技术文件

桥

v.1.1

2023 年 2 月 13 日

内容

1 简介

区块链互操作性是指区块链与其他区块链交换数据的能力。由于近年来区块链生态系统迅速扩张，大量具有不同特定属性的网络应运而生，互操作性已成为区块链设计中的重要考虑因素。如果没有互操作性，网络就有可能与更大的生态系统隔离开来，这一事实假设会激励项目参与互操作性解决方案的研究和开发。为了解决这个问题，已经实施了多种方法，每一种方法都有特定的权衡和基础技术。本文档描述了 Polygon 团队为将本机互操作属性引入 Polygon zkEVM L2 网络而实施的解决方案。

网桥是一个基础设施组件，允许在 L1 和 L2 之间迁移资产和通信。从用户的角度来看，他们应该能够在不改变其价值或功能的情况下将资产从一个网络转移到另一个网络，并且能够在网络之间发送数据有效载荷（跨链消息传递）。

在作为 Polygon zkEVM 的 L2 rollups 中，L2 状态转换的管理和交易的数据可用性由 L1 合约保护，因此通过正确设计 L2 架构，我们还可以仅依赖于合约逻辑来同步桥的两端，无需链下可信依赖者来跨网络同步桥接端。需要注意的是，这种类型的网桥必须包含在 L2 层的设计中。

如图 1 所示，桥接接口是存在于两个网络中的桥接合约实例，用户将能够桥接资产 (1)，即在“源”网络中锁定资产，并最终认领资产的代表将由“目的地”网络中的桥接合约铸造的代币。反向操作也将是可能的（2），即销毁资产的代表代币并解锁“起源”网络中的原始资产。另一种可能的用途是作为跨链通信通道（3），即将数据有效载荷从 L1 发送到 L2，反之亦然。

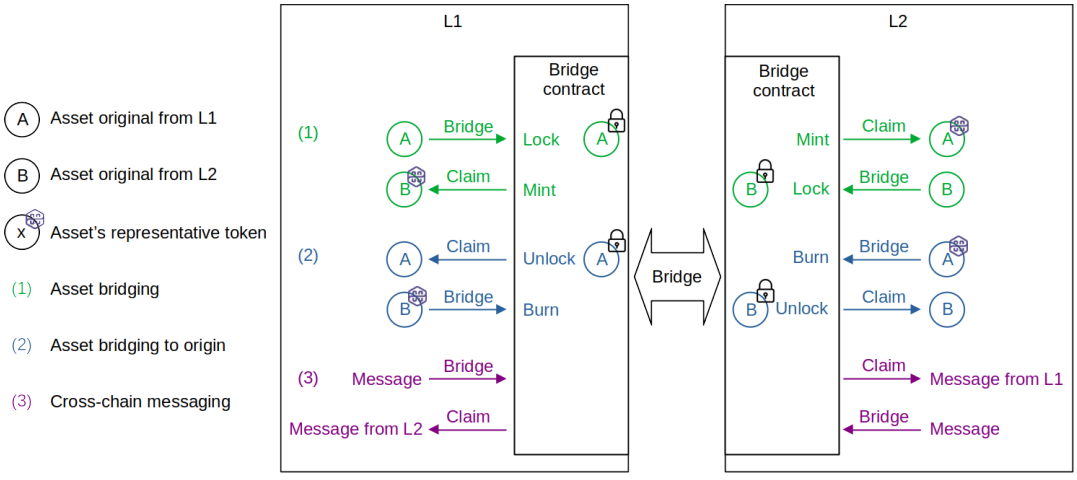

图1：多边形 zkEVM 桥架构。

2 退出 Merkle 树

桥接系统由称为全局出口默克尔树 (GEMT) 的默克尔树组成。在这棵树中，每个叶节点代表特定网络出口的 Merkle 根

默克尔树 (EMT)。GEMT只有两片叶子，一片对应L1 EMT的根，另一片对应L2 EMT的根。图 2 显示了 GEMT 的结构。

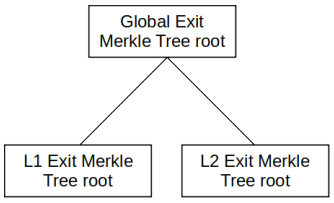

图 2：GEMT结构。

GEMT 是固定的 2 叶常规 Merkle 树，而 EMT 仅附加具有固定深度（在 Polygon zkEVM 中为 32）的稀疏 Merkle 树 (SMT)。SMT 是巨大的 Merkle 树，可以在链上以有效的方式处理（详见附录 A）。

特定网络 EMT 的每个叶子都将代表桥接资产（或资产的代表令牌）或从该网络发送消息的意图。具体来说，叶子将是以下参数的 abi 编码打包结构的 keccak256 哈希：

- **uint8 叶类型：**[0] 资产，[1] 消息。
- **int32 起源网络：**Origin Network ID，原始资产所属的网络 ID。
- **地址来源地址：**如果叶类型 = 0，原始网络令牌地址，“0x0000...0000”地址为以太保留。如果 leaf type = 1 msg.sender 消息。
- **uint32 目标网络：**桥接目标网络 ID。
- **地址 destinationAddress:**将在目标网络中接收桥接资产的地址。
- **uint256 金额：**桥接的代币/以太币数量。
- **bytes32 元数据哈希：**元数据的哈希值。元数据将包含有关资产转移或消息有效负载的信息。

一旦叶子被添加到 EMT，一个新的 EMT 根将被计算，随后，一个新的 GEMT 根。GEMT 根将在网络之间同步，从而可以证明第二个网络上的叶包含并完成桥接操作。

3 合约架构

大多数桥接架构都是在两个网络上使用智能合约实现的。然而，为了在它们之间同步 GEMT，桥接逻辑的一部分必须与 L2 状态管理架构集成。因此，为了提供对桥的全面描述，必须考虑到 L2 状态管理中涉及的链下参与者，包括 Sequencer、Aggregator 和**PolygonZkEVM.sol** 合同。

除了前面提到的组件之外，桥架构中还包含以下组件：

- **PolygonZkEVMBridge.sol:**桥接接口，允许用户与桥交互并执行桥接和声明资产或消息等操作。它在每个网络中都有一个实例并管理其 EMT。

- **PolygonZkEVMGlobalExitRoot.sol:**管理 GEMT，这涉及在每次 PolygonZkEVM.sol 合约（更新 L2 EMT）或 L1 PolygonZkEVMBridge.sol 合约（更新 L1 EMT）更新新的 EMT 时存储树和计算新的根值。充当 GEMT 历史存储库。

- **PolygonZkEVMGlobalExitRootL2.sol:**允许跨网络同步 GEMT 和 L2 EMT 根的特殊合约。具有存储 GEMT 根和 L2 EMT 根的存储槽。该合约的特殊之处在于它的存储槽直接被低级零知识证明/验证系统访问，以确保从L1同步到L2的GEMT的有效性，以及从L2同步到L1的L2 EMT的有效性。

3.1 桥接数据流。

图 3 显示了桥接器的详细架构以及组件如何相互交互以实现桥接器操作的最终性。可以看出，根据桥接操作是从 L1 到 L2（L1 -> L2）还是从 L2 到 L1（L2 -> L1），有两种可能的数据流。

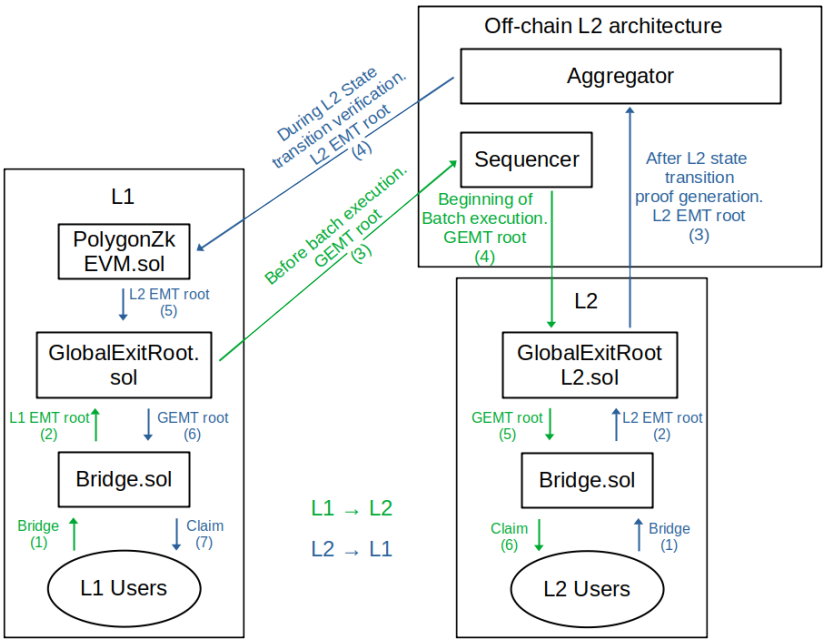

图 3：Polygon zkEVM 桥架构。

下面列举和解释每个执行流程中涉及的步骤： **L1 -> L2：**

- **(1):**用户必须调用 L1 的桥接功能**PolygonZkEVM-Bridge.sol**合同。根据要桥接的资产类型，合约内部结构将有所不同。如果桥接请求有效，合约将根据请求的特征填充 L1 EMT 叶子，并将其添加到树中，并计算新的 L1 EMT 根。

- **(2):**在同一 L1 交易中**PolygonZkEVMBridge.sol**合同将调用 **PolygonZkEVMGlobalExitRoot.sol**合同更新新的 L1 EMT 根并因此计算新的 GEMT 根。

- **(3):**最终，Sequencer 将从中获取新的 GEMT 根**PolygonZkEVMGlobalExitRoot.sol**合同。

- **(4):**新的 GEMT root 会进入到 GEMT 的特殊存储槽中**PolygonZkEVMGlobalExitRootL2.sol**交易批量执行开始时的合约，允许L2用户访问它。

- **(5) & (6):**要完成桥接过程，用户必须调用 L2 的声明功能**PolygonZkEVMBridge.sol**合同并向其提供先前包含的叶子的 Merkle 包含证明。合约将从**PolygonZkEVMGlobalExitRootL2.sol**合约并验证包含证明的有效性。如果包含证明有效，合约将根据被桥接的资产类型不同地完成桥接过程。如果包含证明无效，则交易将被还原。

L2 -> L1：

- **(1):**用户必须调用 L2 的桥接功能**PolygonZkEVM-Bridge.sol**合同。根据要桥接的资产类型，合约内部结构将有所不同。如果桥接请求有效，合约将根据请求的特征填充 L2 EMT 叶子，并将其添加到树中，并计算新的 L2 EMT 根。

- **(2):**在同一 L2 交易中**PolygonZkEVMBridge.sol**合同将调用**PolygonZkEVMGlobalExitRootL2.sol**合同更新新的 L2 EMT。

- **(3):**最终，聚合器将生成包含桥接交易的批处理序列的计算完整性的零知识证明。新的 L2 EMT 将从执行产生的 L2 状态中获得。

- **(4):**聚合器将生成的 L2 EMT 连同零知识证明一起提交给 L1**PolygonZkEVM.sol**合同。

- **(5): PolygonZkEVM.sol**合约将验证零知识证明的有效性，如果有效，合约将调用**PolygonZkEVMGlobalExitRoot.sol** 合同更新新的 L2 EMT 根并因此计算新的 GEMT 根。

- **(6) & (7):**要完成桥接过程，用户必须调用L1的声明功能**PolygonZkEVMBridge.sol**合同并向其提供先前包含的叶子的 Merkle 包含证明。合约将从**PolygonZkEVMGlibalExitRoot.sol**合约并验证包含证明的有效性。如果包含证明有效，合约将根据被桥接的资产类型不同地完成桥接过程。如果包含证明无效，则交易将被还原。

3.2 PolygonZkEVMBridge.sol

PolygonZkEVMBridge.sol是充当特定网络中用户的桥接接口的合约，因此在每个网络中都有一个实例。具有容纳每个网络的 EMT 所需的存储槽以及与之交互的所有必要功能。

3.2.1 桥接功能

桥梁资产是用于将资产桥接到另一个网络的功能：

```
1个   功能桥梁资产（
2个           地址        令牌，
3个           uint32    目的地网络，
4个           地址        目的地地址，
5个           uint256     数量，
6个           字节通话数据          许可数据
7       )
```

bridgeAsset 函数参数：

- **令牌：**源网ERC20代币地址，若为“0x0000...0000”表示用户要桥接以太币。
- **目的地网络：**目标网络的网络 ID，必须与调用函数的网络 ID 不同，否则交易将恢复。
- **目的地地址：**将在目标网络中接收桥接令牌的地址。
- **数量：**桥接代币数量
- **许可数据：**EIP-2612 许可延期的 ERC-20 代币签名许可数据，用于更改账户的 ERC-20 配额，并允许桥接合约将要桥接的代币转移给自己。

如图 4 所示，由于可以桥接三种不同类型的资产，因此存在三种可能的执行流程**桥梁资产**功能：

- **(1）要桥接的资产是以太币。**
- **(2) 资产是来自另一个网络的 ERC-20 代币的代表性 ERC-20 代币。**
- **(3) 资产是来自该网络的 ERC-20 代币。**

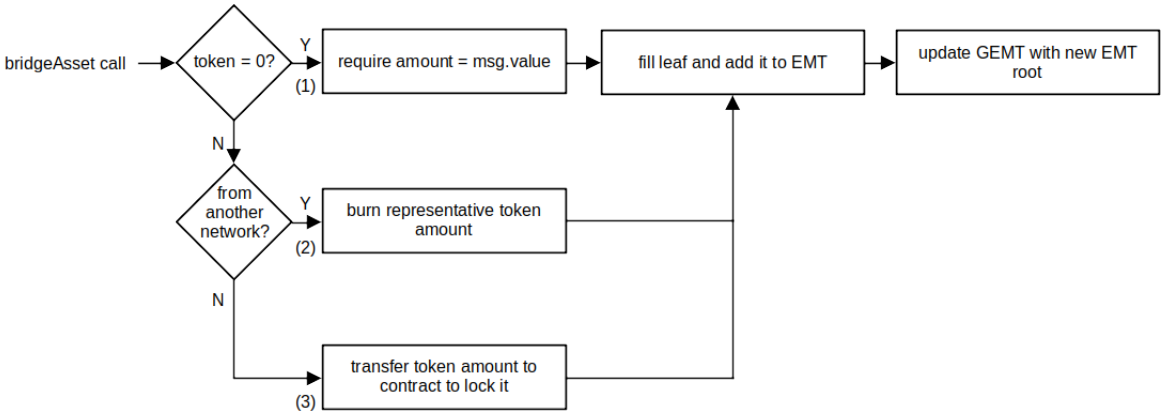

图 4：可能的执行流程**桥梁资产**功能取决于资产的类型。

(1) 桥接资产为以太币：

如果**令牌**参数等于“0x0000...0000”表示要桥接的资产是以太币，交易的 msg.value 应与**数量**争论。以太币数量将被锁定在合约中，以便继续进行桥接过程。请注意，L1 和 L2 中的以太币以相同的方式处理，实际上，它以 1:1 的交换比率桥接，并将成为两个网络中的天然气支付（本机）代币。由于以太是 L1 的原始叶参数**起源网络**将被设置为 L1 网络 ID。

(2) 该资产是来自 ERC-20 代币的代表性 ERC-20 代币另一个网络：

代表性的 ERC-20 代币由**PolygonZkEVMBridge.sol** 合约，负责铸造和销毁它们。此外，合约有一个名为**wrappedTokenToTokenInfo**作为部署在该网络上的代表性 ERC-20 代币合约的列表。对于部署的每个合约代表令牌，此映射将在**代币信息**struct，以代表token合约的地址为key。

```
1个    结构代币信息                        {
2个        uint32     起源网络;
3个        地址         原始令牌地址;
4个    }
```

如果**令牌**参数值作为键存在于**wrappedTokenToTokenInfo** 映射，这意味着被桥接的令牌是来自另一个网络的 ERC-20 令牌的代表性 ERC-20 令牌。为了继续进行桥接过程，对应于**数量**争论将被销毁，桥有权在没有用户许可的情况下销毁代币。叶子的参数，**原地址** 和**起源网络**, 将从**wrappedTokenToTokenInfo**映射的对应条目。

(3) 资产是来自该网络的 ERC-20 代币：

最后一个可能的流程发生在要桥接的资产是来自该网络的 ERC20 代币原始资产时。在这种情况下，为了继续进行桥接过程，对应于**数量**参数将被锁定在合同中。请注意，要成功地将代币转移给自己，合约必须至少允许用户想要桥接的代币数量。如果代币合约支持 EIP-2612 许可延期，则许可也可以在同一笔交易中完成 **许可数据**争论。叶子的参数，**起源网络**和 **originTokenAddress**, 将分别设置为当前网络 ID 和 ERC-20 代币合约地址。这 **元数据哈希**叶子的参数也将计算如下。该参数将确保在目标网络中使用完全相同的 ERC-20 代币参数部署具有代表性的 ERC-20 代币合约。

```
1个      元数据哈希          =keccak256(
2个        阿比 编码（
3个            IERC20MetadataUpgradeable（令牌）。名称（），
4个            IERC20MetadataUpgradeable（令牌）。符号 ()，
5个            IERC20MetadataUpgradeable ( 令牌 )。小数点 （）
6个        )
7       );
```

无论资产类型如何，最后的执行步骤都是常见的。剩余叶子的参数，**叶型**, 将被设置为 0（类型资产）和**目的地网络**和 **目的地址**将根据参数设置**桥梁资产**函数调用。首先，一个**桥接事件**将发出包含叶的所有信息的事件。然后，新叶子将被包含在 EMT 中。最后，将调用 GEMT 合约来更新新的 EMT 根。

桥接消息是用于将消息桥接到另一个网络的函数：

```
1个   功能桥接消息 (
2个           uint32    目的地网络，
3个           地址        目的地地址，
4个           字节记忆          元数据
5个       )
```

bridgeMes sage 函数参数：

- **目的地网络：**目标网络的网络 ID，必须与调用函数的网络 ID 不同，否则交易将恢复。
- **目的地地址：**将接收桥接消息的目标网络中的地址。
- **元数据：**消息负载。

桥接消息函数将直接发出一个**桥接事件**事件，将向 EMT 添加一个新叶子，并将调用 GEMT 更新新的 EMT 根作为**桥梁资产** 功能确实如此。主要区别在于它创建的叶子是消息类型（leafType = 1），因此，**来源地址**和**元数据哈希**leaf 的值将分别是 msg.sender 和消息有效负载的哈希值。用户可以通过简单地将金额作为 msg.value 添加到**桥接消息**函数调用交易，此金额将作为价值发送**目的地址**将被调用以在目标网络中接收消息。

3.2.2 声明函数

由于 L2 帐户默认情况下没有以太币来支付交易费用，因此当从 L1 索取桥接资产或消息时，调用桥接索取功能的 L2 索取交易由协议资助，不需要支付燃料费。

索取资产是用于声明从其他网络桥接的资产的功能：

```
1个   功能索赔资产（
2个           字节 32[]      记忆       smt打样，
3个           uint32    指数，
4个           字节 32     主网出口根，
5个           字节 32     汇总退出根，
6个           uint32    起源网络，
7             地址        原始令牌地址，
8个           uint32    目的地网络，
9             地址        目的地地址，
10            uint256    数量，
11            字节记忆          元数据
12        )
```

claimAsset 函数参数：

- **smt证明：**Merkle证明，即验证叶子所需的兄弟节点数组。
- **指数：**叶子的索引。
- **主网退出根：**包含叶子时的 L1 EMT 根。
- **汇总退出根：**包含叶子时的 L2 EMT 根。
- **起源网络：**Origin Network ID，原始资产所属的网络 ID。

- **原币地址：**原始网络中的 ERC20 令牌地址，如果其 0x0000..0000 表示正在声明以太币。
- **目的地网络：**目标网络的网络 ID，即进行呼叫的网络。
- **目的地地址：**将接收桥接令牌的地址。
- **数量：**认领的代币数量。
- **元数据：**如果被认领的资产不是来自进行调用的网络的原始资产，则原始 ERC-20 令牌（名称、符号、小数）的 ABI 编码元数据将用作元数据。如果是以太币或来自正在进行调用的网络的原始令牌，则该值将为 0。

这**索取资产**函数将验证与用户提供的参数相对应的叶子的有效性。

为防止重放攻击，应采取措施确保给定叶子只能成功验证一次。**PolygonZkEVMBridge.sol**合同有一个 **声称位图**映射为每个已经成功验证的叶索引存储一个无效位。如图 5 所示，为了优化存储槽的使用，映射中的每个条目将为 256 个已验证的叶子保存 256 个无效位。

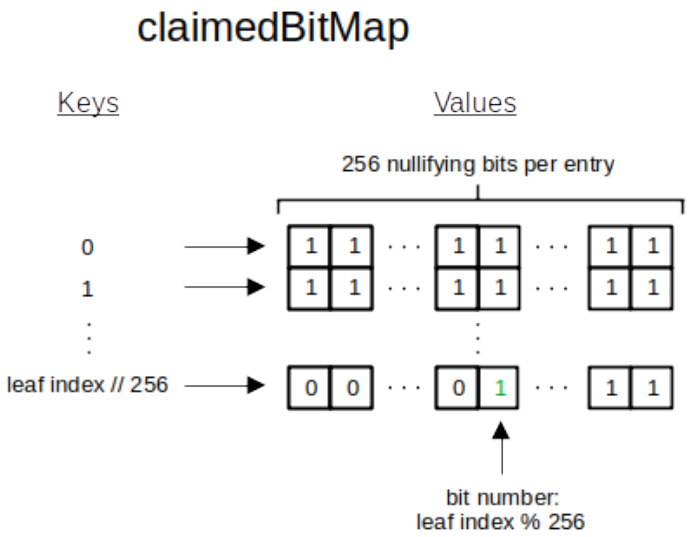

图 5：claimedBitMap无效符映射结构。

为了使特定叶子的 Merkle 证明被视为有效，必须满足以下条件：

- 叶子的无效位必须在**声称位图**映射。
- 叶子的**目的地网络**参数必须与网络 ID 相匹配**索取资产**正在调用函数。
- 散列的 GEMT 根**主网退出根**和**汇总退出根**参数必须已经存在于 **PolygonZkEVMGlobalExitRoot.sol**合同。
- Merkle 证明必须有效，这意味着它应该产生预期的GEMT 根。

如果叶子被成功验证，叶子索引将通过设置它的相应位来取消**声称位图**映射，然后，如图 6 所示，由于可以声明三种不同类型的资产，因此存在三种可能的执行流程**索取资产**功能：

- **(1）要认领的资产是以太币。**
- **(2) 资产是来自该网络的 ERC-20 代币。**
- **(3) 资产是来自另一个网络的 ERC-20 代币的代表性 ERC-20 代币。**

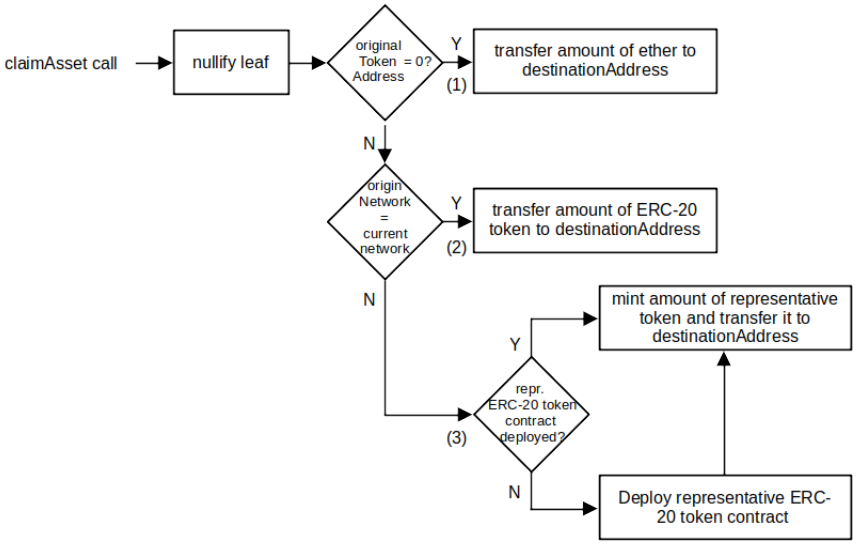

图 6：可能的执行流程**桥梁资产**功能取决于资产的类型。

(1) 被认领的资产为以太币：

如果**originTokenAddress**参数等于“0x0000 ... 0000”意味着被要求的资产是以太币，因此交易价值为**数量**以太币的参数数量将被发送到**目的地址**参数地址。由于不能按需铸造以太币，**PolygonZkEVMBridge.sol**部署在 L2 上的余额为 100000000000 以太币作为以太桥接流动性。假设 L2 中的所有以太币都是从 L1 桥接的，所以 L1 的 **PolygonZkEVMBridge.sol**不需要事先有以太币余额，L2 中的每个以太币 wei 都会有一个支持以太币 wei 在 L1 合约中被阻止。请注意，L2 的预制流动性**PolygonZkEVMBridge.sol**对以太没有任何通胀影响。

(2) 资产是来自该网络的 ERC-20 代币：

如果**起源网络**参数等于进行调用的网络 ID，这意味着被认领的资产是该网络的原始资产，应该已经锁定在**PolygonZkEVMBridge.sol**合同。因此，**数量** ERC-20 代币的参数数量将被发送到**目的地址**参数地址。

(3) 该资产是来自ERC-20代币的代表性ERC-20代币另一个网络：

最后一个可能的流程发生在被认领的资产是来自另一个网络的 ERC20 代币时。

这**PolygonZkEVMBridge.sol**合同有一个映射称为**tokenInfoToWrapped-令牌**它存储部署在上的代表性 ERC-20 代币合约的地址

网络。在使用 create2 函数部署合约期间使用的盐用作映射的每个条目中的键。

盐的计算如下**起源网络**和**originToke地址** 参数：

```
1个    keccak256(
2个        阿比 encodePacked（原始网络，                    来源令牌地址）
3个    );
```

合约将检查被认领资产的代表性 ERC-20 代币合约是否存在于**tokenInfoToWrappedToken**映射。如果是，则表示代表ERC-20代币合约已经部署，并且在**数量**参数将被铸造到指定的地址 **目的地址**争论。

如果不存在，将使用 create2 函数和先前计算的盐部署一个新的代表性 ERC-20 代币合约。create2 函数和此特定盐的使用将确定性地将代表性代币合约的地址绑定到原始网络上原始代币合约的地址。然后，在指定的数量**数量**参数将被铸造到指定的地址 **目的地址**参数，和一个**NewWrappedToken**事件将被发出。

一个新条目**tokenInfoToWrappedToken**和**wrappedTokenToTokenInfo** 将为新的代表性 ERC-20 代币合约添加映射。

最后，无论索赔的资产种类如何，**索赔事件**事件将被发出。

认领留言是用于声明从其他网络桥接的消息的函数：

```
1个    功能声明消息 (
2个            字节 32[]       记忆       smt打样，
3个            uint32     指数 ，
4个            字节 32      主网出口根，
5个            字节 32      汇总退出根，
6个            uint32     起源网络，
7              地址         原地址，
8个            uint32     目的地网络，
9              地址         目的地地址，
10             uint256     数量 ，
11             字节记忆              元数据
12         )
```

认领留言， 作为**索取资产**函数确实如此，将尝试验证用户给出的叶子，因为叶子格式相同，所以两个函数将采用相同的参数。再一次，作为**索取资产**函数确实如此，如果叶子被成功验证，叶子索引将通过设置它的相应位来取消**声称位图**映射。然后，将进行低级调用**目的地址**参数如下：

```
1个 //    执行          信息
2个 //    转移           醚
3个 /*      solhint - 禁用              避免 - 低 - 水平 - 电话                 * /
4个 (bool 成功,) = 目的地地址。调用{值：数量}（
5个     阿比 编码调用 (
6个         IBridgeMes  sageReceiver 。onMessageReceived ,
7           ( originAddress , originNetwork , 元数据 )
8个     )
9    );
```

可以看出，调用数据是为了调用一个函数而设置的**onMessageReceived**通过 **原地址,起源网络,元数据**参数，如果消息包含以太币，

调用值将设置为**数量**争论。**元数据**将是消息有效负载。请注意，消息服务可用于将以太币转移到外部拥有的账户（EOA），但是它们无法解释该消息，因此消息有效负载将无法使用。

最后，如果消息发送成功，**索赔事件**事件将被发出。

3.3 PolygonZkEVMGlobalExitRoot.sol

PolygonZkEVMGlobalExitRoot.sol是计算和存储每个新 GEMT 根的 L1 合约。为了将来可以验证任何添加的叶子，需要存储每个 GEMT 根。名为的映射**globalExitRootMap**持有所有计算的 GEMT 根。L1**PolygonZkEVMBridge.sol**将在叶验证过程中从该映射中获取 GETM 根。

如图 7 所示，**更新退出根目录**函数用于更新 EMT 根的值并计算新的 GEMT 根。如果它被调用**PolygonZkEVM.sol** 合同，L2 EMT根将被更新。如果它被调用**PolygonZkEVM-Bridge.sol**合同，L1 EMT根将被更新。这个函数将只接受来自**PolygonZkEVM.sol**或来自**L1 PolygonZkEVMBridge.sol**合同。

```
1个    功能更新出口根（字节 32新根）外部的
```

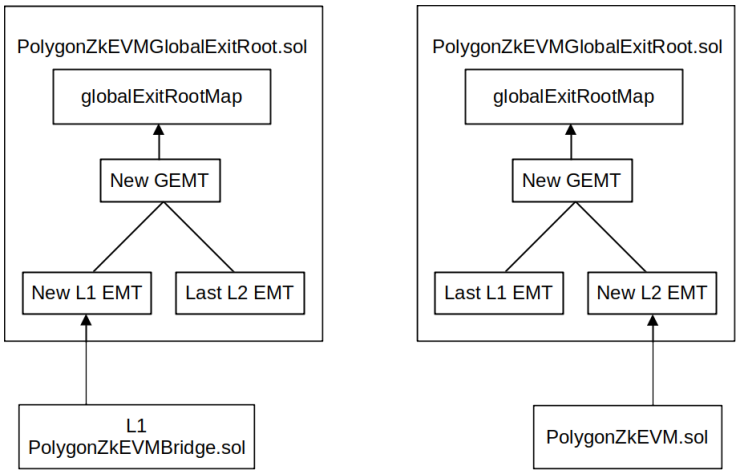

图 7：的两个执行流程**更新退出根目录**功能。

当每个 L2 状态转换由**PolygonZkEVM.sol**合约（通过 Aggregator 给出的 ZK proof 验证），一个新的 L2 EMT 将在**PolygonZkEVMGlobalExitRoot.sol**通过调用合同**更新退出根目录**功能。

随着每个新叶子添加到 L1 EMT**PolygonZkEVMBridge.sol**合同，一个新的 L1 EMT 根将需要更新**PolygonZkEVMGlobalExitRoot.sol**合同等**PolygonZkEVMBridge.sol**将会通知**更新退出根目录**函数以更新新的 L1 EMT 根。

终于有活动了**更新全局退出根**将被发射。

3.4 PolygonZkEVMGlobalExitRootL2.sol

PolygonZkEVMGlobalExitRootL2.sol是部署在 L2 中的“特殊”合约，允许 L2 **PolygonZkEVMBridge.sol**访问 GEMT 根的实例他-

保守党。请注意，L2**PolygonZkEVMBridge.sol**实例不能直接调用 L1 **PolygonZkEVMGlobalExitRoot.sol**合同查询 GEMT 根，因为它部署在另一个网络中。

PolygonZkEVMGlobalExitRootL2.sol作为**PolygonZkEVMGlobalExitRoot.sol** 有映射**globalExitRootMap**存储 GEMT 根，而不是自己计算它们**globalExitRootMap**关于 L1 中的映射，映射在网络之间同步**PolygonZkEVMGlobalExitRoot.sol**合同。

PolygonZkEVMGlobalExitRootL2.sol存储由 L2 更新的最后一个 L2 EMT 根值 **PolygonZkEVMBridge.sol**每次将新叶子添加到 L2 EMT 时的实例。
PolygonZkEVMGlobalExitRootL2.sol是“特殊”的，因为**globalExitRootMap** 和 **lastRollupExitRoot**在事务批处理执行期间，zkEVM 节点 SW 直接访问存储槽。

为了允许 L2 用户索取桥接资产，**globalExitRootMap** 必须在批处理开始时由 zkEVM 节点同步，然后是 L2 **PolygonZkEVMBridge.sol**合约将有权访问有效的 GEMT 根以验证用户声明批次中包含的交易。

L2 EMT根查询形式**lastRollupExitRoot**zkEVM 节点在批处理执行后充当聚合器的变量。然后 L2 EMT 根将与 L2 状态转换证明一起发送到 L1**PolygonZkEVM.sol**合约，如果证明验证成功，它将在 L1 中更新新的 L2 EMT 根**PolygonZkEVM-GlobalExitRoot.sol**合同和新的 GEMT 将被计算以允许 L1 **PolygonZkEVMBridge.sol**合同以访问有效的 GEMT 根以验证用户声明包含在聚合批次中的交易。

一种 Gas 高效的仅追加稀疏 Merkle 树

稀疏 Merkle 树是一种大小难以处理的 Merkle 树，可以以有效的方式处理，假设它几乎是空的，实际上，最初它是空的。当其中的所有叶子都具有相同的零（空）值时，它被认为是空的，由于这个假设，可以通过计算来计算根 $日志_{2个}(n)$ 哈希运算，其中n是树上叶子的数量。请注意，在非稀疏树中，我们需要计算 $2ñ$ -1 哈希运算计算根。在计算空树根时，每一层的所有节点都取相同的值，因此不需要计算每一层的所有子树。当树的特定级别为空时，每个取一个节点的值都被命名为零散列，并且将代表一个子树x零叶。

例如，对于 3 层（8 个叶子）的空 sparce Merkle，零哈希列表将如下所示：

- 0 级节点 = 0
- 1 级节点 = H(0,0) = $ZH_{1个}$
- 2 级节点 = H($ZH_{1个}$,$ZH_{1个}$) = $ZH_{2个}$
- 3 级节点 = H($ZH_{2个}$,$ZH_{2个}$) = $ZH_{3个}$=默克尔根

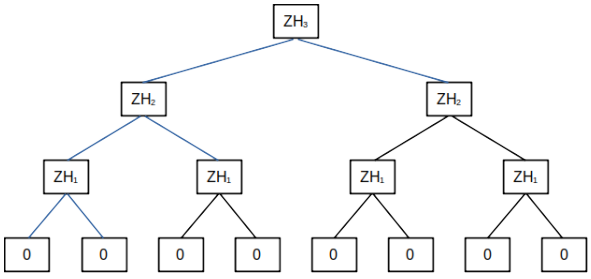

图 8： 8 叶空稀疏 Merkle 树。

要向树中添加新值，我们将仅使用 1 + *日志*2个(*n*) 哈希操作，如果我们没有预先计算它们，除了 ZH 评估。事实上，动态计算 ZH 比使用预先计算的值读取存储槽更节省 gas。添加的每个新值将填充树的一个空叶。

继续前面的示例，添加第一个值 (L0) 我们将计算以下哈希操作：

- 高（长0) =乙0
- 高（乙0,0) = 乙1个
- 高（乙1个,ZH1个) =乙2个
- 高（乙2个,ZH2个) = 默克尔根

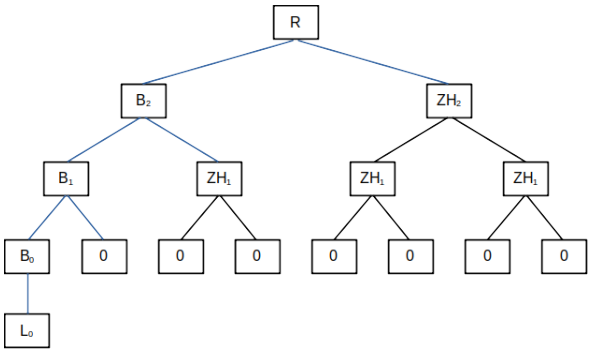

图 9： 8 叶稀疏 Merkle 树第一叶包含。

请注意，我们正在计算一个*n*离开 Merkle 根，仅使用 1 + *日志*2个(*n*) 哈希操作，对于任何连续添加的叶子，我们都会遇到相同的情况。乙*x*
(Branch) 是所有已经计算出的子树的值，它将是链上存储树所需的存储槽。因此，在链上实现增量 Sparce Merkle 树所需的整个存储槽将为 1 + *日志*2个(*n*).

关于包含证明验证操作，常规 Merkle 树和稀疏树没有区别，可以继续使用 1 + 高效地完成 *日志*2个(*n*) 哈希运算。此外，为了生成包含证明，需要访问所有已经包含的树级别，但是不需要使用存储槽，因为我们在包含交易调用数据中或通过发出包含期间的事件。

总而言之，SMT 是巨大的 Merkle 树，可以以高效的方式处理，并且可以使用几个存储槽进行存储。链上计算的效率意味着更少的交易 gas 成本以及更低的存储使用量。

乙　桥接流程

B.1　L1 -> L2

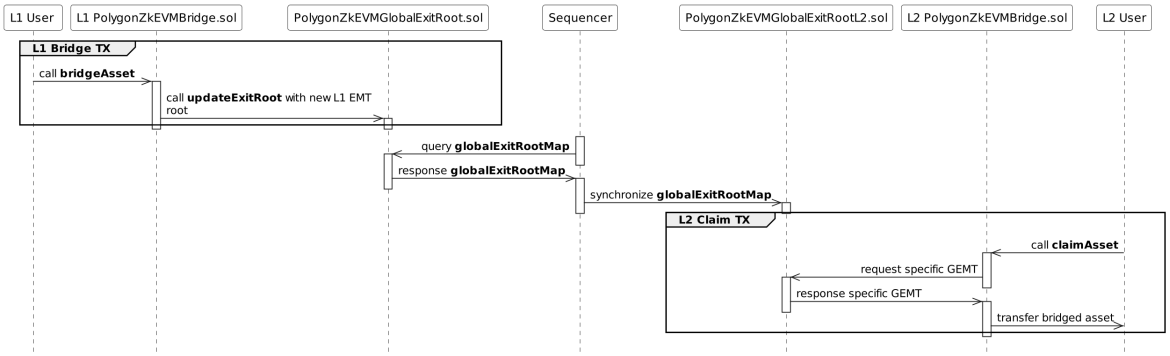

图 10：L1 -> L2 桥接流。

B.2 L2 -> L1

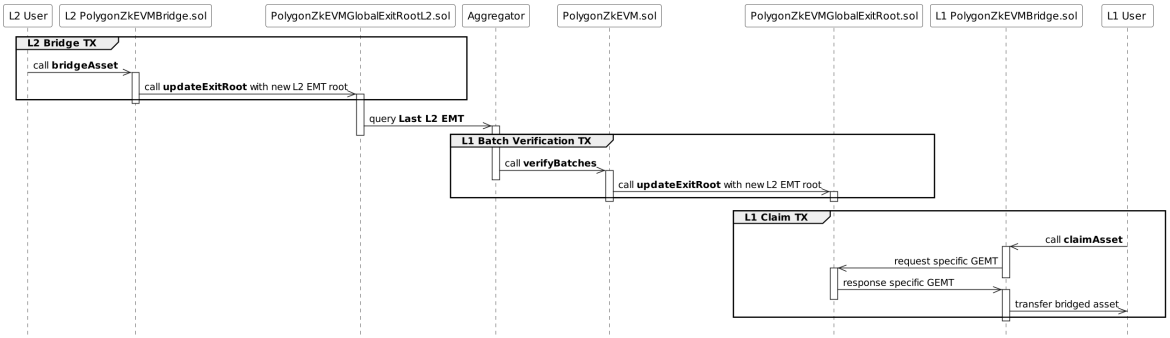

图 11：L2 -> L1 桥接流。