C:\Users\sarap\Github\Blockchain\cryptochain\wallet\index.test.js — 6.783s

Wallet	*has a `balance`*	passed	0.177s
Wallet	*has a `publicKey`*	passed	0.136s
Wallet > signing data	*verifies a signature*	passed	0.78s
Wallet > signing data	*does not verify an invalid signature*	passed	1.15s
Wallet > createTransaction() > and the amount exceeds the balance	*throws an error*	passed	0.162s
Wallet > createTransaction() > and the amount is valid	*creates an instance of `Transaction`*	passed	0.322s
Wallet > createTransaction() > and the amount is valid	*matches the transaction input with the wallet*	passed	0.369s
Wallet > createTransaction() > and the amount is valid	*outputs the amount the recipient*	passed	0.314s

C:\Users\sarap\Github\Blockchain\cryptochain\wallet\transaction-pool.test.js — 13.647s

TransactionPool > setTransaction()	*adds a transaction*	passed	0.319s
TransactionPool > existingTransaction()	*returns an existing transaction given an input address*	passed	0.262s
TransactionPool > validTransactions()	*returns valid transaction*	passed	5.041s
TransactionPool > validTransactions()	*logs errors for the invalid transactions*	passed	3.339s
TransactionPool > clear()	*clears the transactions*	passed	0.147s
TransactionPool > clearBlockchainTransactions()	*clears the pool of any existing blockchain transactions*	passed	1.067s

C:\Users\sarap\Github\Blockchain\cryptochain\wallet\transa

Transaction	*has an `id`*
Transaction > outputMap	*has an `outputMap`*
Transaction > outputMap	*outputs the amount to the recipient*
Transaction > outputMap	*outputs the remaining balance for the `sen*
Transaction > input	*has an `input`*
Transaction > input	*has a `timestamp` in the input*
Transaction > input	*sets the `amount` to the `senderWallet` ba*
Transaction > input	*sets the `address` to the `senderWallet` p*
Transaction > input	*signs the input*
Transaction > validTransaction() > when the transaction is valid	*returns true*
Transaction > validTransaction() > when the transaction is invalid > and a transaction outputMap value is invalid	*returns false and logs an error*
Transaction > validTransaction() > when the transaction is invalid > and the transaction input signature is invalid	*returns false and logs an error*
Transaction > update() > and the amount is invalid	*throws an error*
Transaction > update() > and the amount is valid	*outputs the amount to the next recipient*
Transaction > update() > and the amount is valid	*subtracts the amount from the original sen*
Transaction > update() > and the amount is valid	*maintains a total output that matches the*
Transaction > update() > and the amount is valid	*re-signs the transaction*
Transaction > update() > and the amount is valid > and another update for the same recipient	*adds to the recipient amount*
Transaction > update() > and the amount is valid > and another update for the same recipient	*subtracts the amount from the original sen*
Transaction > rewardTransaction()	*creates a transaction with the reward inpu*
Transaction > rewardTransaction()	*creates one transaction for the miner with*