Consider a set of decentralized solutions connected to ETH main net. It could be blockchains, rollups, or something else.

For each solution X suppose there is a way to transfer tokens (say USDT) back and forth between the main net and X. This means that there is a smart contract on the main net that freezes tokens on deposit, clones are printed on X and then burnt on exit.

Note, that in the worst case when X is hacked, money lost is limited to what was deposited on X.

Now, lets suppose you have X, Y, Z etc. and you want to enable inter-solution token transfers.

Clearly, you do not want to burn on X and print on Y

, since it couples the security. A hack in X breaks the entire system globally, not just the funds deposited on X.

Therefore, it seems that the only satisfactory solution

is atomic swaps, where some interested parties keep funds (USDT) on all chains and help you quickly transfer. The interested parties assume the risk and get reimbursed by the fee.

The atomic swaps may be hash-locked or simply reputation-based (you could transfer in small chunks to minimize counterparty risk)

The question is, how to build an efficient market so that there are many parties (not just one) and the fees are dynamically changed (maybe in a curve-like style) so higher risk and higher demand lead to higher fees …