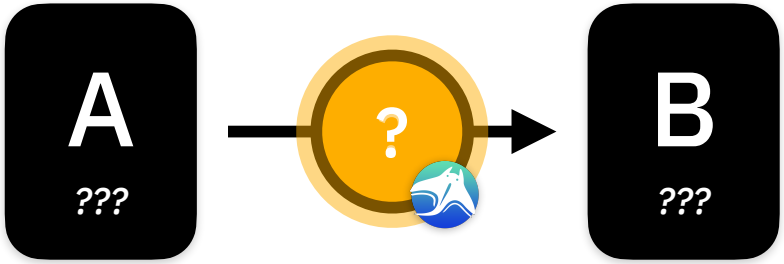
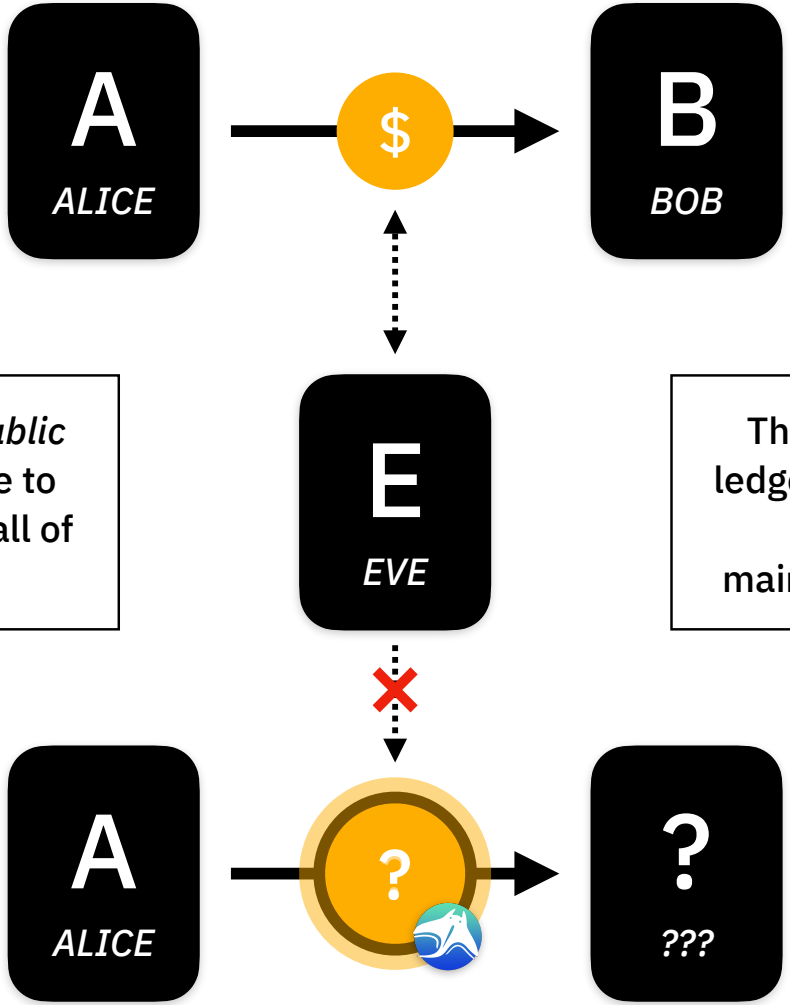
The MantaPay Protocol

Brandon H. Gomes





EVE is spying on ALICE. Using a *public ledger protocol*, EVE would be able to learn the amount and recipient of all of ALICE's transactions.

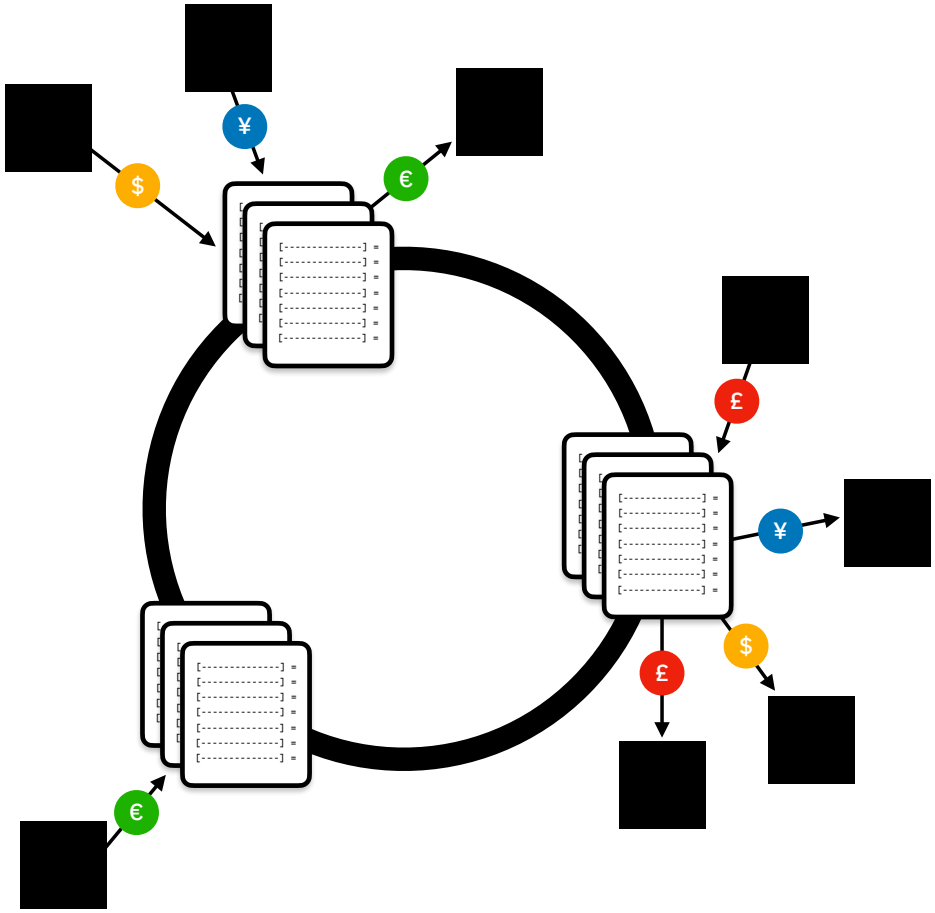
The goal is to find a way to design a ledger so that ALICE's transactions can be hidden from EVE, while still maintaining the integrity of the ledger.

Properties of a Decentralized Anonymous Payment Protocol

We would like to have the following two properties hold true of whatever ledger design we decide to use for private payments:

1. A transaction cannot modify the total supply of an asset, only the existing ownership, i.e. the right to spend the asset in the future
2. Reading the ledger cannot reveal any information about a transaction or existing user balances

These two properties seem to be at odds with each other since we want to verify that the total supply remains fixed but we must do so without learning what the underlying transaction is! This is where some clever cryptography can save the day.

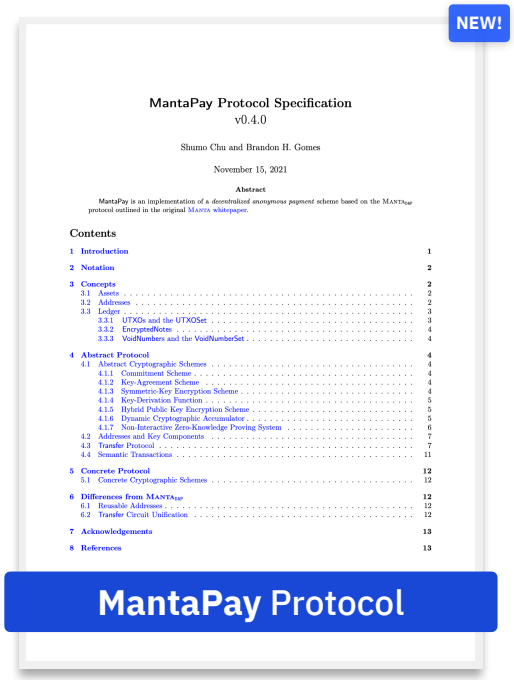


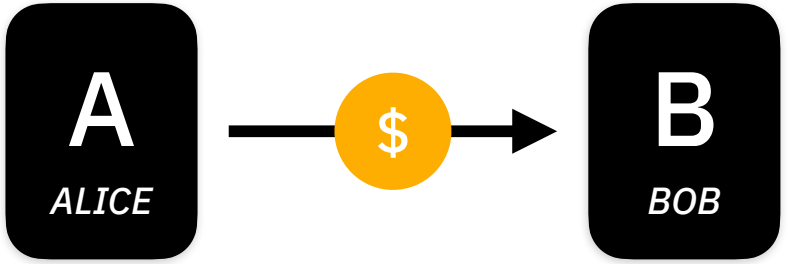
MantaPay

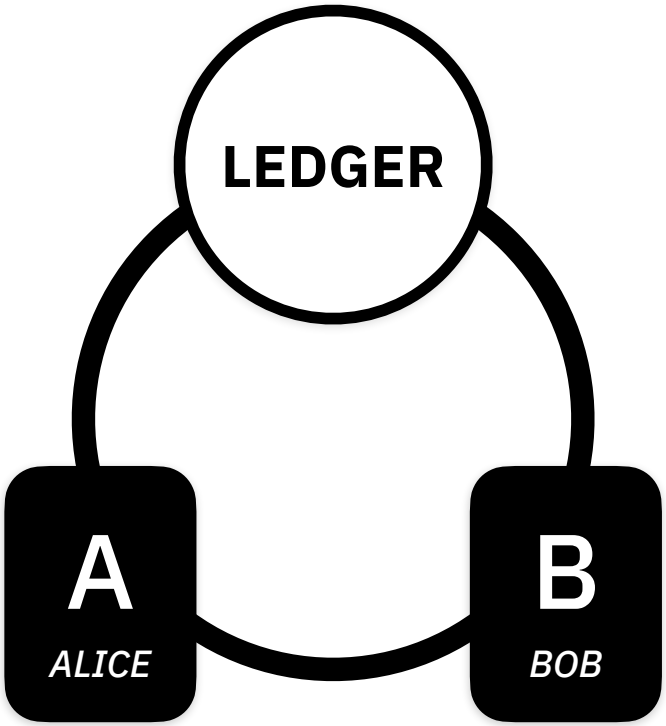
To describe the MantaPay protocol, we will follow one particular transaction from ALICE to BOB and go through the different steps involved in updating the ledger. We will follow this transaction by viewing it from different levels of abstraction:

1. Send / Receive
2. Encrypt / Decrypt
3. Shared Secrets, UTXOs, and Void Numbers
4. Ledger

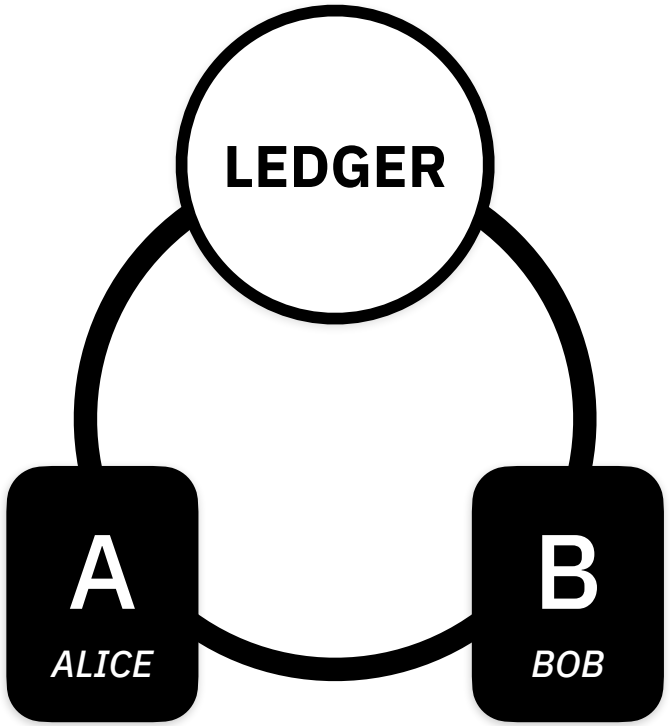
Then, we will generalize the discussion to multiple different senders and receivers and summarize the complete protocol.



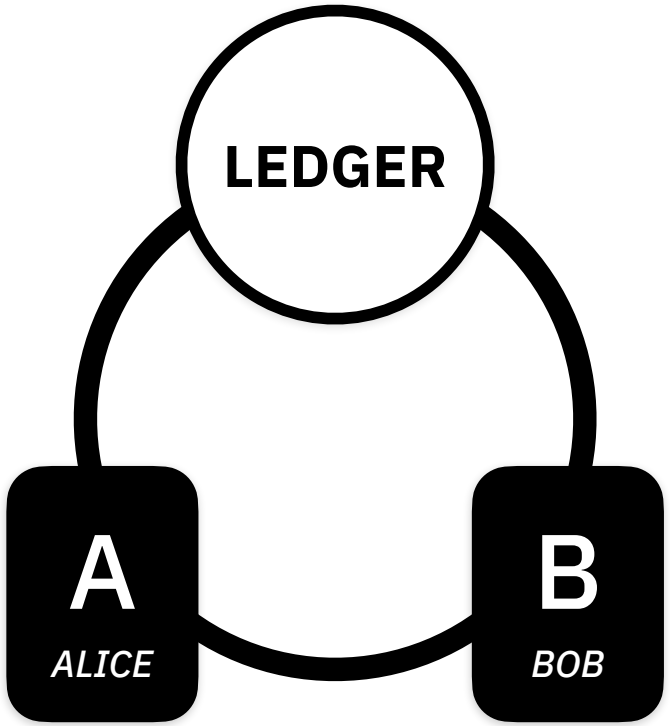



LEDGER
A
ALICE
B
BOB

ALICE represents the *sender*: a participant that already has access to some assets and is guaranteed by the LEDGER the ability to spend them.

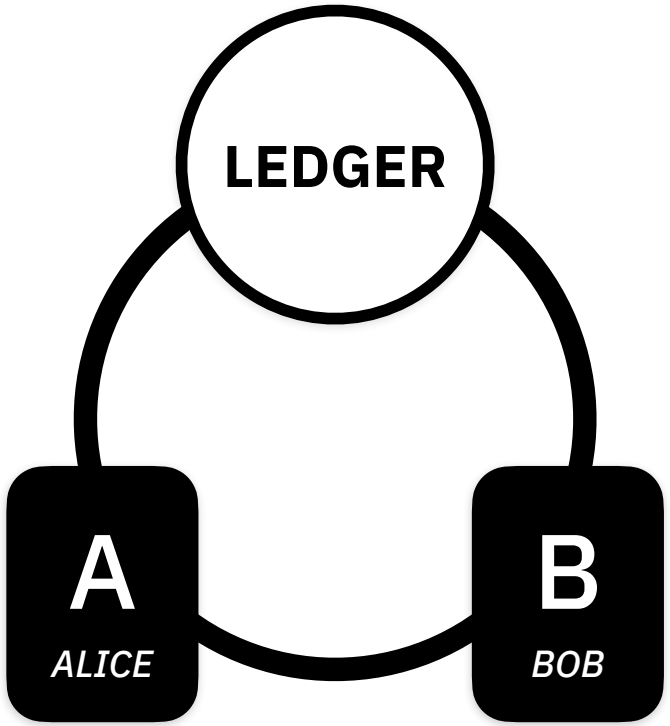


ALICE represents the *sender*: a participant that already has access to some assets and is guaranteed by the LEDGER the ability to spend them.



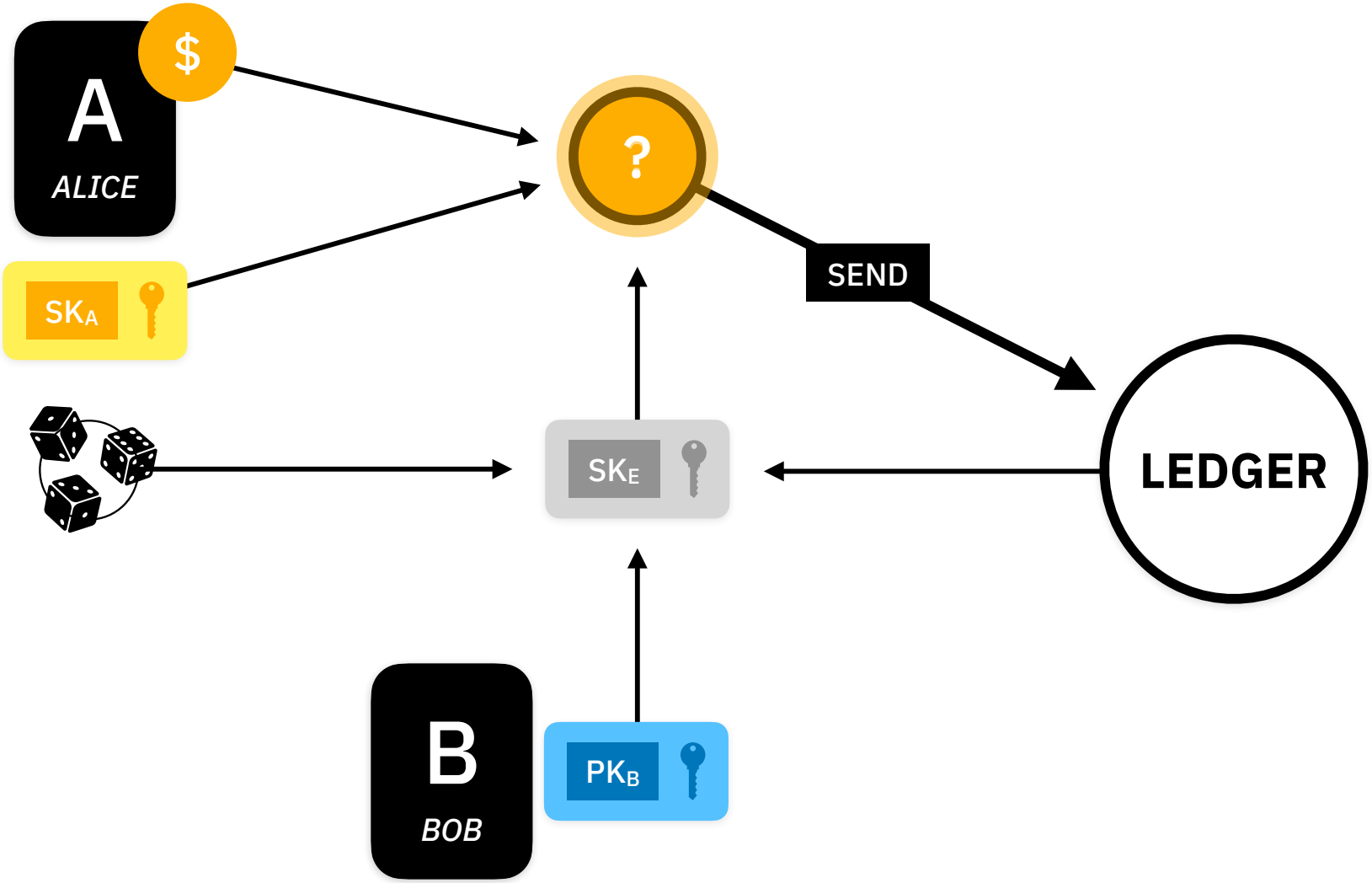
BOB represents the *receiver*: a participant that can be identified as the sole recipient of an asset which will be guaranteed the ability to spend it in the future.

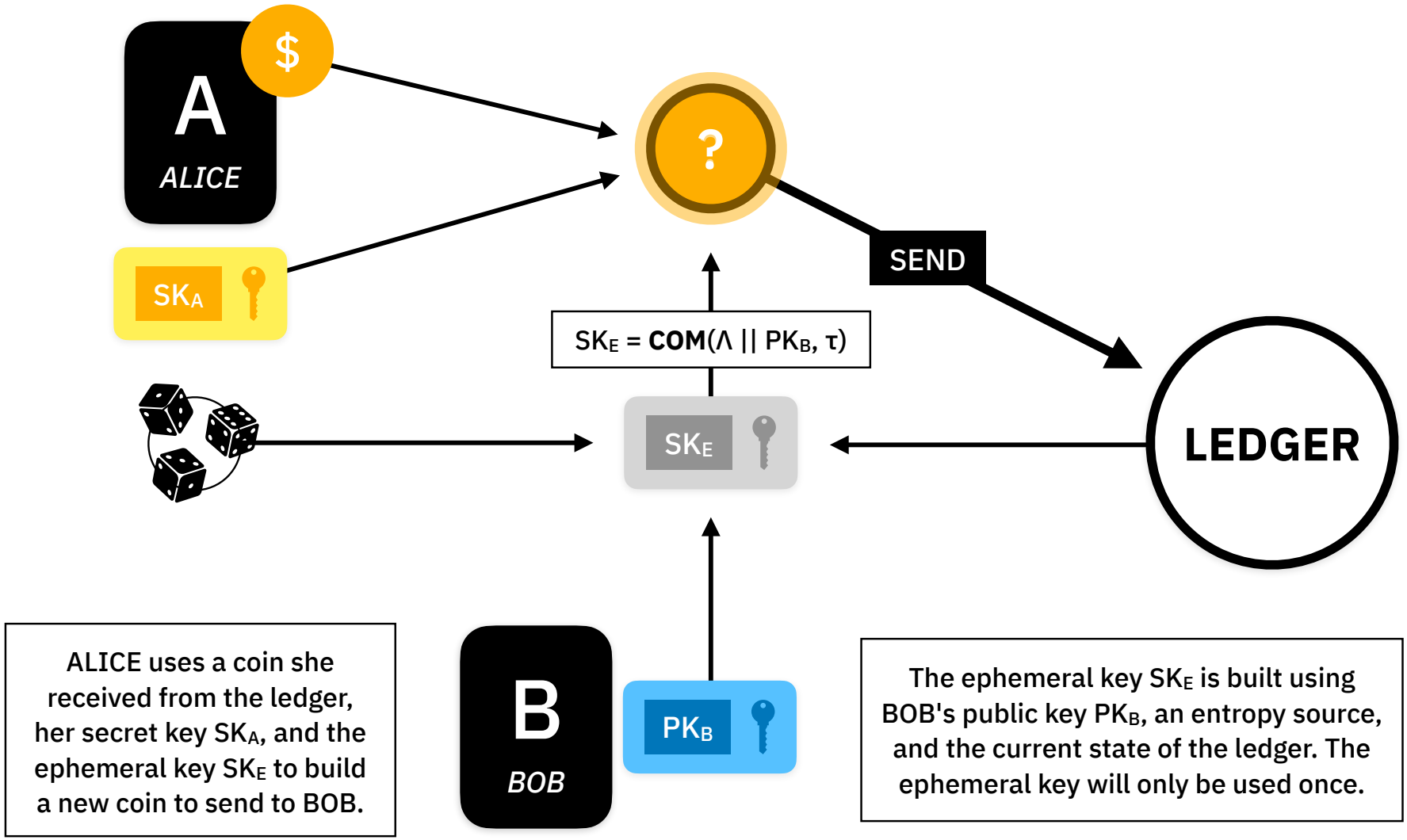
The LEDGER represents all public information and the validation of the transfer of ownership between *senders* and *receivers*.

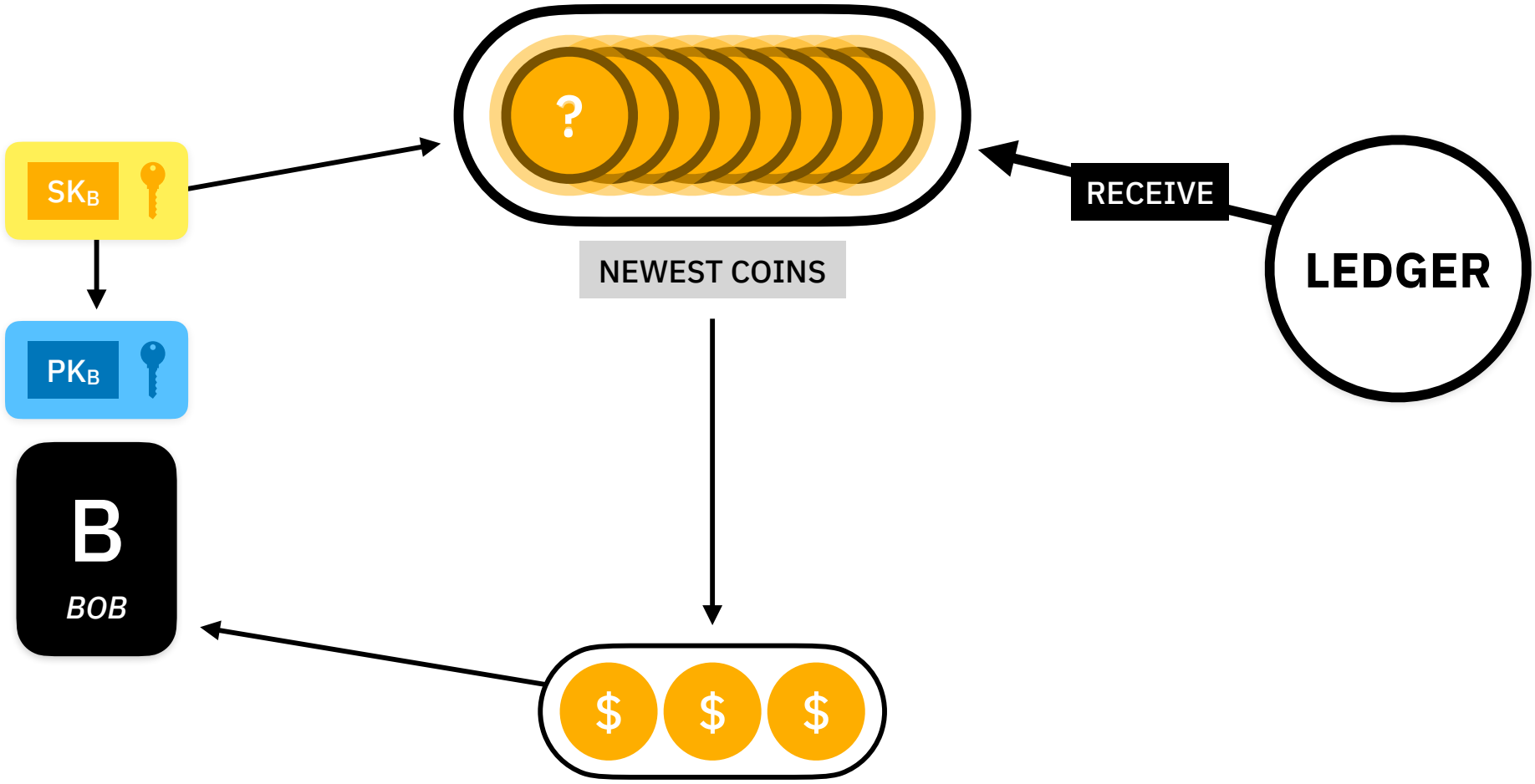


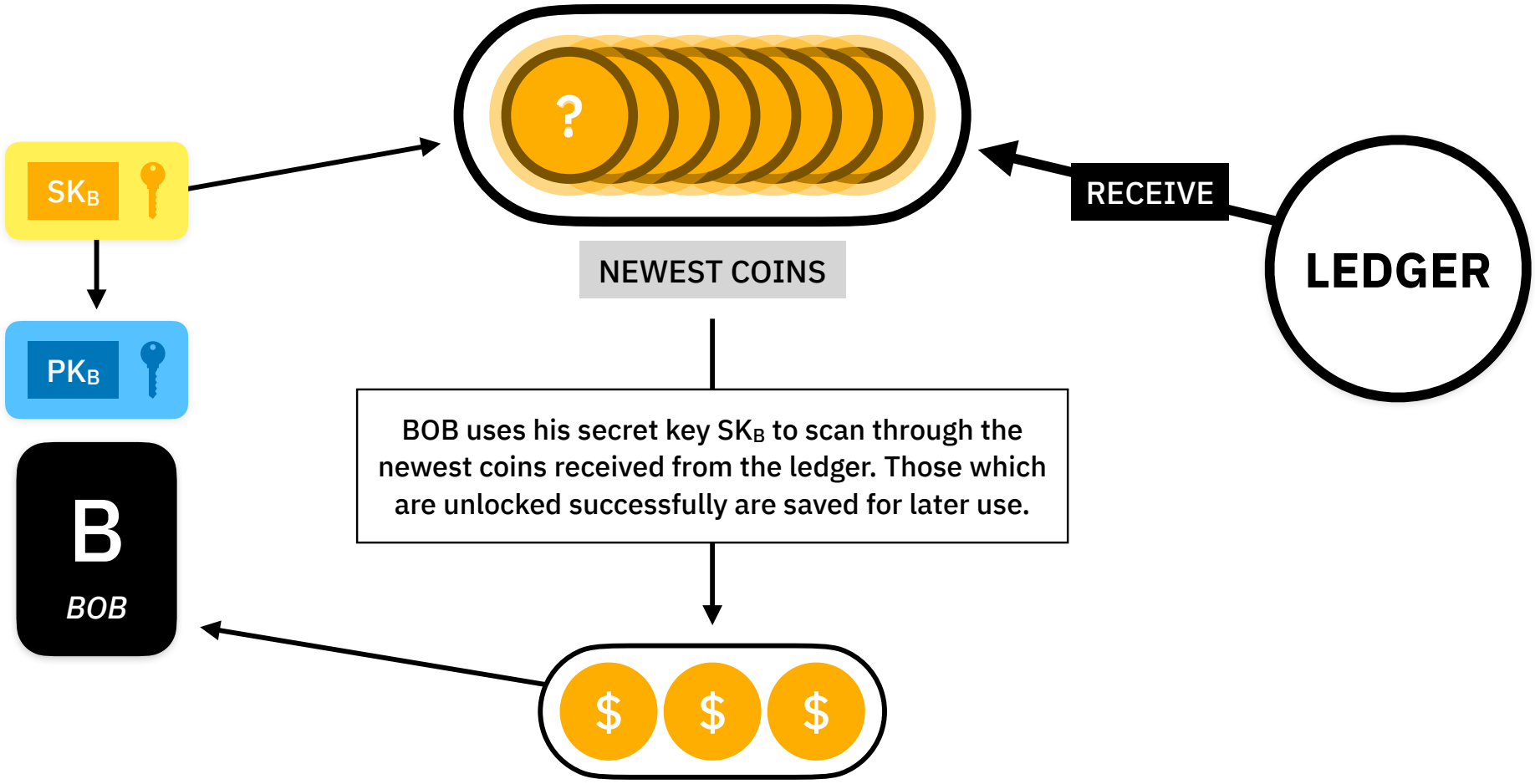
ALICE represents the *sender*: a participant that already has access to some assets and is guaranteed by the LEDGER the ability to spend them.

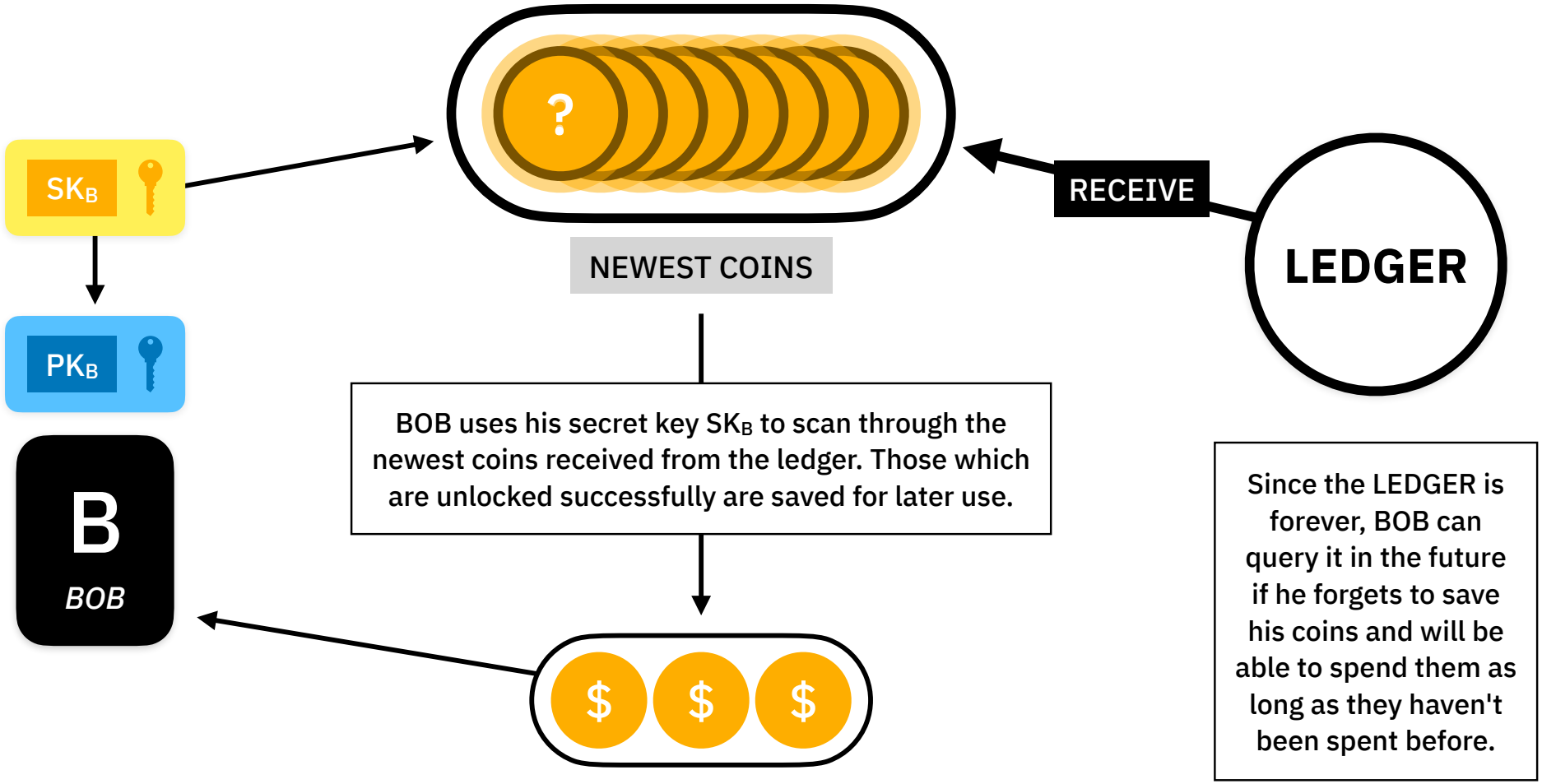
BOB represents the *receiver*: a participant that can be identified as the sole recipient of an asset which will be guaranteed the ability to spend it in the future.

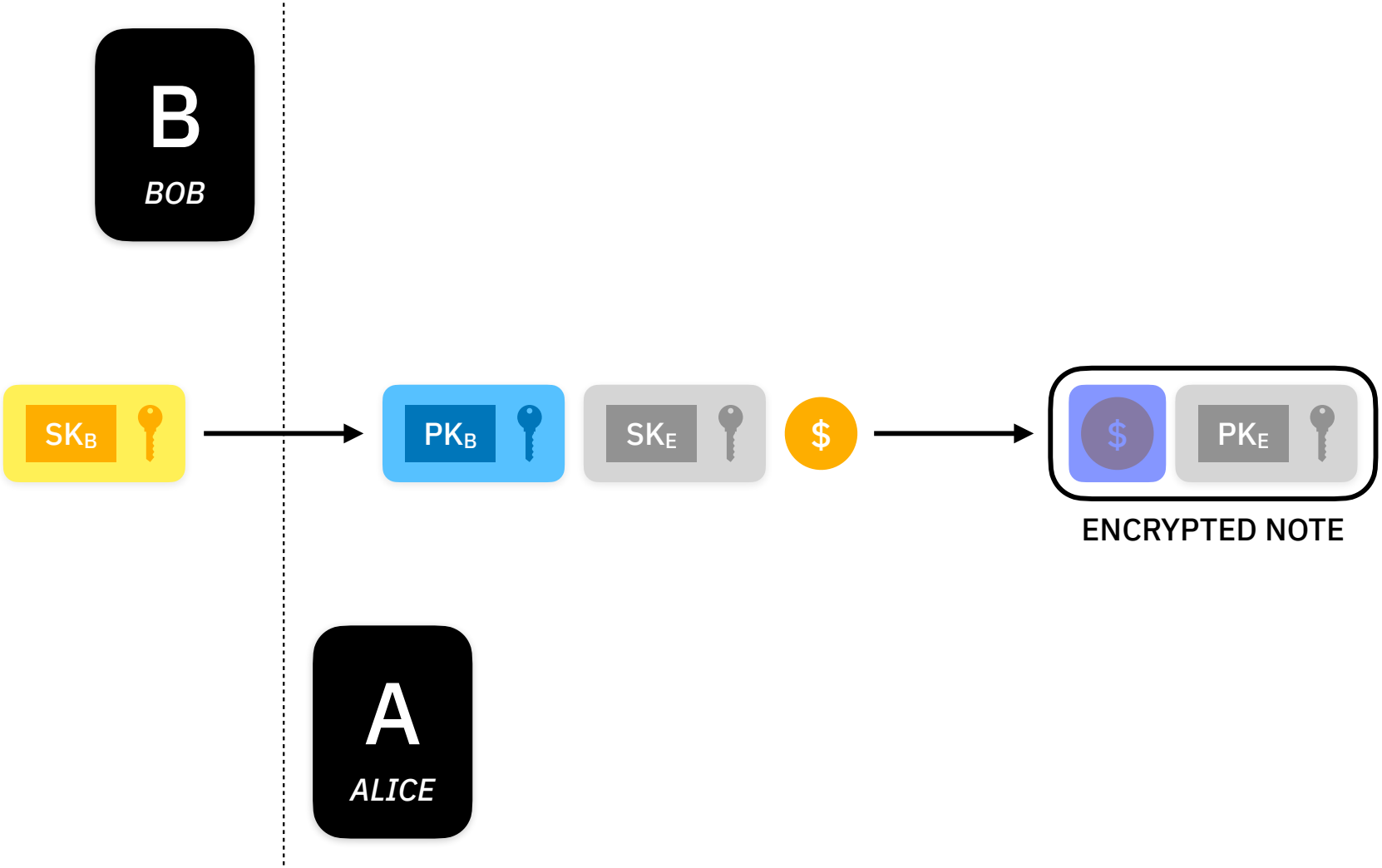

A
ALICE
$
SK_A
?
SEND
LEDGER
SK_E
B
BOB
PK_B

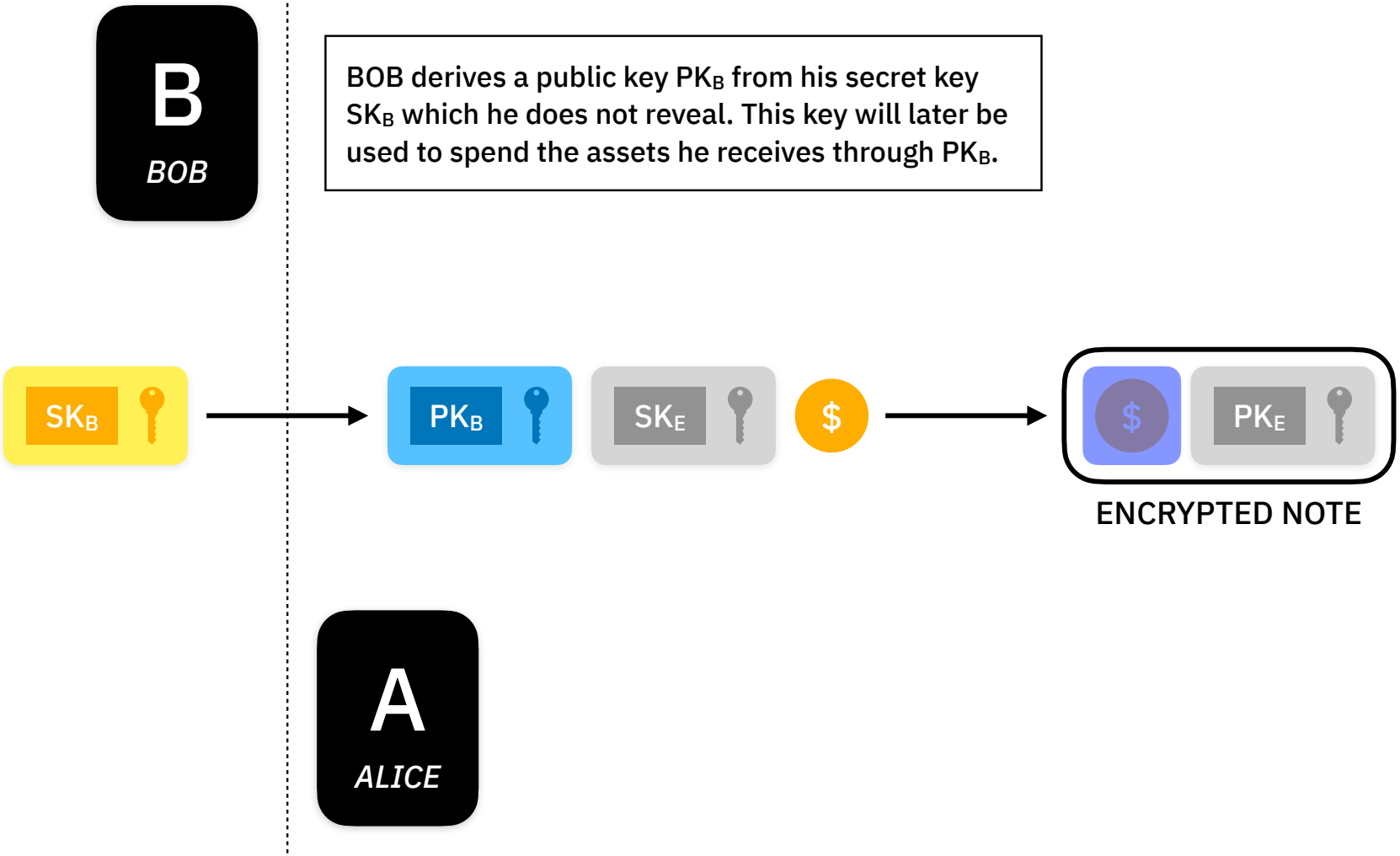

A
ALICE
$
?
SEND
SK_A
SK_E = COM(Λ || PK_B, τ)
SK_E
LEDGER
B
BOB
PK_B
ALICE uses a coin she received from the ledger, her secret key SK_A, and the ephemeral key SK_E to build a new coin to send to BOB.
The ephemeral key SK_E is built using BOB's public key PK_B, an entropy source, and the current state of the ledger. The ephemeral key will only be used once.

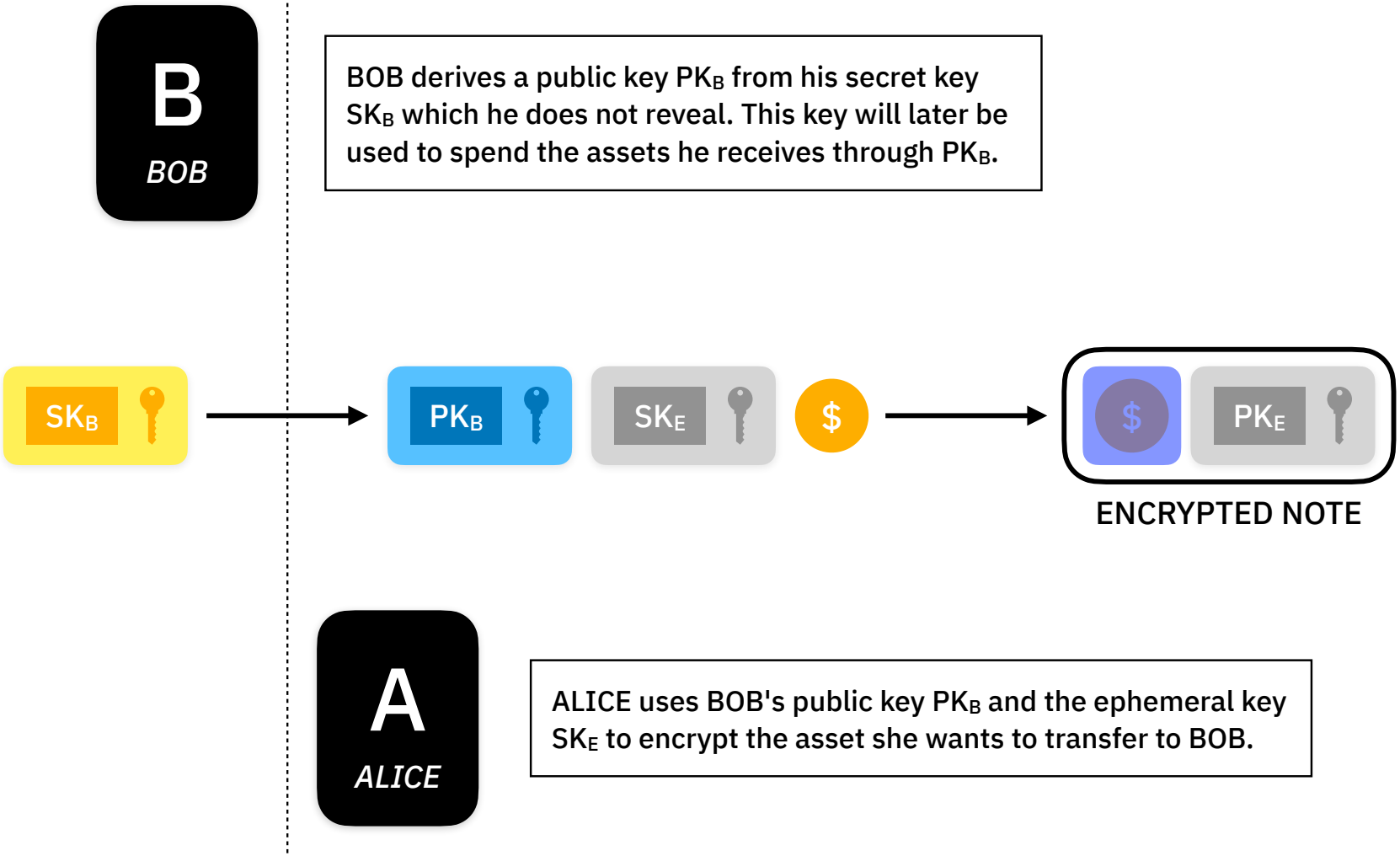

?
SK_B
PK_B
B
BOB
NEWEST COINS
RECEIVE
LEDGER
$
$
$

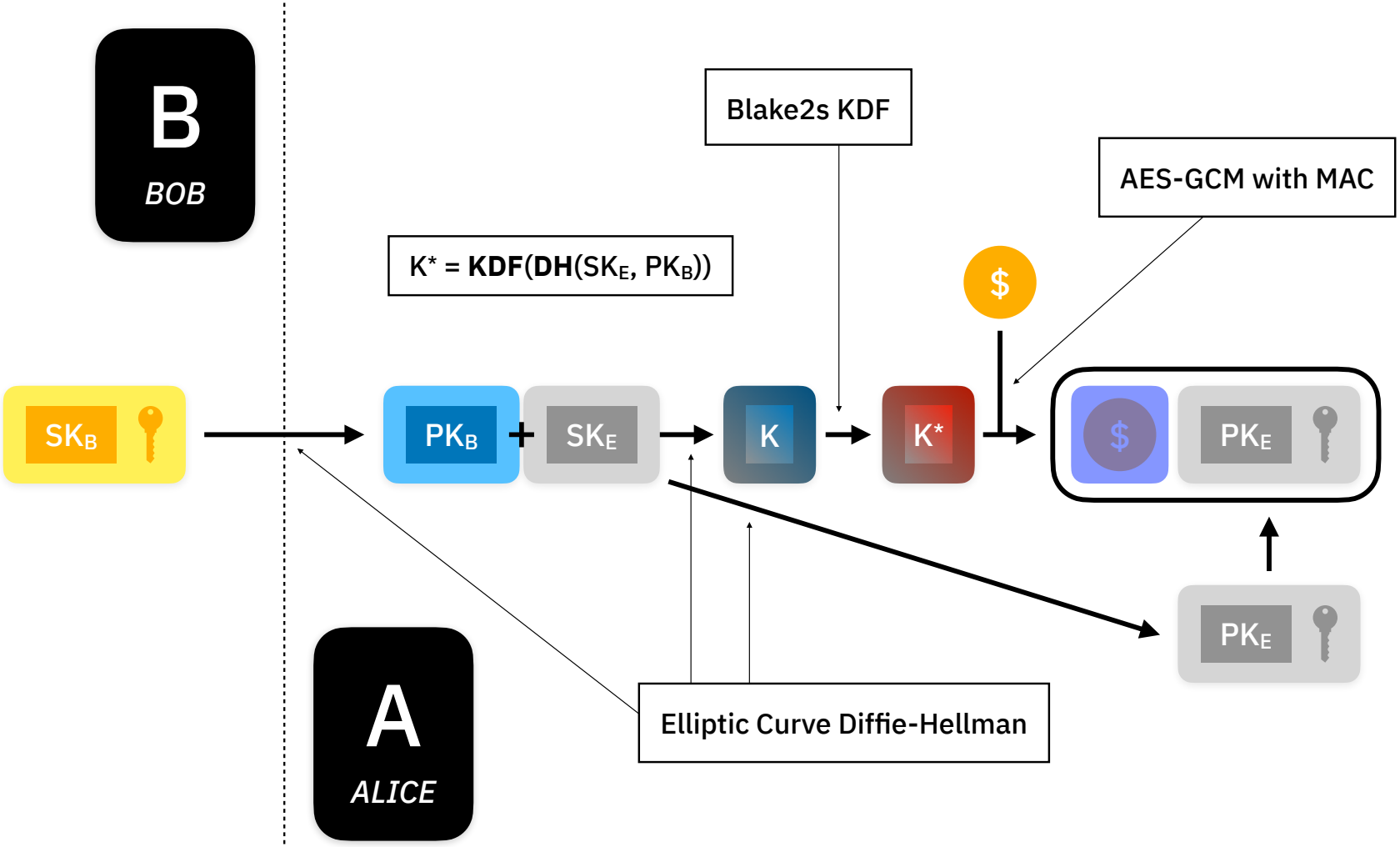

?
SK_B
PK_B
B
BOB
RECEIVE
LEDGER
NEWEST COINS
BOB uses his secret key SK_B to scan through the newest coins received from the ledger. Those which are unlocked successfully are saved for later use.
$
$
$

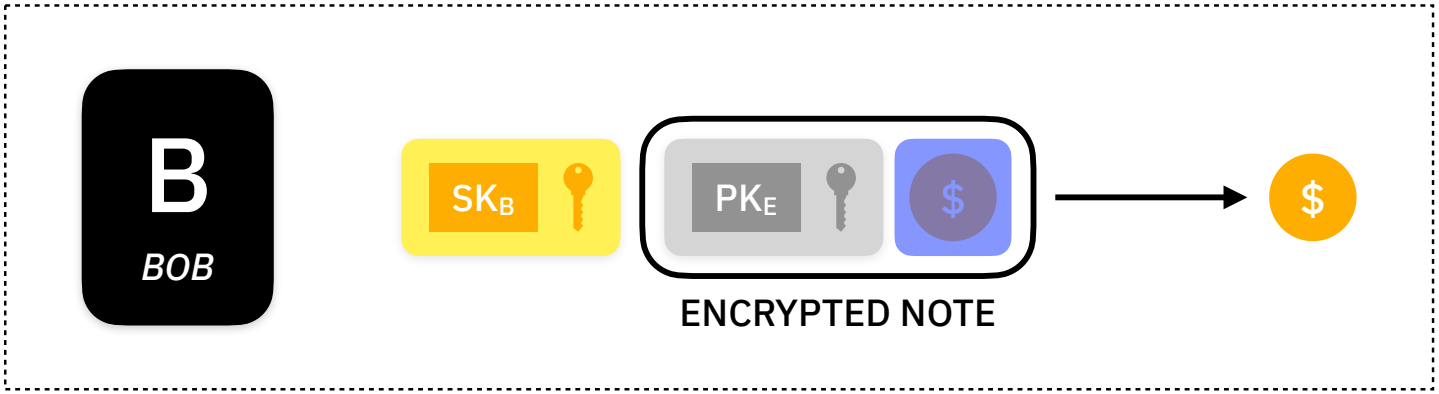

?
RECEIVE
LEDGER
SK_B
PK_B
B
BOB
NEWEST COINS
BOB uses his secret key SK_B to scan through the newest coins received from the ledger. Those which are unlocked successfully are saved for later use.
$
$
$
Since the LEDGER is forever, BOB can query it in the future if he forgets to save his coins and will be able to spend them as long as they haven't been spent before.

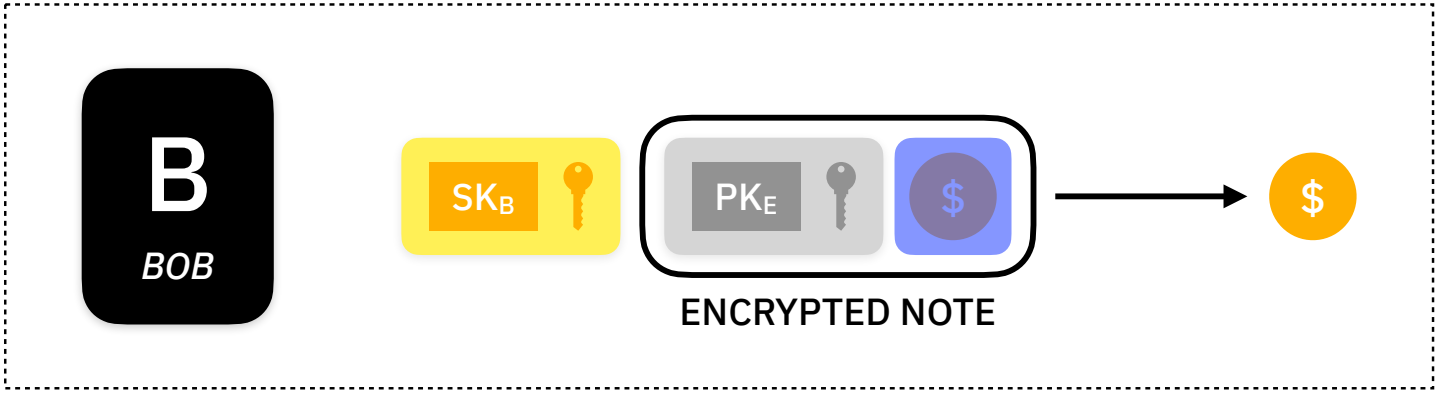

B
BOB
SKB
PKB
SKE
$
$
PKE
ENCRYPTED NOTE
A
ALICE


B
BOB
BOB derives a public key PK_B from his secret key SK_B which he does not reveal. This key will later be used to spend the assets he receives through PK_B.
SK_B
PK_B
SK_E
$
$
PK_E
ENCRYPTED NOTE
A
ALICE

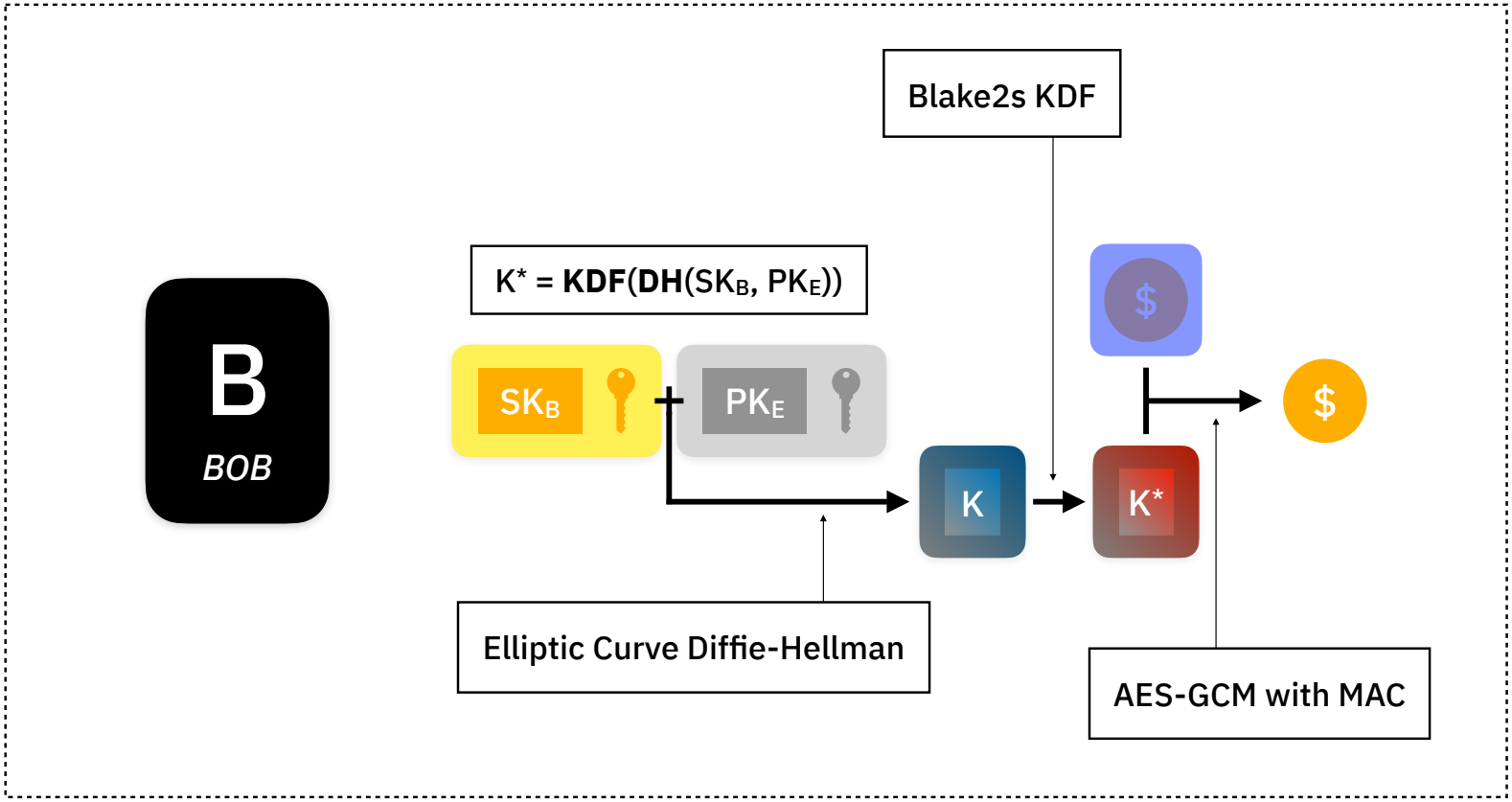

B
BOB
BOB derives a public key PK_B from his secret key SK_B which he does not reveal. This key will later be used to spend the assets he receives through PK_B.
SK_B
PK_B
SK_E
$
$
PK_E
ENCRYPTED NOTE
A
ALICE
ALICE uses BOB's public key PK_B and the ephemeral key SK_E to encrypt the asset she wants to transfer to BOB.


B
BOB
Blake2s KDF
AES-GCM with MAC
K* = KDF(DH(SKE, PKB))
$
SKB
PKB
SKE
K
K*
$
PKE
PKE
A
ALICE
Elliptic Curve Diffie-Hellman

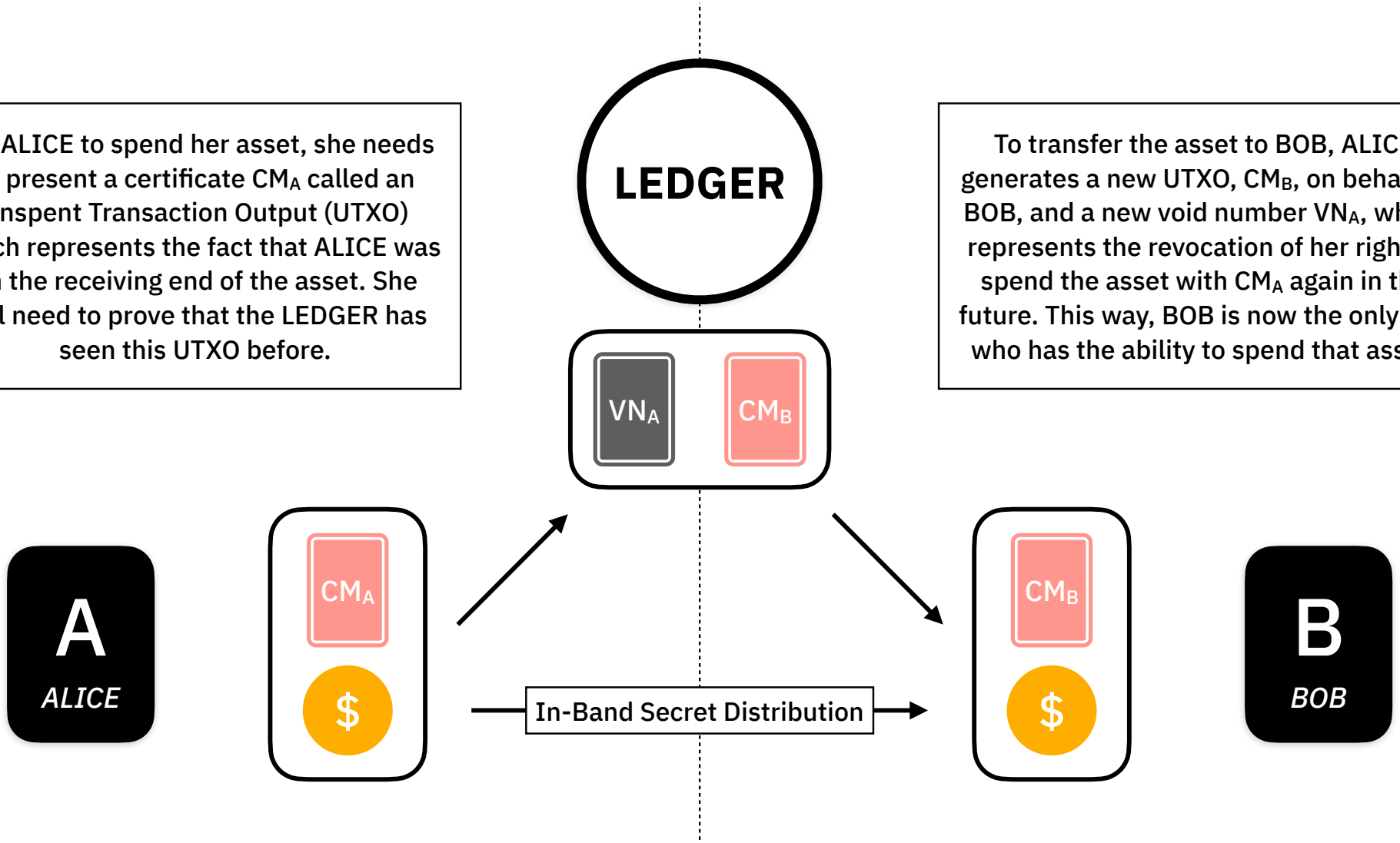

B
BOB
SK_B
PK_E
$
$
ENCRYPTED NOTE



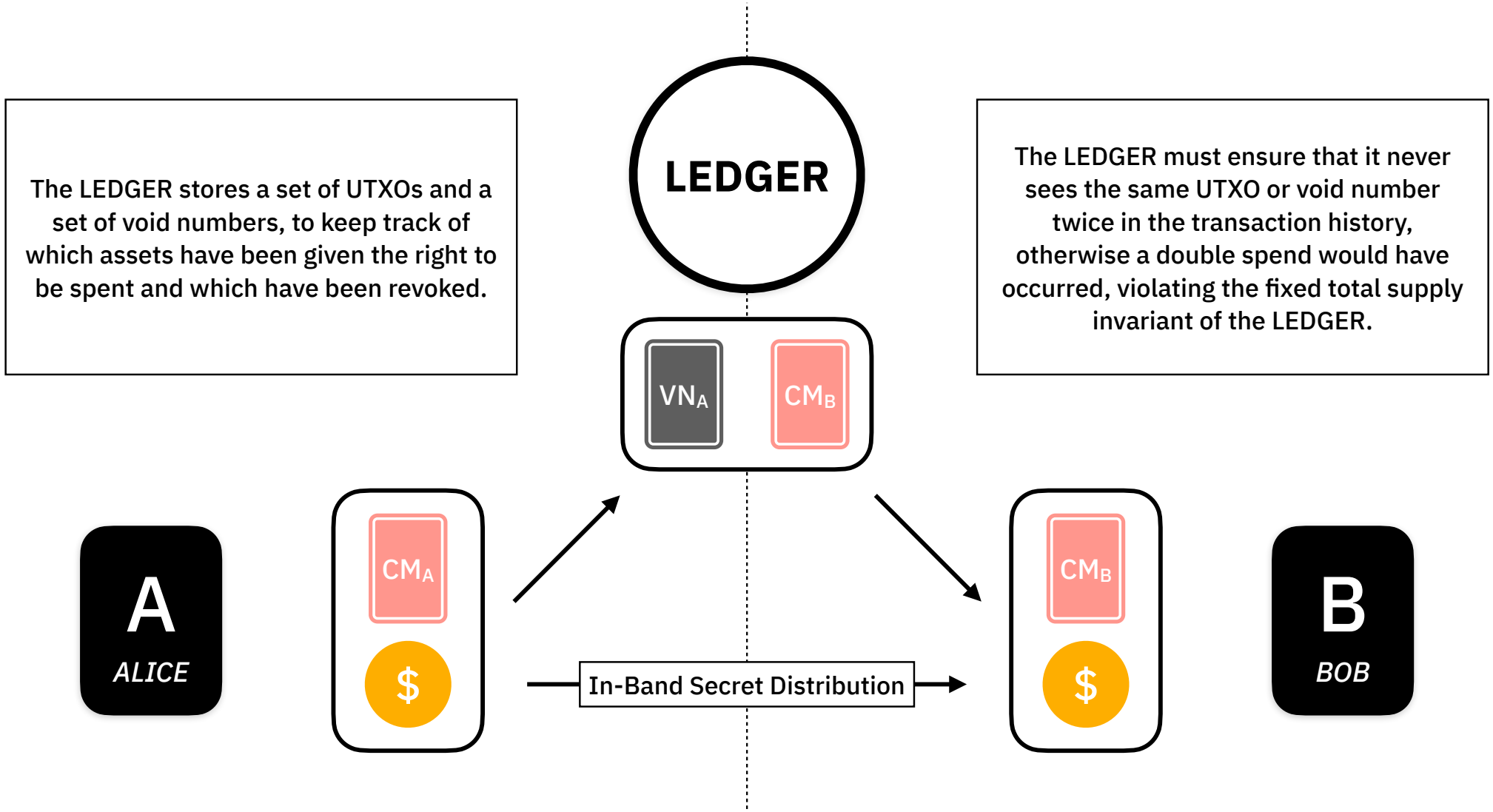
When BOB receives assets from the ledger, he attempts to decrypt the ENCRYPTED NOTEs using his secret key SK_B. If the decryption succeeds, then he uses the ephemeral key PK_E and his secret key SK_B to spend the asset.

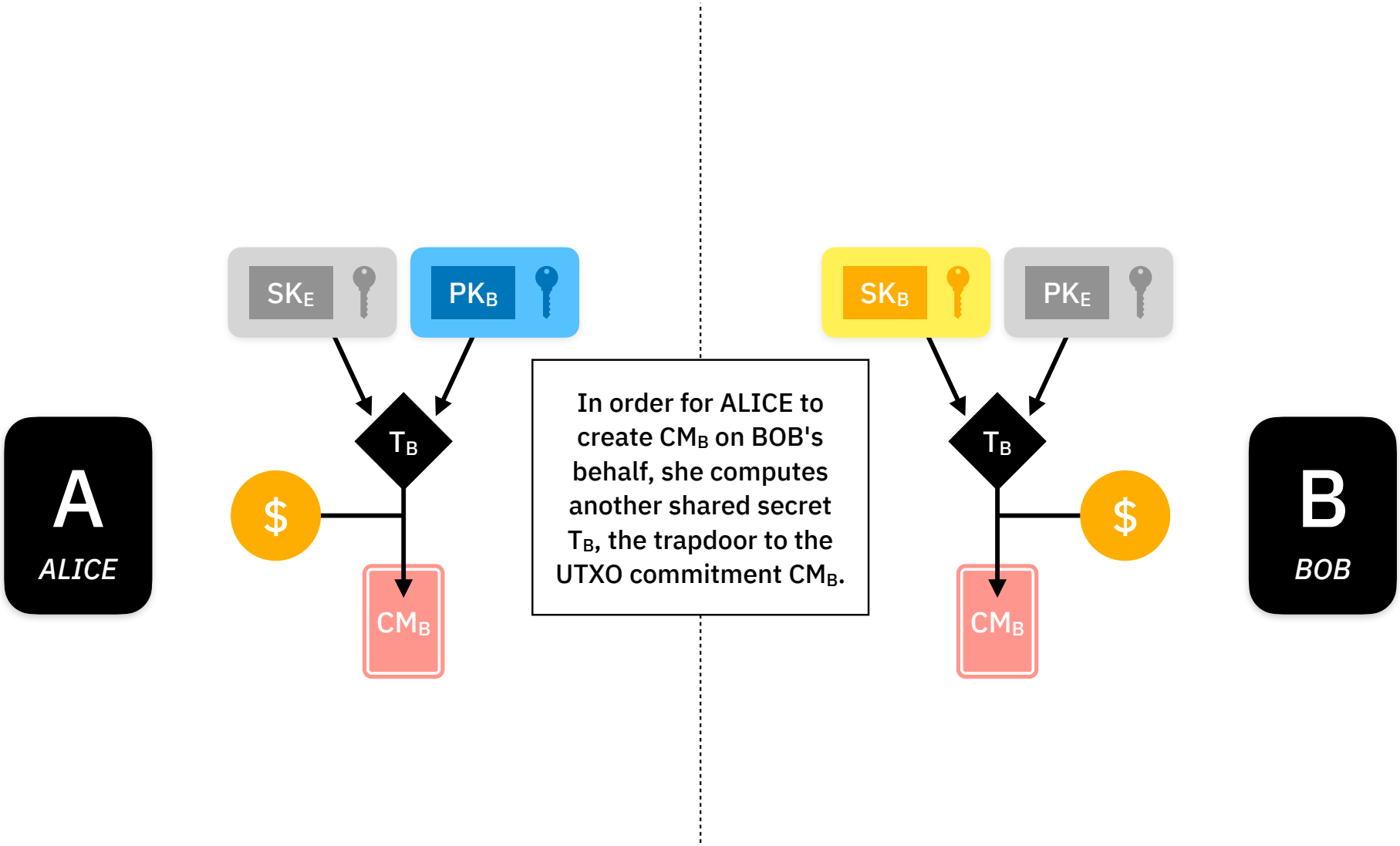

Blake2s KDF
K* = KDF(DH(SKB, PKE))
B
BOB
SKB
PKE
$
$
K
K*
Elliptic Curve Diffie-Hellman
AES-GCM with MAC

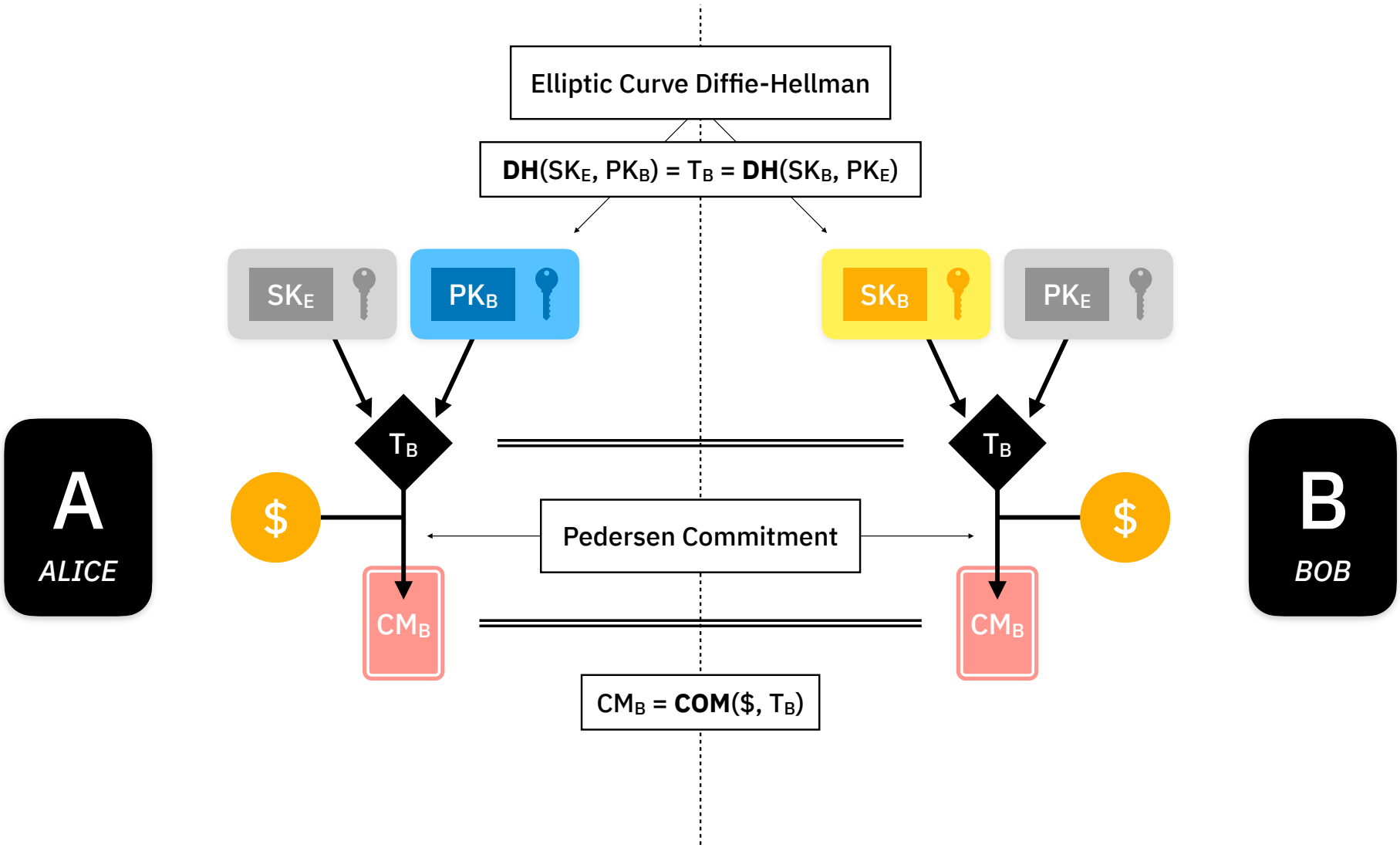
For ALICE to spend her asset, she needs to present a certificate CM_A called an Unspent Transaction Output (UTXO) which represents the fact that ALICE was on the receiving end of the asset. She will need to prove that the LEDGER has seen this UTXO before.

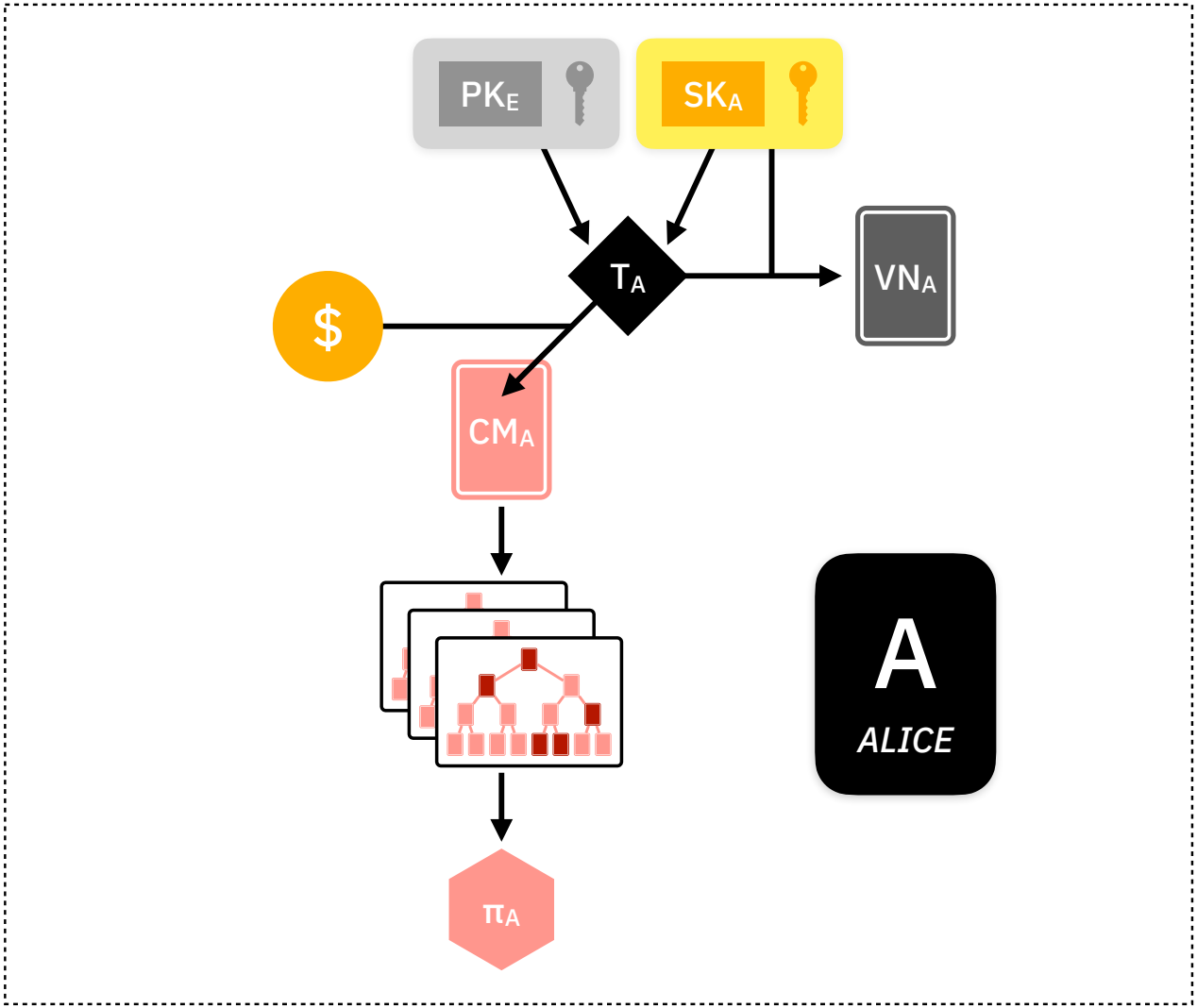


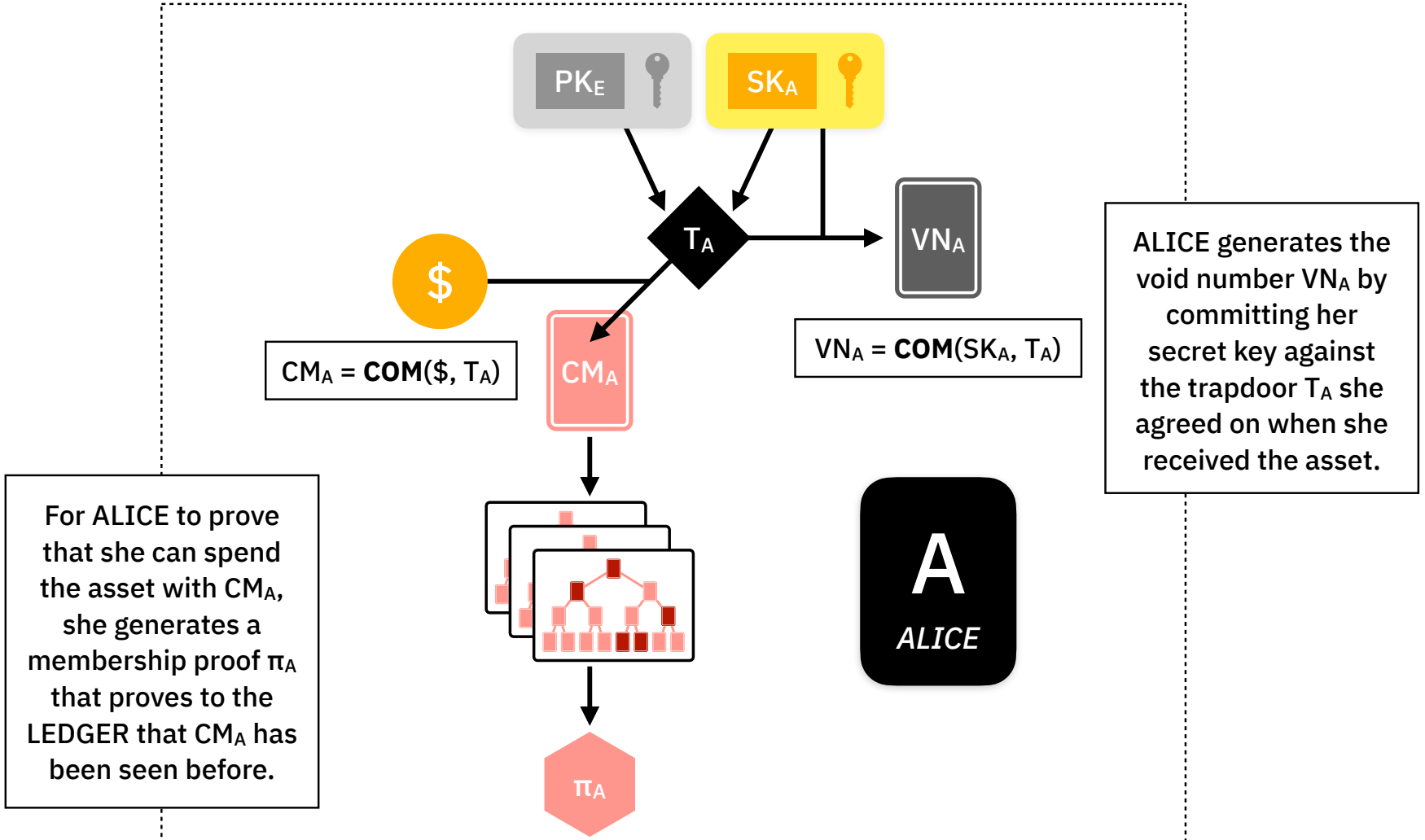
To transfer the asset to BOB, ALICE generates a new UTXO, CM_B, on behalf of BOB, and a new void number VN_A, which represents the revocation of her right to spend the asset with CM_A again in the future. This way, BOB is now the only one who has the ability to spend that asset.

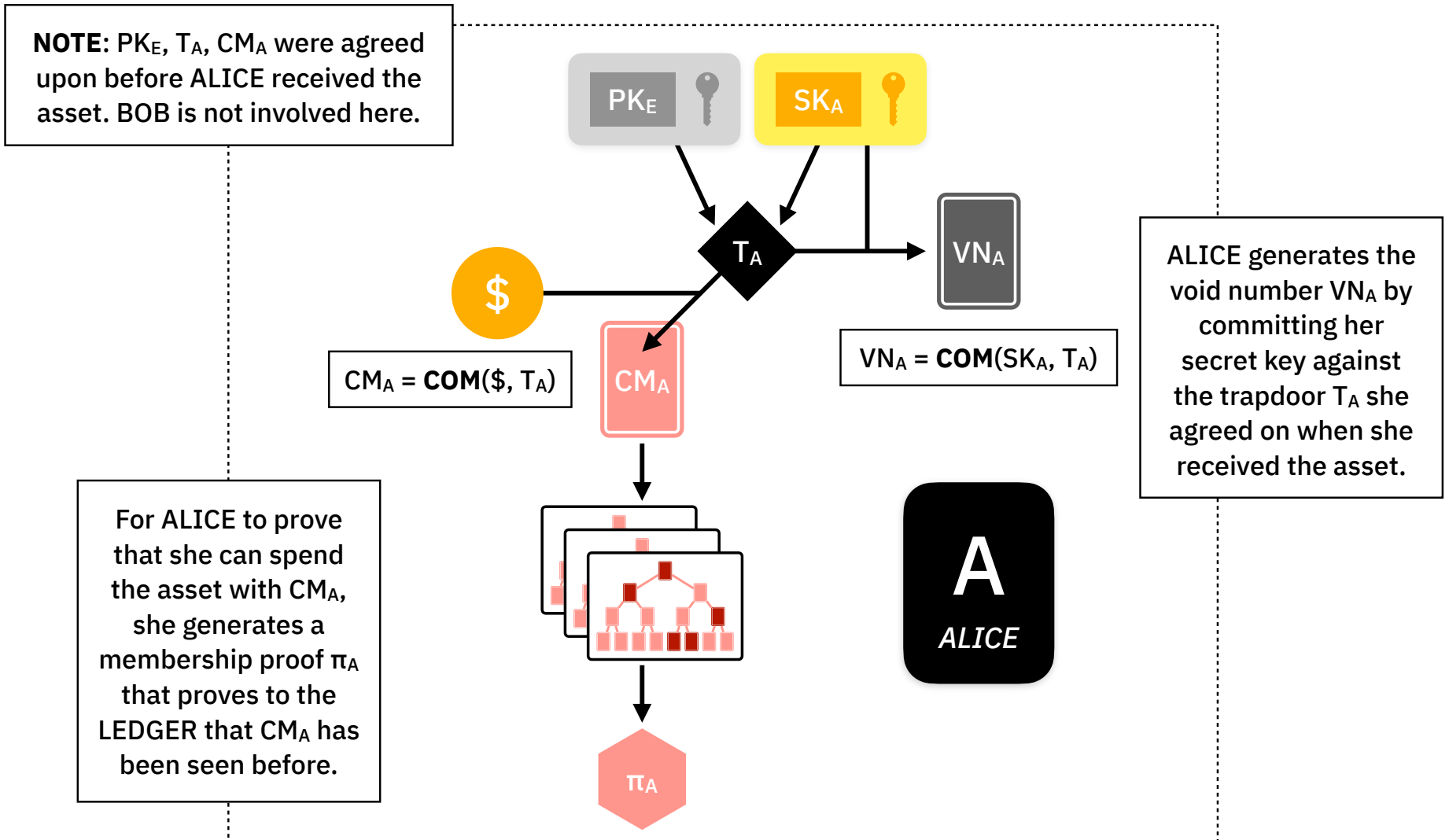

The LEDGER stores a set of UTXOs and a set of void numbers, to keep track of which assets have been given the right to be spent and which have been revoked.
LEDGER
The LEDGER must ensure that it never sees the same UTXO or void number twice in the transaction history, otherwise a double spend would have occurred, violating the fixed total supply invariant of the LEDGER.
VNA
CMB
A
ALICE
CMA
$
In-Band Secret Distribution
CMB
$
B
BOB

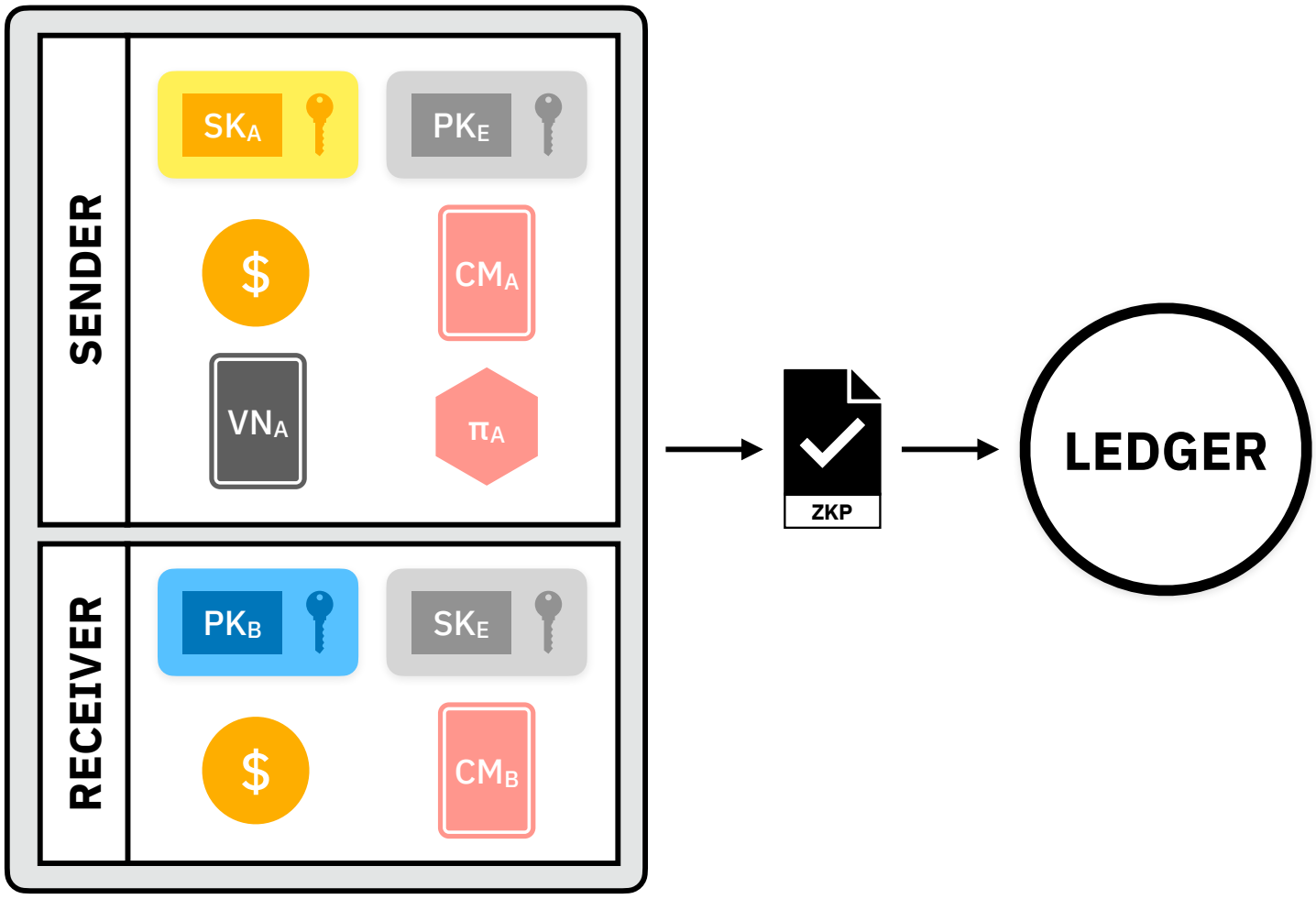

A
ALICE
SK_E
PK_B
T_B
$
CM_B
In order for ALICE to create CM_B on BOB's behalf, she computes another shared secret T_B, the trapdoor to the UTXO commitment CM_B.
SK_B
PK_E
T_B
$
CM_B
B
BOB


Elliptic Curve Diffie-Hellman
DH(SKE, PKB) = TB = DH(SKB, PKE)
SKE
PKB
SKB
PKE
TB
TB
A
ALICE
B
BOB
$
$
Pedersen Commitment
CMB
CMB
CMB = COM($, TB)


PK_E
SK_A
T_A
VN_A
$
CM_A
π_A
A
ALICE

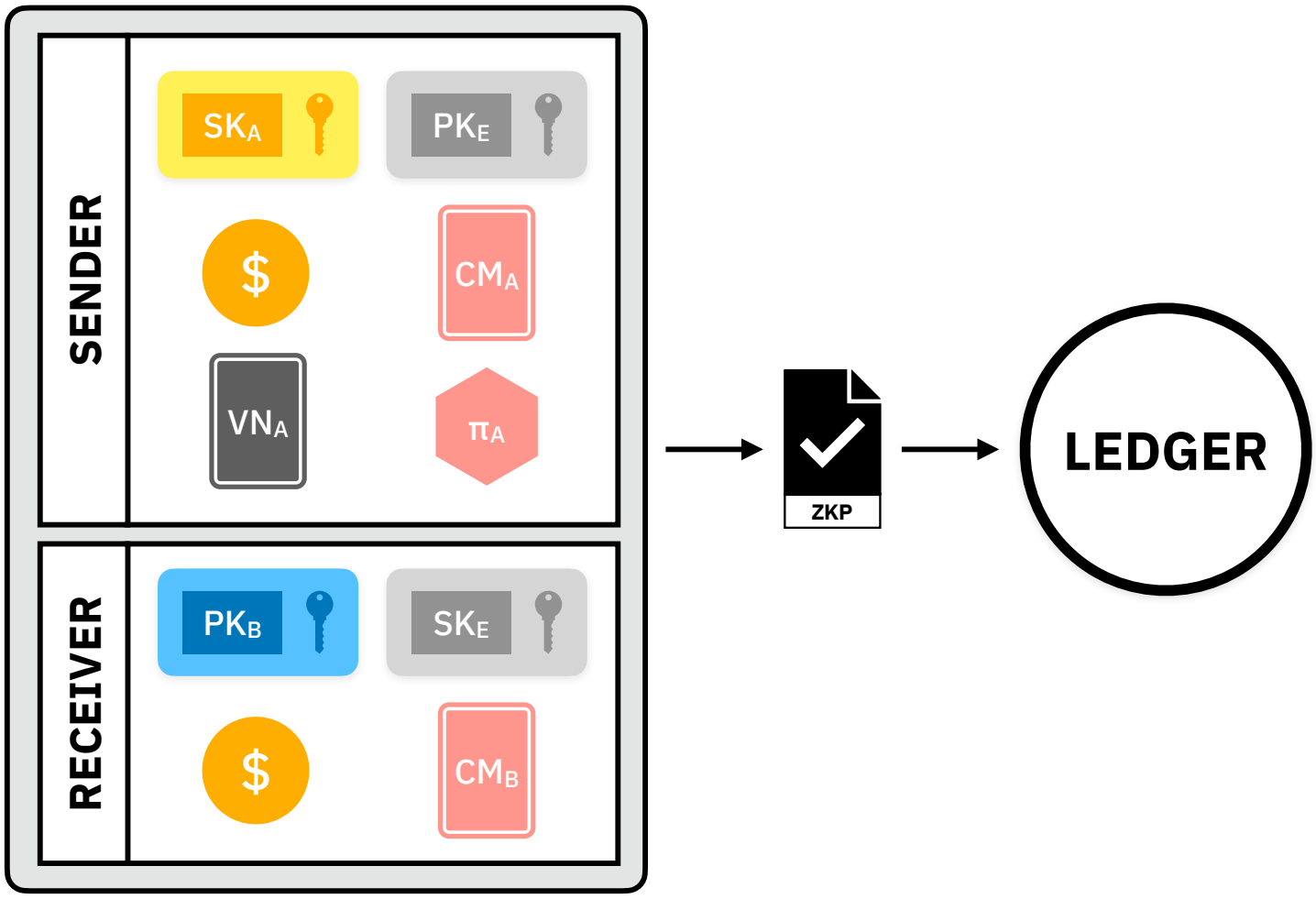

PK_E
SK_A
T_A
VN_A
$
CM_A = COM($, T_A)
CM_A
VN_A = COM(SK_A, T_A)
ALICE generates the void number VN_A by committing her secret key against the trapdoor T_A she agreed on when she received the asset.
For ALICE to prove that she can spend the asset with CM_A, she generates a membership proof π_A that proves to the LEDGER that CM_A has been seen before.
π_A
A
ALICE


NOTE: PK_E, T_A, CM_A were agreed upon before ALICE received the asset. BOB is not involved here.
PK_E
SK_A
T_A
VN_A
$
CM_A = **COM**($, T_A)
CM_A
VN_A = **COM**(SK_A, T_A)
ALICE generates the void number VN_A by committing her secret key against the trapdoor T_A she agreed on when she received the asset.
For ALICE to prove that she can spend the asset with CM_A, she generates a membership proof π_A that proves to the LEDGER that CM_A has been seen before.
A
ALICE
π_A



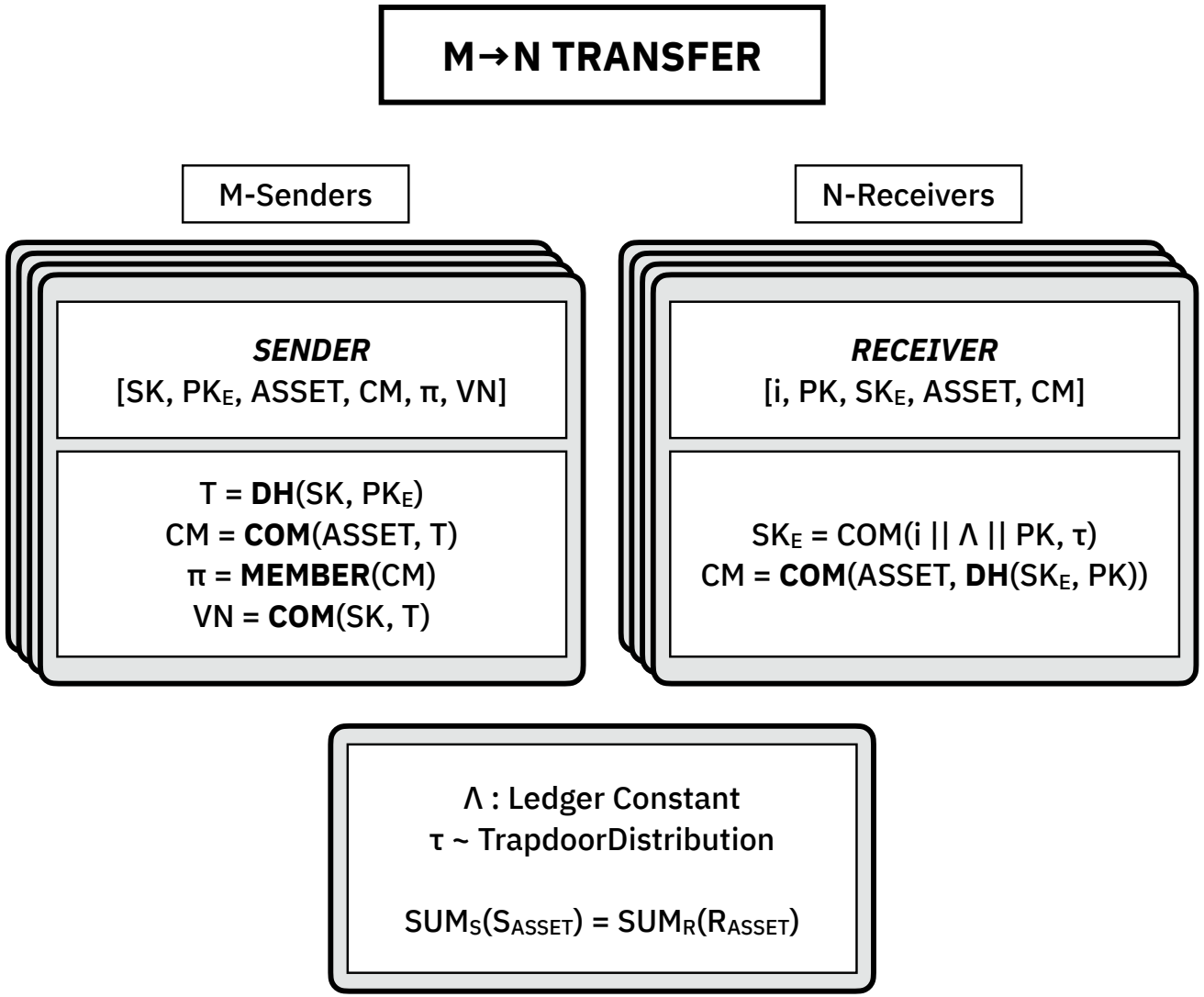
In order to demonstrate to the LEDGER the the transaction preserves the ledger invariants, ALICE computes a *zero-knowledge proof* that her asset and BOB's new asset are well-formed. This way, the LEDGER can be assured of the following:

1. ALICE had the ability to spend the asset in the past
2. BOB will have the ability to spend the asset in the future
3. ALICE will not have the ability to spend the asset again in the future



The LEDGER will also need to verify that VN_A and CM_B have not been seen before, and should reject any transaction in which this is not the case. The LEDGER will then accept the transaction otherwise and will store VN_A, CM_B, and the ENCRYPTED NOTE for future use.

If the LEDGER follows this verification protocol, then it can be assured the ALICE and BOB cannot break the total supply invariant of the ledger. ALICE and BOB can still de-anonymize the transaction if they wish, they are not bound by the LEDGER in this way.


M→N TRANSFER
M-Senders
N-Receivers
SENDER
[SK, PK_E, ASSET, CM, π, VN]
T = DH(SK, PK_E)
CM = COM(ASSET, T)
π = MEMBER(CM)
VN = COM(SK, T)
RECEIVER
[i, PK, SK_E, ASSET, CM]
SK_E = COM(i || Λ || PK, τ)
CM = COM(ASSET, DH(SK_E, PK))
Λ : Ledger Constant
τ ~ TrapdoorDistribution
SUM_S(S_ASSET) = SUM_R(R_ASSET)

THANK YOU!

SPEC: *github.com/manta-network/spec*