Potential of xERC20 tokens

**Monolithic vs Modular world**

In the ever-developing world of blockchain technologies, the debate between monolithic and modular blockchain design systems has been ongoing for years. These contrasting design architectures offer unique perspectives on how to structure and manage a distributed ledger, and both have their fair share of [advantages and disadvantages](https://www.alchemy.com/overviews/modular-vs-monolithic-blockchains).

The “monolithic” blockchain design revolves around having a single blockchain do everything in one place. This means that transactions are processed and verified for correctness, and consensus is achieved on the main layer of the “core” chain. This design is the one that blockchains have usually been spun up with. This design is simpler to architect and, from a certain perspective, more secure since nodes can check the validity of transactions before reaching consensus. However, the scalability of such systems is often limited since every node must process the same transactions (and store the data related to them) to achieve consensus. Also, use-case-specific chains may be required that can successfully interoperate. Some examples of such blockchains are Ethereum 1.0, Bitcoin, and Solana.

On the other hand, modular blockchain design architecture is more flexible and incorporates the concept of “division of responsibilities” between various chain layers. This means that consensus, execution, data availability, and settlement may be divided into various layers combined in various ways. An example of such a system is the current Ethereum 2.0 architecture. [Rollups](https://www.coindesk.com/learn/what-are-rollups-zk-rollups-and-optimistic-rollups-explained/) (or other L2s) built on top of the Ethereum mainnet (L1) handle the execution part of transactions and submit the proof of execution on the mainnet. [Sharding](https://ethereum.org/en/roadmap/danksharding/) (yet to be implemented) could further increase the division of labor since only certain transactions would be executed on a single shard. While modular design architecture is more complex to build (and hence may introduce potential vulnerabilities), it might offer greater scalability and rapid adaptation.

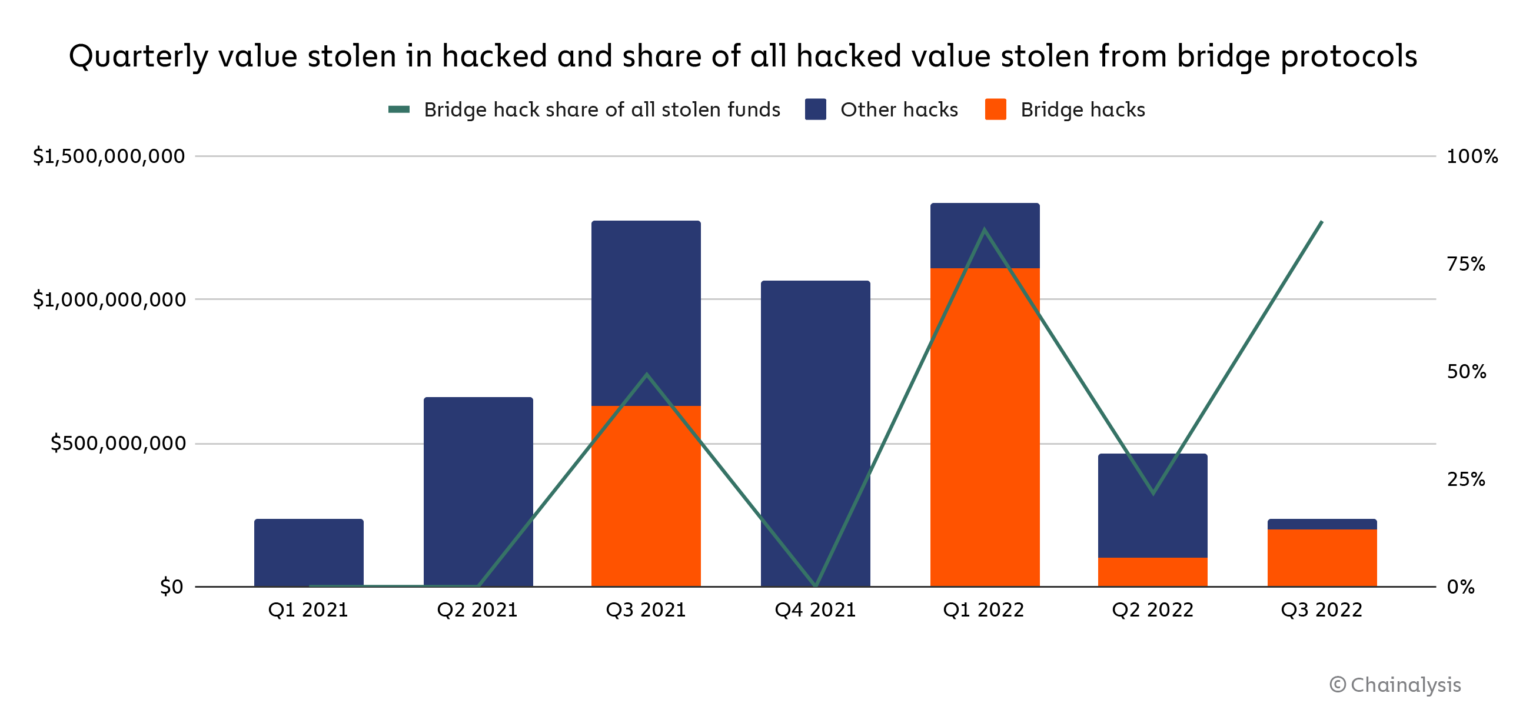
However, no matter what blockchain design wins the debate of eternal security and scalability, the division of user assets and transactions between different chains (monolithic) or layers (modular) will certainly occur, and a need for bridges will always be there.

**What are bridges and how do they operate?**

A bridge is a certain “gateway” between various blockchains that allows inter-chains to transfer assets and data. Currently, the largest amount of chain transfers occur for the ERC20 tokens (such as WETH, DAI, USDT, etc.). Protocols like [Across](https://across.to/), [Connext](https://www.connext.network/), [and Stargate](https://stargate.finance/) are a few of the [largest](https://defillama.com/bridges) bridges by volume that currently allow ERC20 transfers between Ethereum, Polygon, BNB, and other blockchains.

Though bridges are commonly used today to transfer value between chains, transferring ERC20 tokens is a vulnerable process and often results in hacks and loss of funds. For instance, according to [Chainalysis](https://blog.chainalysis.com/reports/cross-chain-bridge-hacks-2022/), attacks on bridges account for 69% of total funds stolen in 2022. The reason for this is that bridges today operate in two ways:

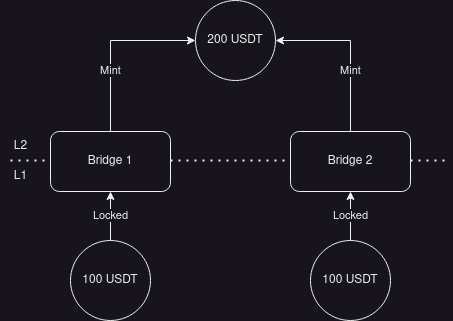
1. **Liquidity pool bridges**. For the flow of ERC20 tokens from chain A to chain B these bridges require the maintenance of 2 liquidity pools for the token on both chains. When the user requests a transfer, the funds enter the pool on chain A, and the user withdraws funds from chain B. The major disadvantage of this method is that slippage may occur in such transfers. The user is generally the bearer of slippage fees or any pool rebalancing fees that may apply.
2. **Minting-based bridges**. In a minting-based bridge, whenever ERC20 tokens are transferred, tokens on the source chain are locked in the bridge contract, and an equivalent amount of tokens are minted on the destination chain. The security of this model is controversial since the asset transferrer locks himself into the bridge until he transfers all the assets back to the source chain. The ERC20 token’s supply may also be affected if there is a vulnerability in the bridge’s logic.



Chainalysis estimates that Attacks on bridges account for 69% of total funds stolen in 2022

Traditional bridges present issues other than slippage and the risk of minting an unlimited token supply if hacked. One of these problems is liquidity fragmentation, as they don't cooperate.

**xERC20 token standard**



Imagine a scenario where a user sends 100 USDT through the Optimism bridge (Bridge 1) and an additional 100 USDT through Connext (Bridge 2) to Optimism, resulting in a total of 200 USDT on Optimism (as shown in Figure above). If the user then attempts to bridge back the 200 USDT through the Optimism bridge, they can only unlock the first 100 USDT. The remaining 100 USDT will be stuck.

Why does this happen? The issue lies in the fact that the liquidity on Layer 1 (L1) is divided between the Optimism bridge and Connext. Because of this fragmentation, the user can't retrieve the full amount through a single bridge, leading to complications and limitations in the usability of these bridging systems.

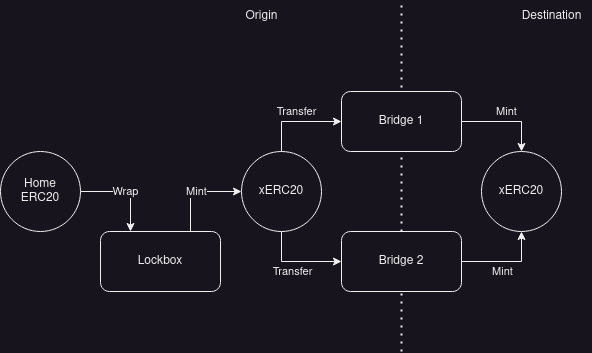
As a solution that tries to eliminate the problems related to the types mentioned above of bridges, Connext introduces the [xERC20](https://docs.connext.network/usecases/xerc20) token design. These xERC20 tokens are:

* Transferable across chains with no slippage.
* Deployed and fully controlled by you, the token issuer.
* Secured fully by Ethereum L1 and the canonical bridges of each chain.

The path is well-defined for a token issuer that wants to make their ERC20 token compatible with the xERC20 standard.

If the token contract is upgradeable, the issuer must upgrade the contracts to be compatible with the standard by implementing the relevant functions needed to make the bridging possible.

However, two options are available if the token contract is not upgradeable (or if the user does not want to upgrade the contract for other reasons). The first option is to move the liquidity of the original token into a newly deployed token and transfer all the liquidity from the original token to the newly deployed one. Though this solution may be impractical, another option is to use the Lockbox contract that wraps the original token into a token compatible with the xERC20 standard. Since the wrapping logic is well-tested (the most prominent example being the ETH-WETH pair) and reliable, this option is considered safe.



Once the xERC20 token is created and available, the bridging becomes a piece of cake. Let us assume that a user wants to bridge DAI tokens from Ethereum to Optimism. The steps he would follow are as follows:

1. DAI on Ethereum are "wrapped" (similar to how ETH are wrapped to WETH) on a smart contract named Lockbox into xDAI (this name is arbitrary - do not confuse it with Gnosis xDAI).
2. The wrapped tokens are sent to the bridge, which burns/locks the wrapped token.
3. The bridge sends a message through Optimism's canonical bridge stating that the tokens have been received on the mainnet and mints an equivalent amount of xDAI on Optimism.

For tokens that are upgraded to the xERC20 standard (or are deployed after the standard is accepted to be compatible with it), the transfer process would start directly at step 2.

On top of all this, each token issuer (the developers behind DAI in this case) can set specific limits on how many tokens can be minted by a specific bridge in a specified time period. This, coupled with the chronology mentioned above, ensures that even if a bridge is hacked, the hacker cannot mint an infinite number of DAI. This mechanism also safeguards the user in case the bridge is hacked. Since the xERC20 token is standardized across the industry (assuming the [ERC](https://connext.notion.site/Public-xERC20-Explainer-0ac9ac7dfcaa4e30b0b29c1012e72ac1) standard is widely accepted by the Ethereum community), the user may retrieve their DAI on the mainnet whenever they wish to by simply using another bridge.

**xERC20 vs OFT**

The Omnichain Fungible Token ([OFT](https://layerzero.gitbook.io/docs/evm-guides/layerzero-omnichain-contracts/oft)) standard developed and used by the LayerZero team is also known for having a "seamless and secure bridge-specific" token design. However, it is much different than the xERC20 standard.

|  | xERC20 | OFT |
| --- | --- | --- |
| Cross-bridge interoperability ? | Yes, since this would be an ERC standard, all bridges could become compatible with it | No, works only with LayerZero smart contract and 14 chains that LayerZero supports |
| Control on tokens issued ? | Yes, allows to set bridge-specific max. Limits on amount of tokens issued in a period of time | No |

Even though the OFT standard allows for an alternative method of cross-chain bridging of tokens, the ownership of the LayerZero contracts remains with the LayerZero team. Hence, while xERC20 allows for broad cross-bridge compatibility and retains control over token issuance, the OFT standard is specific to LayerZero, works only on LayerZero-supported-chains, relinquishes control over tokens, and locks them with LayerZero until the tokens remain on the destination chain.

**Implementing a xERC20 token**

Multiple teams are collaborating with the developers behind the [EIP](https://github.com/ethereum/EIPs/pull/7281) to create the first widely accepted implementations of the xERC20 token standard.

**Alchemix**

Alchemix has recently [announced](https://cointelegraph.com/news/connext-alchemix-cross-chain-token-standard-reduce-bridge-exploit-losses) the support of the xERC20 standard and onboarding projects into safely bringing their tokens to every chain. They have been working with the Connext team to bring [$alUSD](https://twitter.com/search?q=%24alUSD&src=cashtag_click), [$alETH](https://twitter.com/search?q=%24alETH&src=cashtag_click), and [$ALCX](https://twitter.com/search?q=%24ALCX&src=cashtag_click) tokens toArbitrum andOptimism using the xERC20 token standard.



**Defi Wonderland**

[Defi Wonderland](https://twitter.com/DeFi_Wonderland) is one of the teams closely researching the implementation of tokens compatible with the EIP that Connext is suggesting. They have published a suggested implementation on their [GitHub](https://github.com/defi-wonderland/xTokens/tree/52d1e5c2e8671a05f8020ab61f5e204250d945d5/solidity/contracts), which recently was audited by members of [Creed](https://thecreed.xyz/), Shayan Eskandari, and Dominik Muhs.

A closer look at the implementation reveals that the [interface](https://github.com/defi-wonderland/xTokens/blob/52d1e5c2e8671a05f8020ab61f5e204250d945d5/solidity/interfaces/IXERC20.sol) for the xERC20 contract has eight core functions that the token issuer must implement to avail the power of keeping liquidity concentrated across bridges. These are functions related to setting the LockBox contract (setLockbox), issuance limits for bridges (setLimits, mintingMaxLimitOf, burningMaxLimitOf, etc.), and the core mint and burn functions.

Other teams researching and developing the standard in collaboration with Connext include DappRadar and TokenSoft.

**Conclusion**

In the evolving world of interoperability between a plethora of independent chains, the xERC20 standard emerges as a promising solution for bridging assets across chains. Since the EIP proposed by Connext suggests the xERC20 standard as an open token standard, its permissionless nature works in line with the ethos of distributed ledgers, empowering developers and users alike.

To learn more about developments regarding the EIP, stay connected on the Ethereum Magicians [discussion](https://ethereum-magicians.org/t/erc-7281-sovereign-bridged-tokens/14979), or join the "xERC20 Builders" Telegram [group](https://t.me/+FSzAqZvIC_w2YTZh) to stay updated.