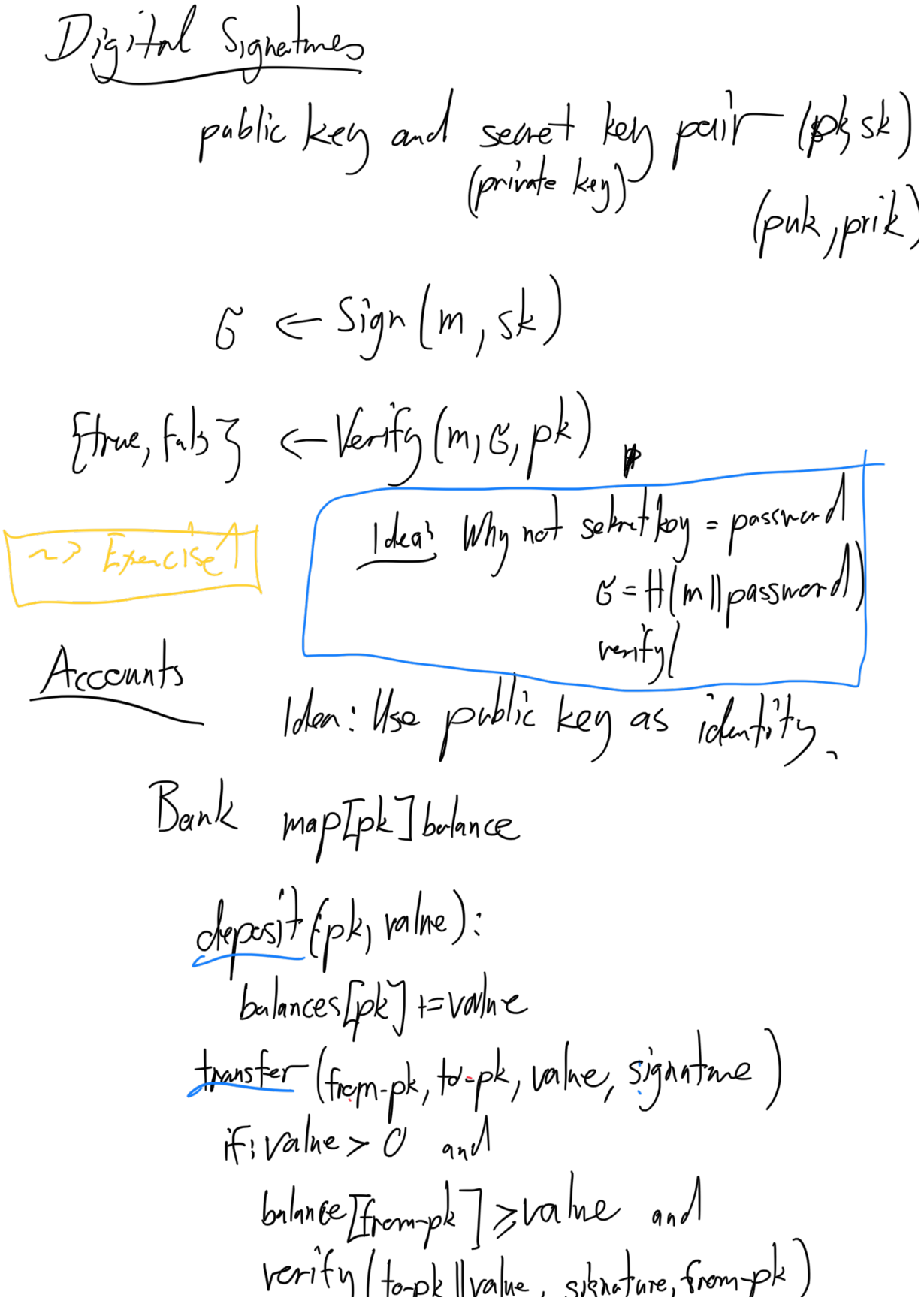
Transactions and UTXO



then

balance[from] -= val

bal[to] += val

Idea: Everybody runs the bank.

- Put transfers publicly on the blockchain.
- Everybody verifies and applies them

Problem: Replay attack!

UTXO: Unspend transaction output.

Idea: No balances but coins:

- for each coin:
  store pk of the owner.
- transfer:
  provide signature with owners key
  provide pk of new owner

Data structure:

Array $[pk_0 \mid pk_1 \mid pk_2]$

Q: What should you sign?

Ex: Send Coin 0 to pk_4

$sign("0" \| pk_4)$

Problem: Replay if pk_1 owns coin 0 again.

$[pk_4 \mid pk_1 \mid pk_3]$

Solution: Datastructure map/dict

$\{a \to pk_1, b \to pk_1, c \to pk_3\}$

$\downarrow$ transfer $(a, pk_4, sig_{pk_1}("a" \| pk_4))$

$\{d \to pk_4, b \to pk_1, c \to pk_3\}$

Now: Have coins with different value

$\{a \to (v, pk)\}$ - called output

v: value

pk: spending condition

output (v, pk)

value

spending condition, usually public key

input (id, sig)

identifier points to one output

redeeming argument (signature)

transfer (inputs , outputs)

Example: $\{a \mapsto (2, pk_1), b \to (2, pk_1)\}$

Send 3 to pk_3

two sigs; can be with different key.

inputs $(a, sig_1), (b, sig_1)$

outputs $(3, pk_3), (1, pk_1)$

change coming back.

Q: What to sign?

$$sig_1 = sk_1.sign("a" || "b" || 3 || pk_3 || 1 || pk_1)$$

Adresses vs. Utxo

+ One balance for one user.
− Easy to track one user

− One user has multiple outputs
+ Easy to use multiple private keys. (Default → create new pk for change.)
+ No storage used for empty accounts.

Spending condition:

- Small redeeming conditions safe storage
- Programmable redeeming conditions allow advanced payment control.

Examples:

	Redeeming condition	argument
a)	public key	signature
b)	hash	pubickey (hashes to condi and signature
c)		

c)	5 public keys	signature with 5 of the 5 public keys.
d)	script	argument that makes script return true

Note: Base 56 encoding ← for writing down on paper

lI

Error code