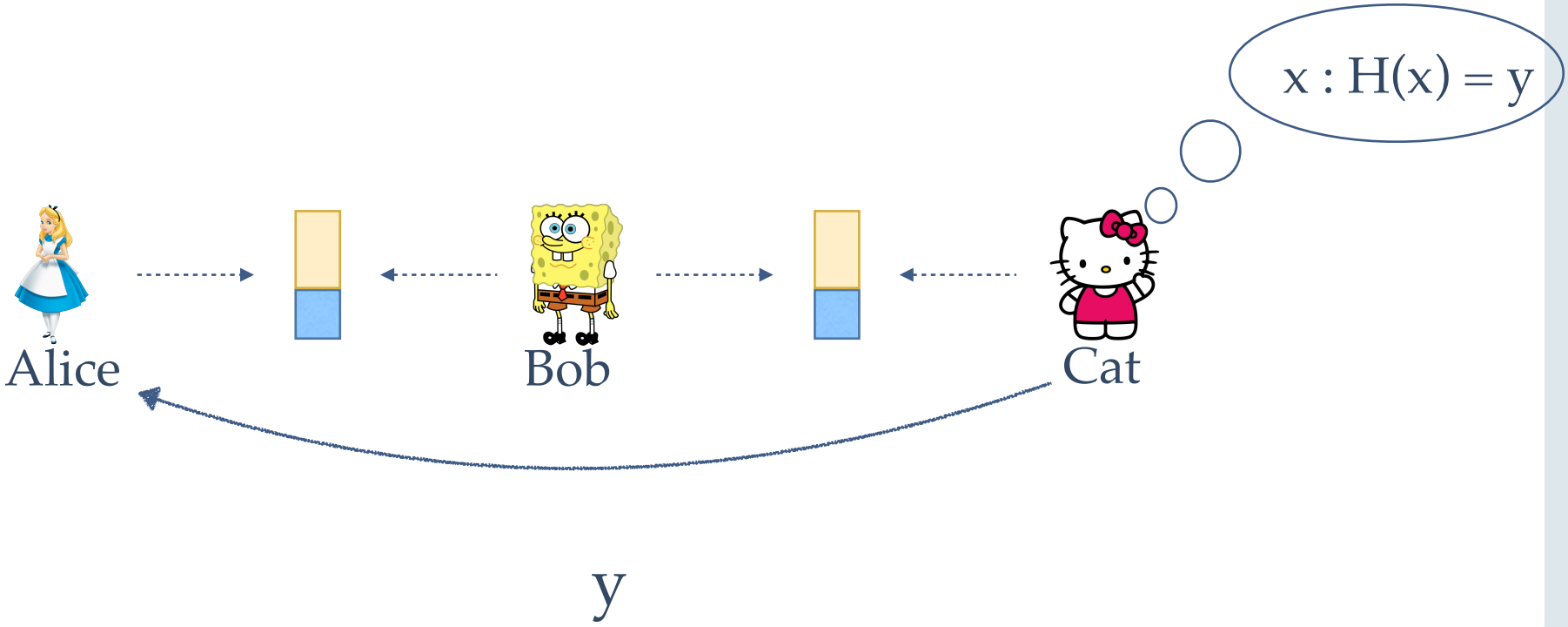
Security and HTLC

- Balance security: Honest users do not lose coins in a payment
- Security tool: Hash-Time Lock Contract (HTLC): Payment conditioned on revealing the pre-image of a hash function



The Lightning Network: Setup

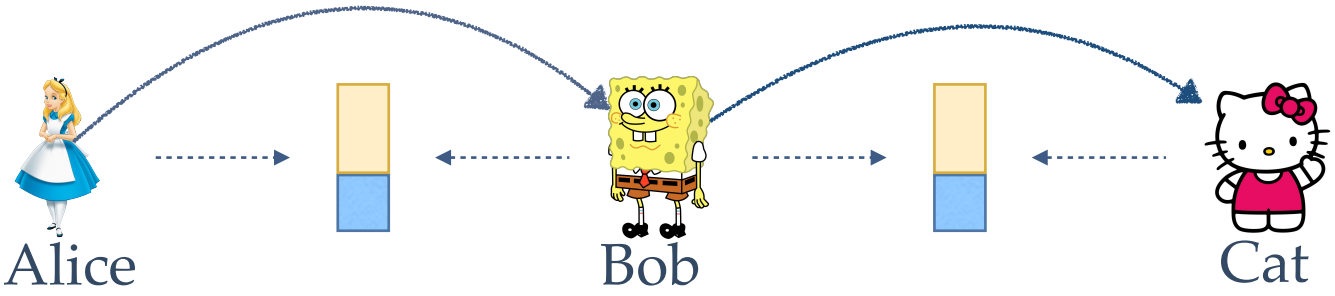
- Multiple "chained" HTLC allow multi-hop payments in the presence of malicious intermediaries



The Lightning Network: Lock

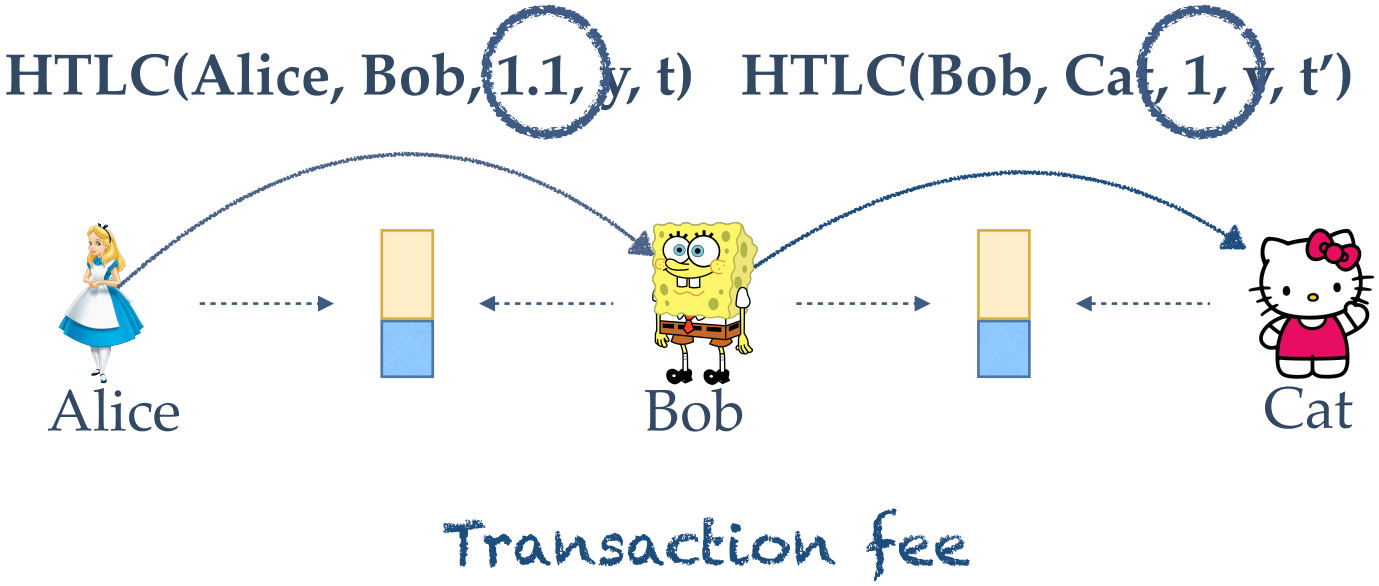
- Multiple "chained'' HTLC allow multi-hop payments in the presence of malicious intermediaries

HTLC(Alice, Bob, 1.1, y, t) HTLC(Bob, Cat, 1, y, t')



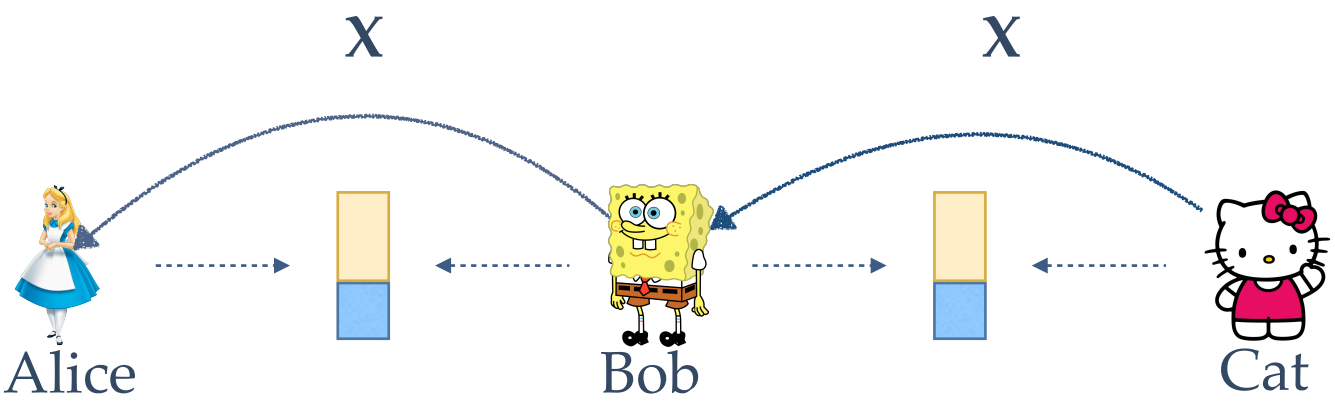
The Lightning Network: Lock

- Multiple "chained" HTLC allow multi-hop payments in the presence of malicious intermediaries



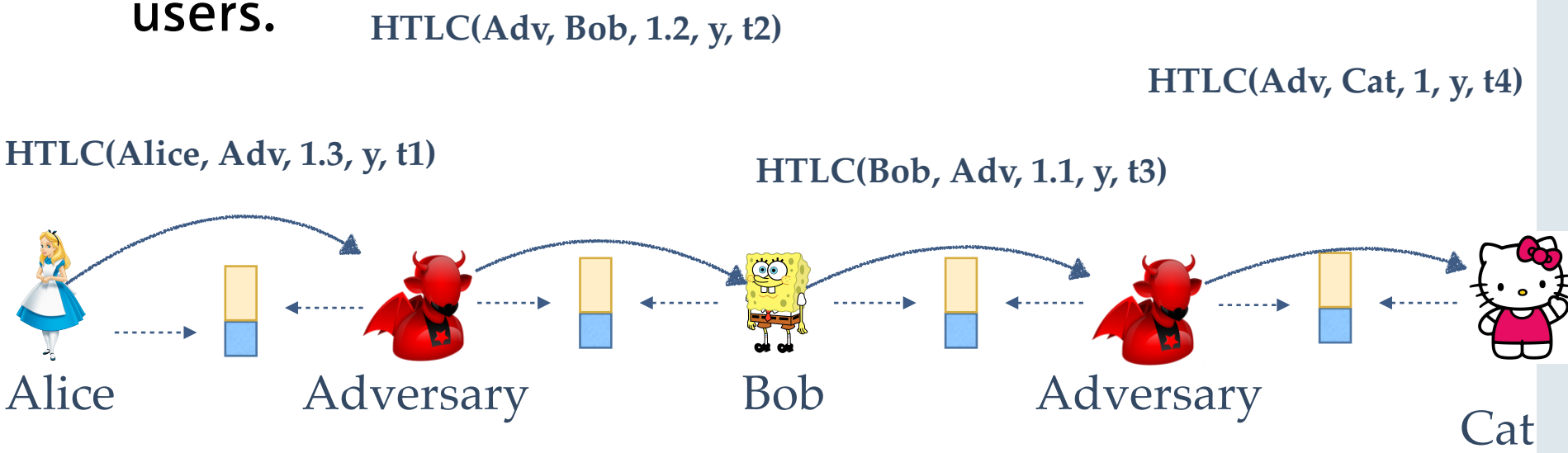
The Lightning Network: Release

- Multiple "chained" HTLC allow multi-hop payments in the presence of malicious intermediaries



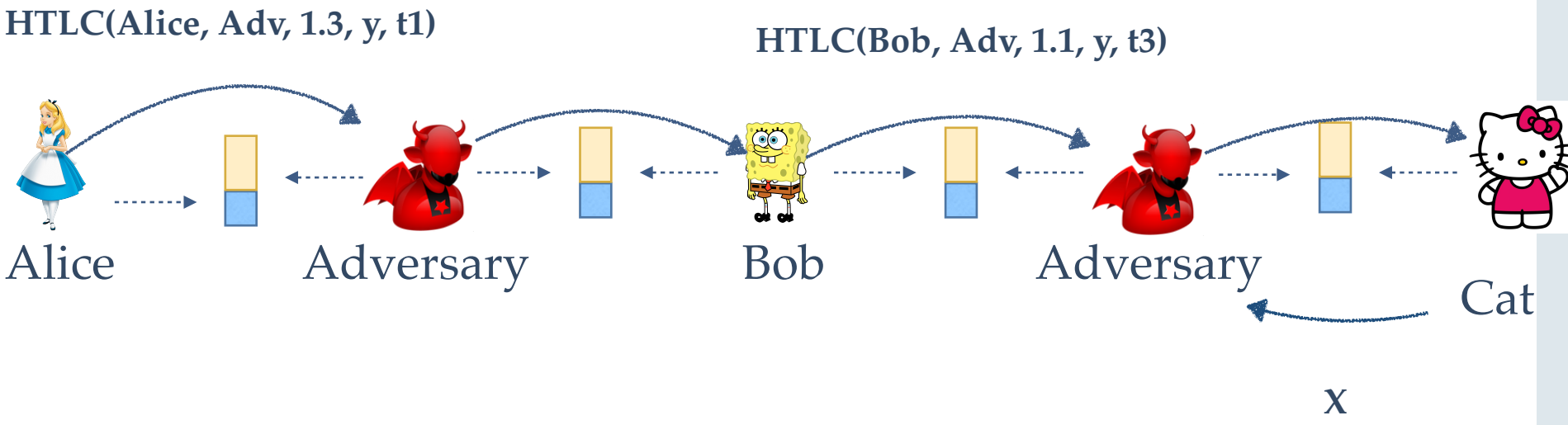
A Novel Wormhole Attack

- Idea: Exclude intermediate honest users from successful completion.
- Consequence: Adversary steals fees from honest users.



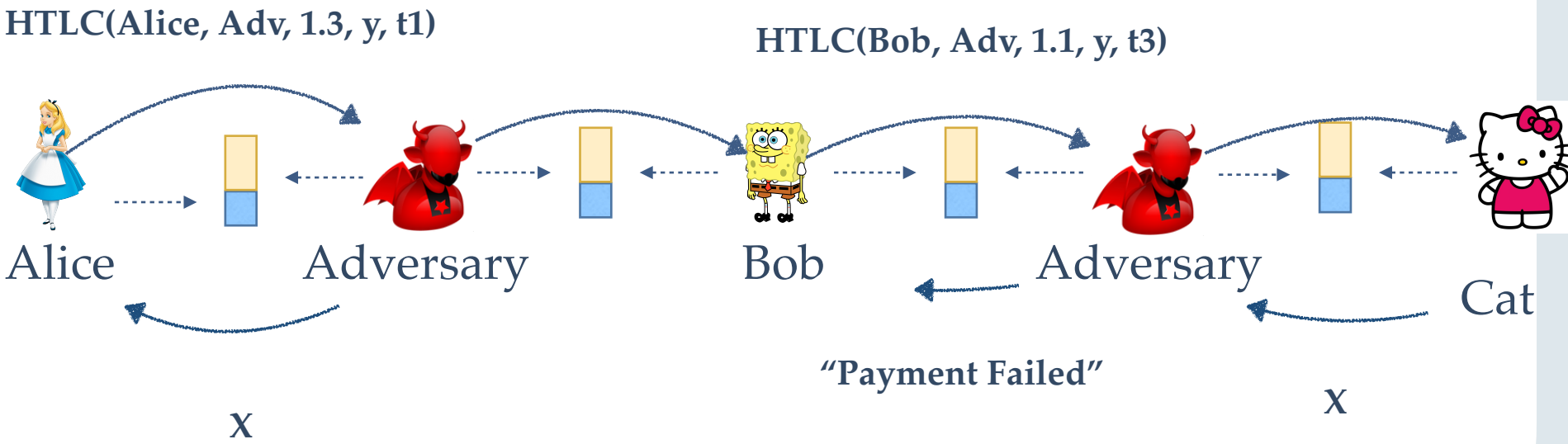
A Novel Wormhole Attack

- Idea: Exclude intermediate honest users from successful completion.
- Consequence: Adversary steals fees from honest users.



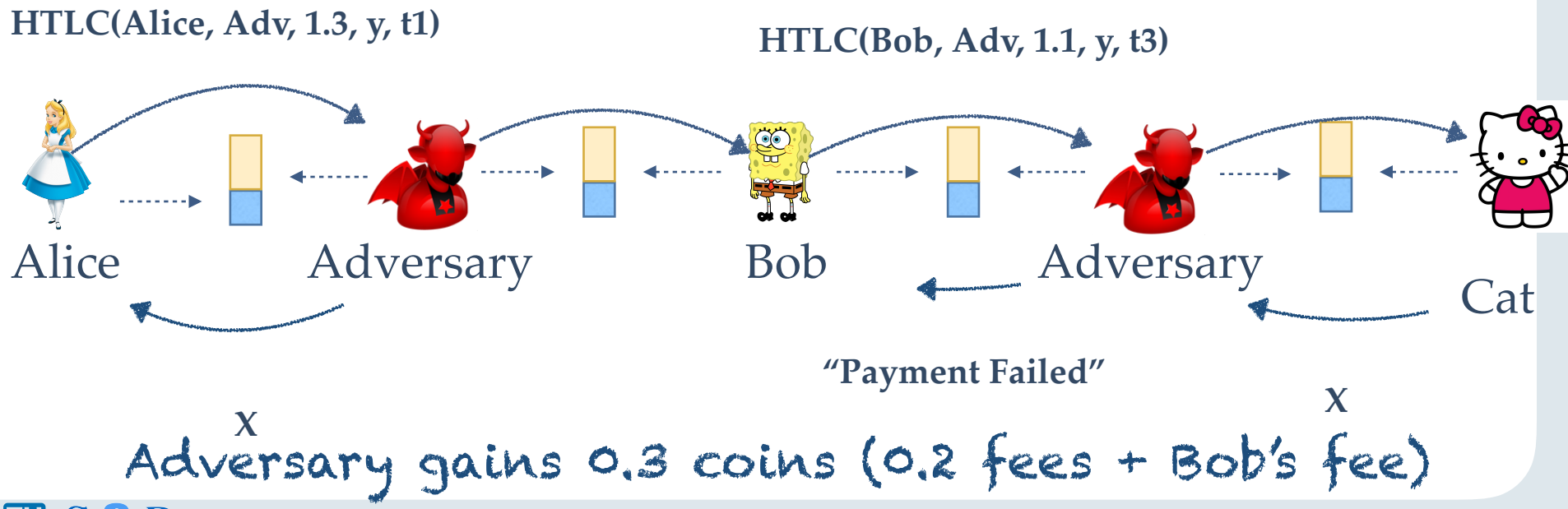
A Novel Wormhole Attack

- Idea: Exclude intermediate honest users from successful completion.
- Consequence: Adversary steals fees from honest users.



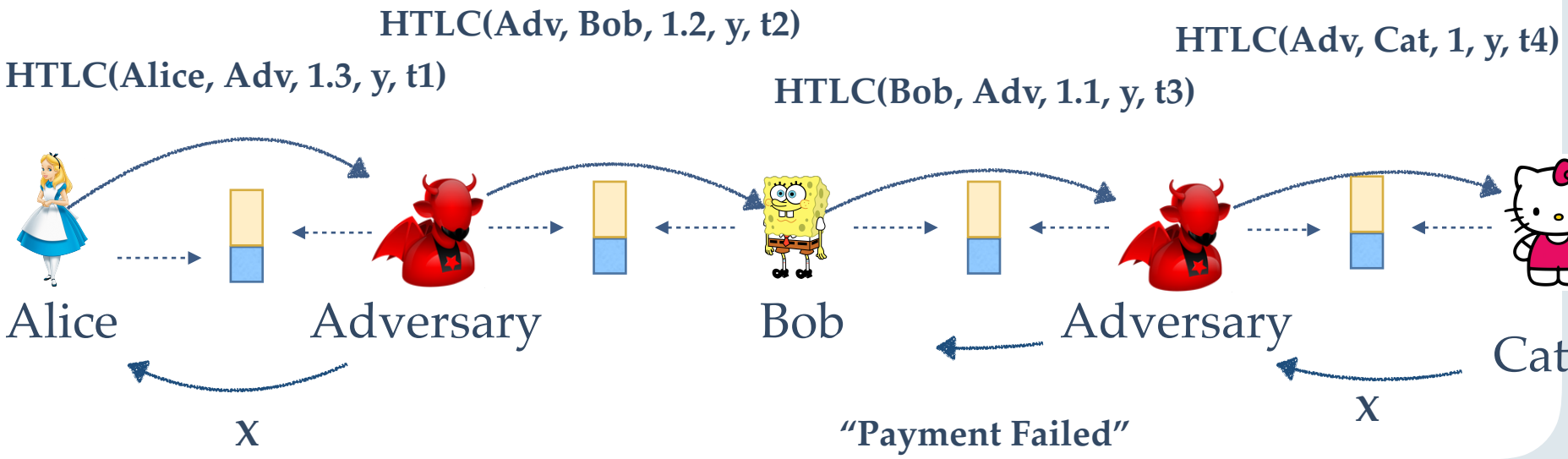
A Novel Wormhole Attack

- Idea: Exclude intermediate honest users from successful completion.
- Consequence: Adversary steals fees from honest users.



The Wormhole Attack: Discussion

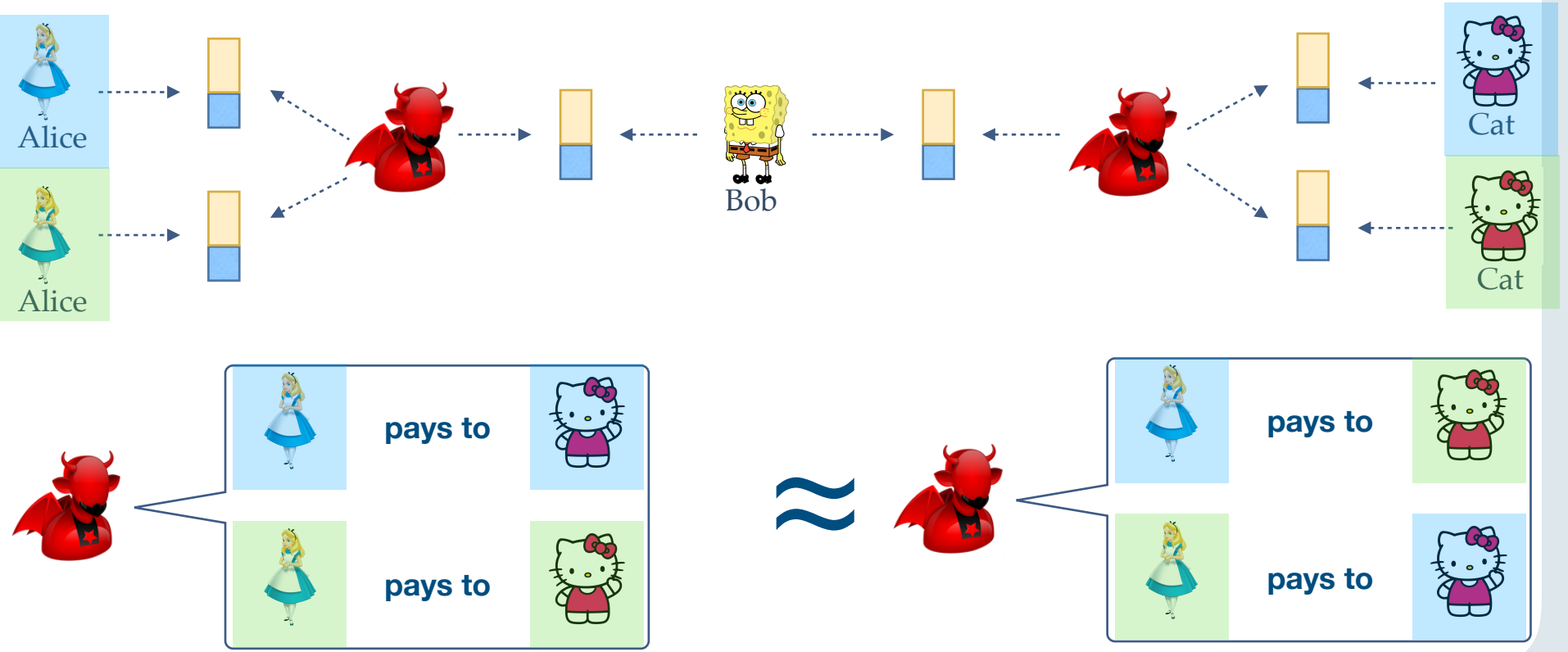
- Same condition along the path enables this attack
- More intermediaries, more benefit
- Fees are the base of PCNs. Thus, attack on fees is important
- Intermediary (Bob) believes payment is unsuccessful



What about privacy?

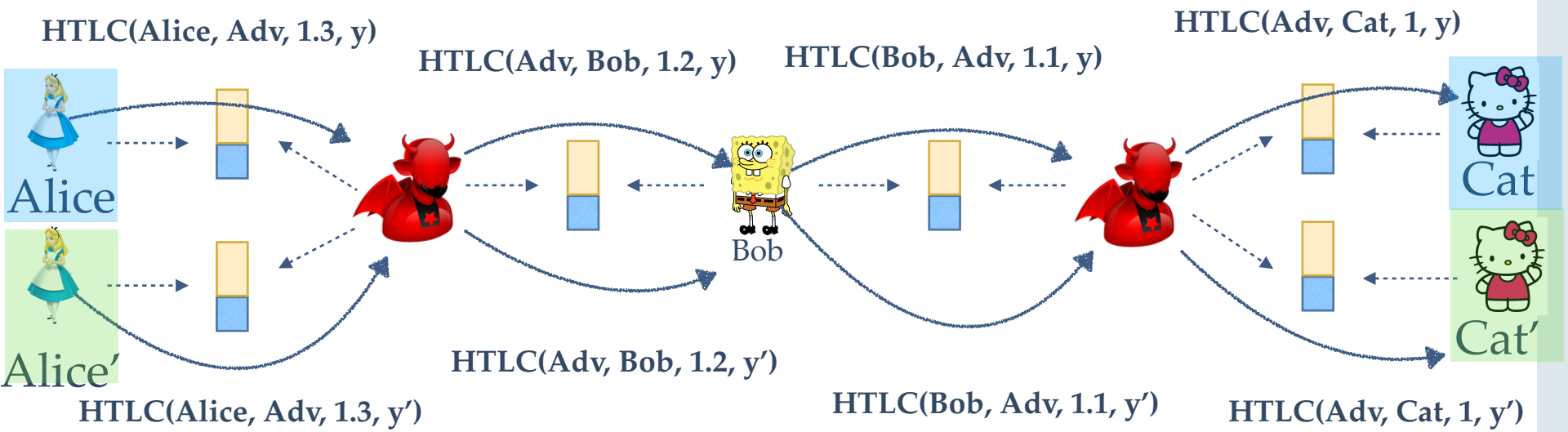
Privacy Notion

- Relationship Anonymity: The adversary cannot tell who is paying to whom



Privacy in PCNs

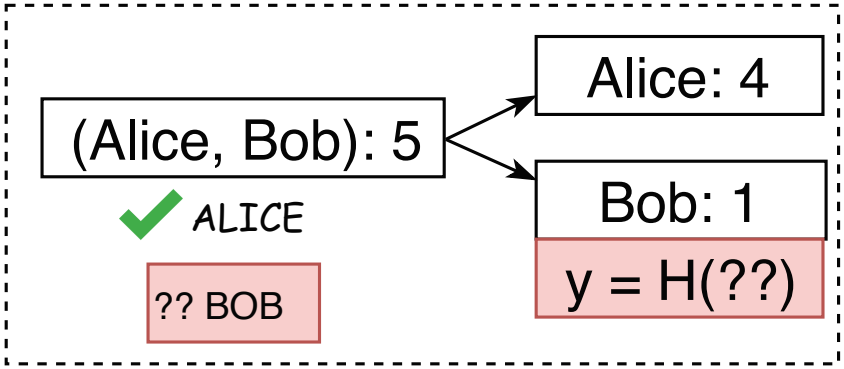
- Relationship Anonymity: The adversary cannot tell who is paying to whom



Problem: The same condition is used in the complete path!

Other Practical Considerations

- Scalability issues:
  - Two keys to define the deposit
  - Payment condition + signatures required
- Privacy issues:
  - Users sharing a channel revealed
- Interoperability
  - Support for specific hash function required



Summary Current PCN

	Current PCN
Security	
Privacy	
Interoperability / Compatibility	
Reduced Tx Size	

Wormhole Attack

Who pays to whom

Specific hash function

Two keys; HTLC script

What can we do with the signatures?

2-party ECDSA Signing [Lindell17]

(pk_A, sk_A)

Alice

(pk_B, sk_B)

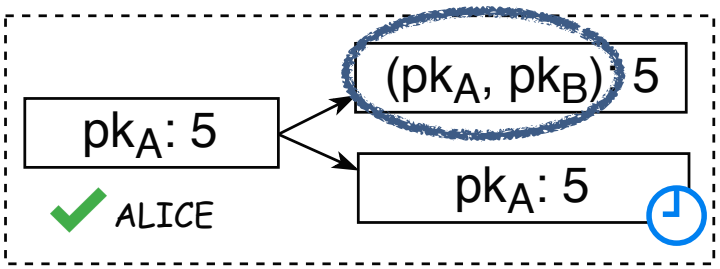
Bob

- Jointly compute a signature σ on a transaction
- It requires the knowledge of both sk_A and sk_B
- It can be publicly verified using $PK_{AB} := (sk_A * sk_B) * G$

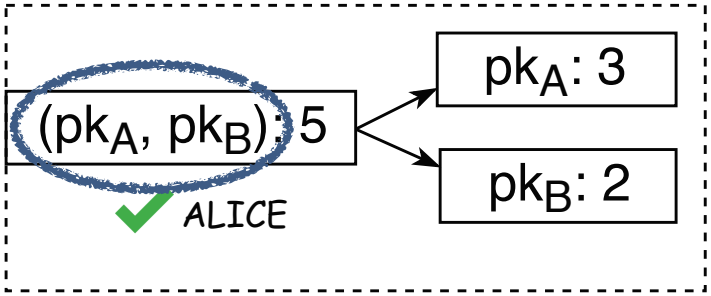
ECDSA: 2-party channel

Current

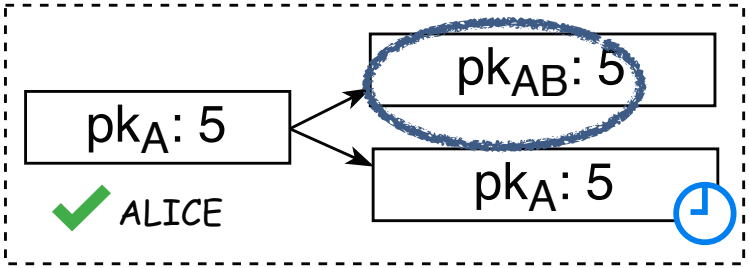
Open Channel

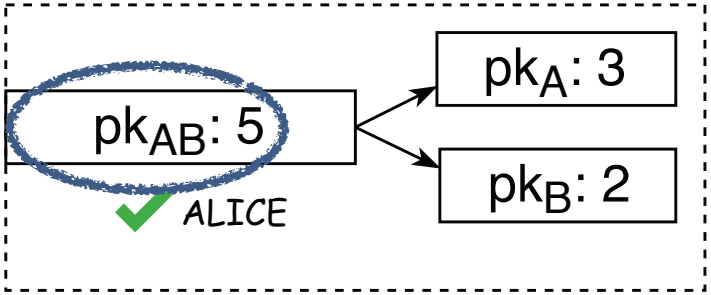


Off-chain Payment



SS-ECDSA





What if we encode the conditions in the signatures themselves?

Scriptless Scripts (Schnorr)

- Original idea proposed by Andrew Poelstra
- “Encode” payment condition within the Schnorr signatures
- In our work: formal description and analysis
- Unfortunately, Schnorr is not used in many cryptocurrencies today

Scriptless Scripts (SS-ECDSA)

- Was an open problem before our work
- Main challenge is the signature structure: No longer a linear combination
  - Schnorr signature: $(r_1 + r_2) + (k_1 + k_2)\, m$
  - ECDSA signature: $(r^{-1} * r^{-2})\, R_x\, (k_1 * k_2) + (r^{-1} * r^{-2})\, m$
    - Requires inverse, x coordinate of an elliptic curve point and multiplicative shares of the key $k = k_1 * k_2$
- In our work: formal description and analysis

2-party ECDSA Conditional Signing

(pk_A, sk_A)

Alice

Condition: (pk_C)

(pk_B, sk_B)

Bob

Goals:

- Alice can create a "half-signature" that Bob can finish only with sk_C
- If Bob creates a signature, Alice learns sk_C

2-party ECDSA Conditional Signing

(pk_A, sk_A)

Condition: (pk_C)

(pk_B, sk_B)

Alice

Bob

Create pk_{AB} and combine randomness $R := (pk_C, r_A, r_B)$

Send "1/3-signature" σ_B

Send "1/3-signature" σ_A

Send whole signature: $\sigma := \sigma_A * \sigma_B * \sigma_C$

Learn sk_C

Compute $\sigma_C := \sigma * (\sigma_B)^{-1} * (\sigma_C)^{-1}$

Retrieve sk_C from σ_C

2-party ECDSA Conditional Signing

(pk_A, sk_A)

Condition: (pk_C)

(pk_B, sk_B)

Alice

Bob

Create pk_{AB} and combine randomness $R := (pk_C, r_A, r_B)$

Send "1/3-signature" σ_B

Send "1/3-signature" σ_A

LOCK

Send whole signature: $\sigma := \sigma_A * \sigma_B * \sigma_C$

Learn sk_C

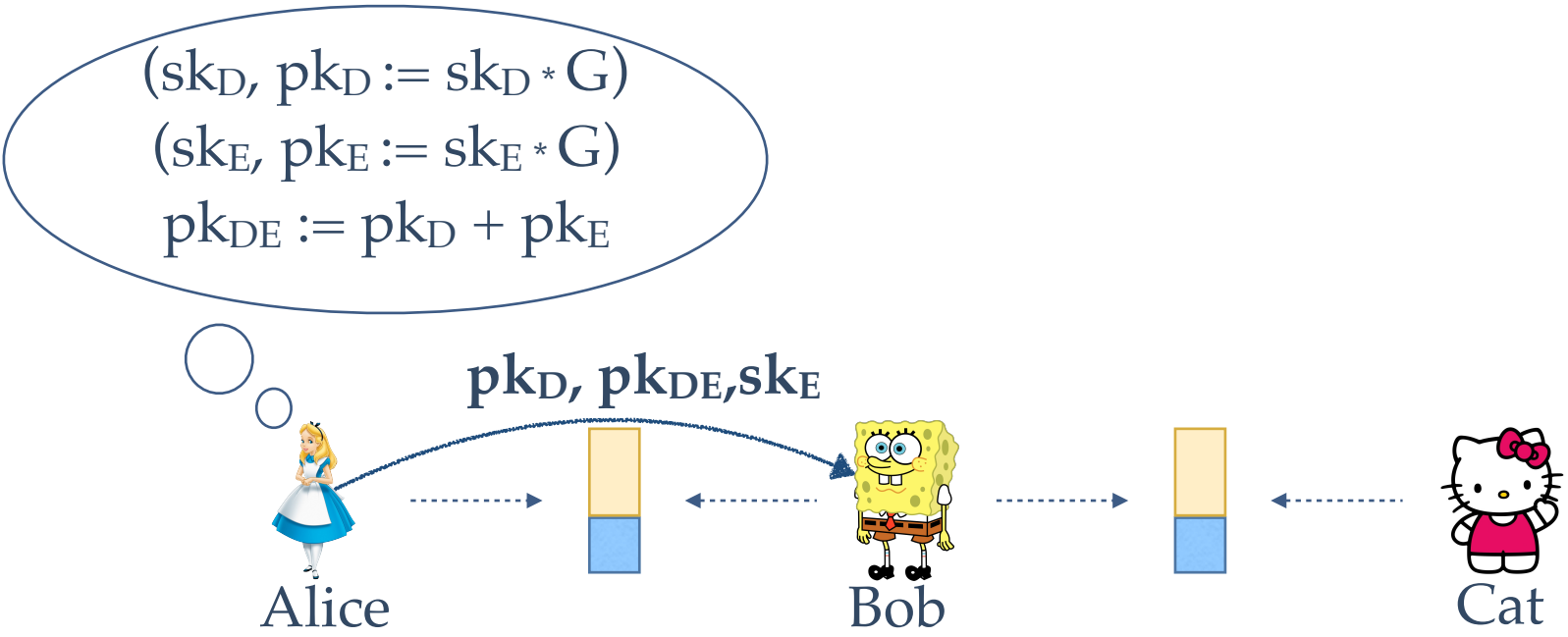
Compute $\sigma_C := \sigma * (\sigma_B)^{-1} * (\sigma_C)^{-1}$

Retrieve sk_C from σ_C

RELEASE

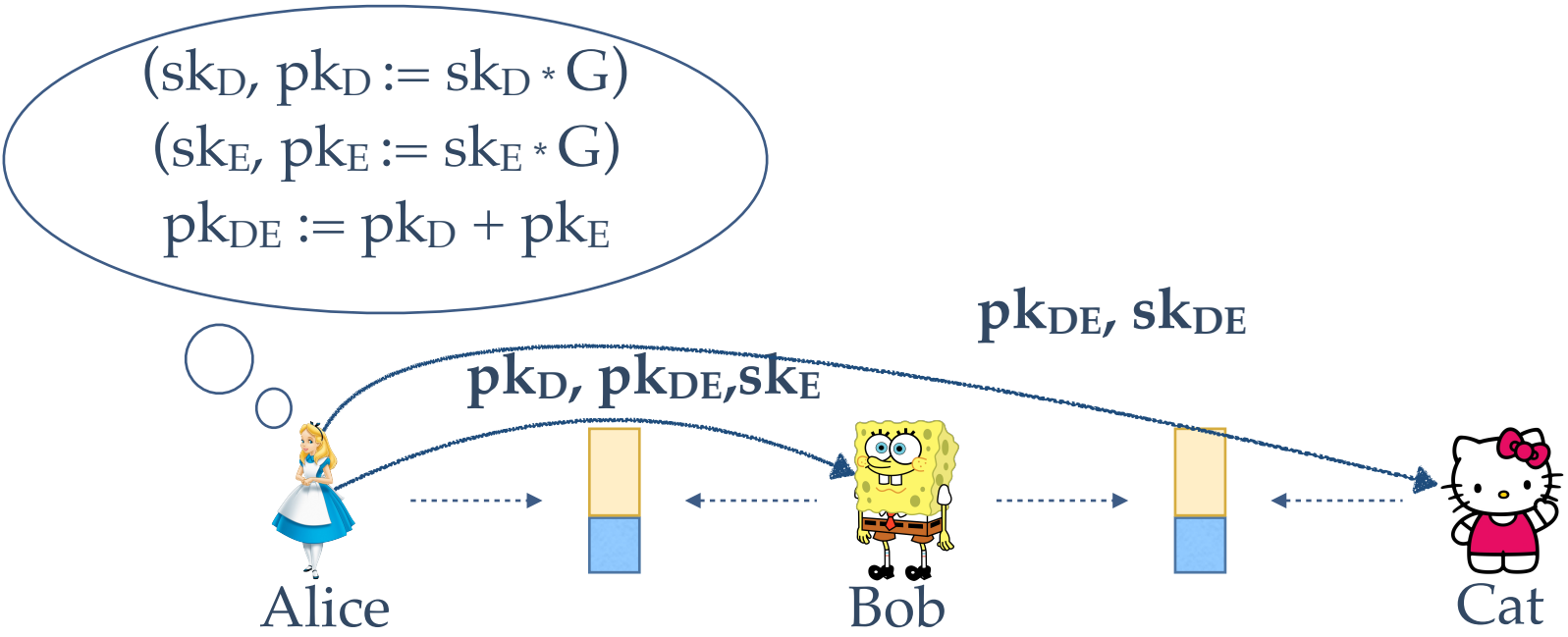
ECDSA-based PCN: Setup

- Multiple "chained" ECDSA conditional payments allow multi-hop payments in the presence of malicious intermediaries



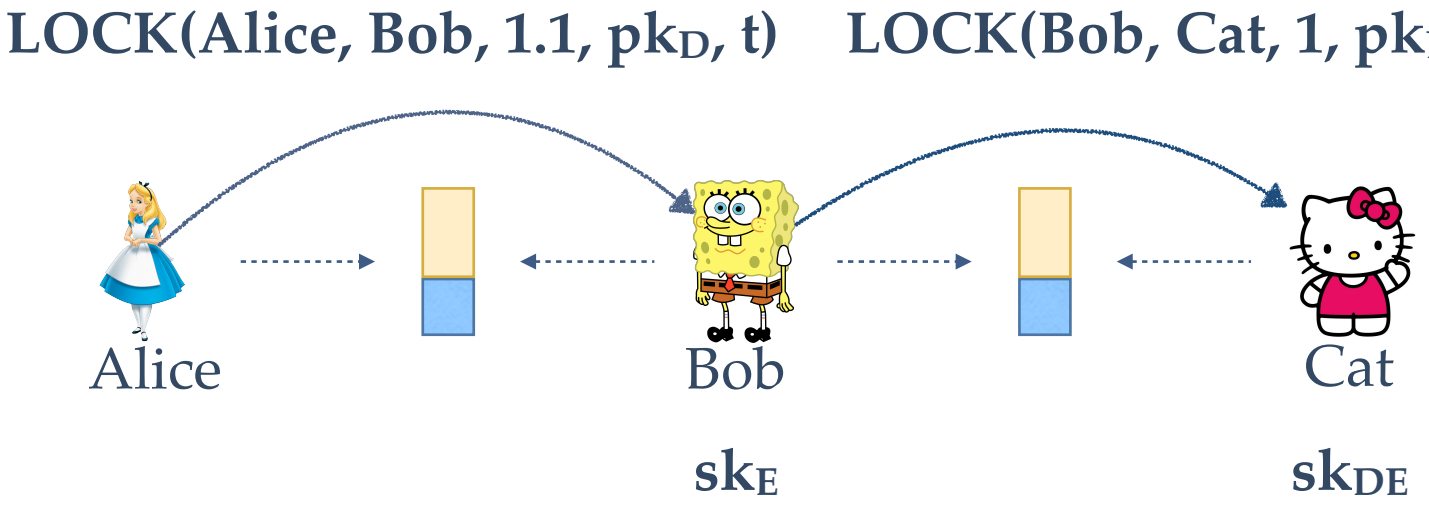
ECDSA-based PCN: Setup

- Multiple "chained" ECDSA conditional payments allow multi-hop payments in the presence of malicious intermediaries



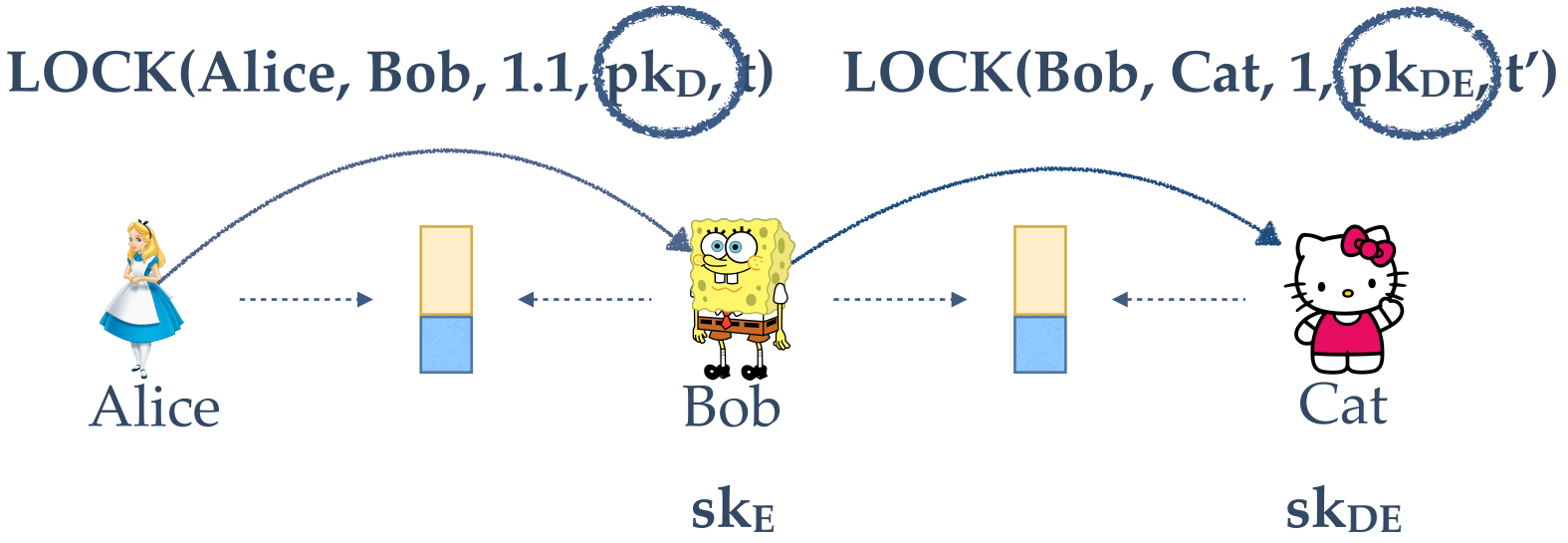
ECDSA-based PCN: Lock

- Multiple "chained" ECDSA conditional payments allow multi-hop payments in the presence of malicious intermediaries



ECDSA-based PCN: Lock

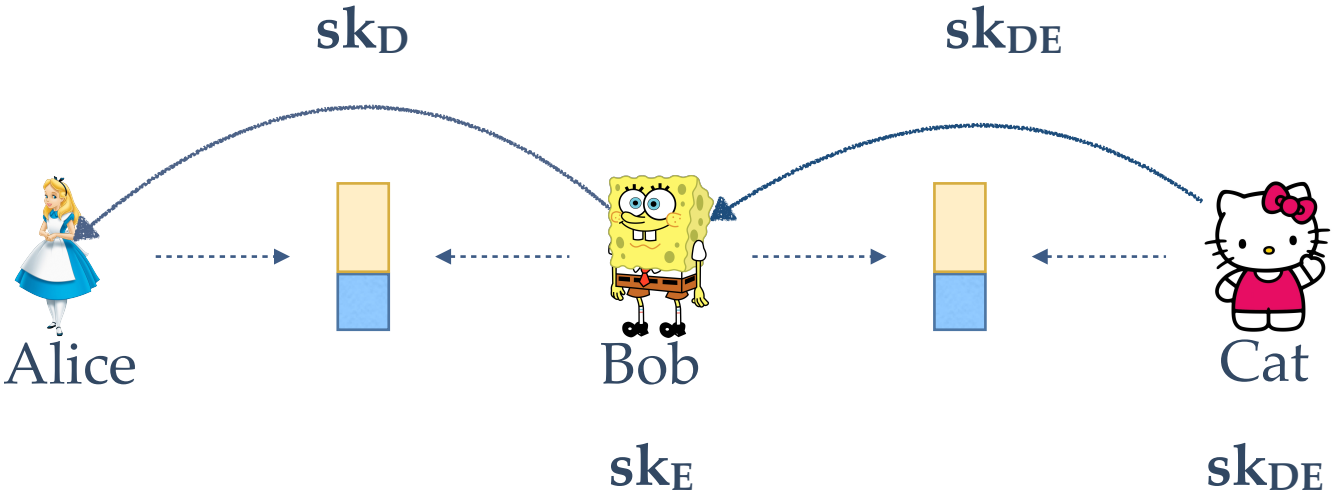
- Multiple "chained" ECDSA conditional payments allow multi-hop payments in the presence of malicious intermediaries



Randomized conditions in the path: Security and Privacy

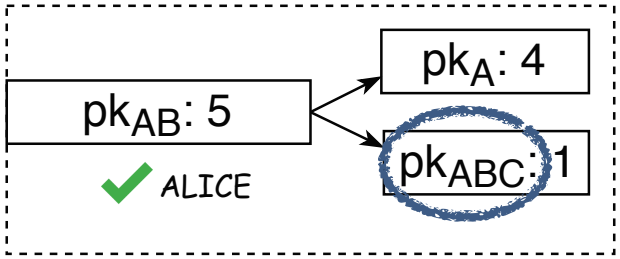
ECDSA-based PCN: Release

- Multiple "chained" ECDSA conditional payments allow multi-hop payments in the presence of malicious intermediaries



ECDSA-based PCN: Discussion

- It can be extended to arbitrary number of hops
- It reduces transaction size for conditional payments



- Evaluation: <500 bytes communication. Few ms computation
- Improve interoperability. Useful for other applications (e.g., atomic swaps and cross-chain payments)
- Compatible with Bitcoin

Summary Current ECDSA

	Current PCN	ECDSA-based PCN
Security		
Privacy		
Interoperability / Compatibility		
Reduced Tx Size		

One secret per user

Randomized conditions

Only ECDSA required

Condition in one key

Summary

- More in the paper:
  - One-way homomorphic functions suffice for multi-hop locks in full script setting
  - Possible to combine OWH-Schnorr-ECDSA locks in the same path
  - Security and privacy modelled and proven in the Universal Composability Framework —> Composability guarantees
- Multi-hop locks implemented in the Lightning Network
- It enables a plethora of applications (e.g., atomic swaps and cross-chain payments)