**1.1 结构简介**

算法的输入为比特的初始密钥和比特的初始向量，输出为比特的密钥字序列。其整体结构如下图所示，由线性反馈移位寄存器，非线性函数和组成。如下图所示：

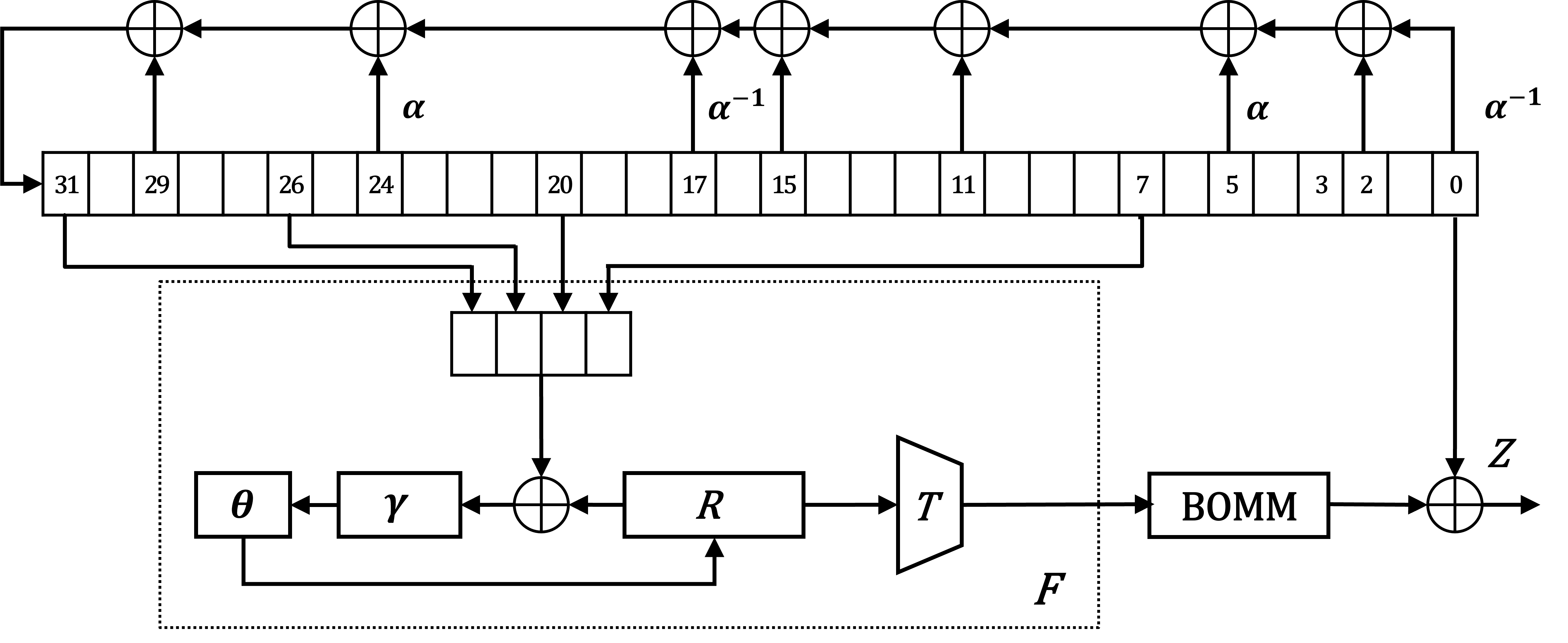


图 1-1 的整体结构

算法分两种状态：初始化状态和运行状态。

算法开始时，先将一个比特的初始密钥和一个比特的初始向量放到中，也是如此，并且将中的记忆单元全部置。在初始化阶段的输出参与的反馈计算而不输出密钥流。此过程经过轮之后，的输出不再参与的反馈计算，二是与当前时刻的，进行异或运算，从而产生密钥流，这就是运行状态。

其中结构如下图所示：

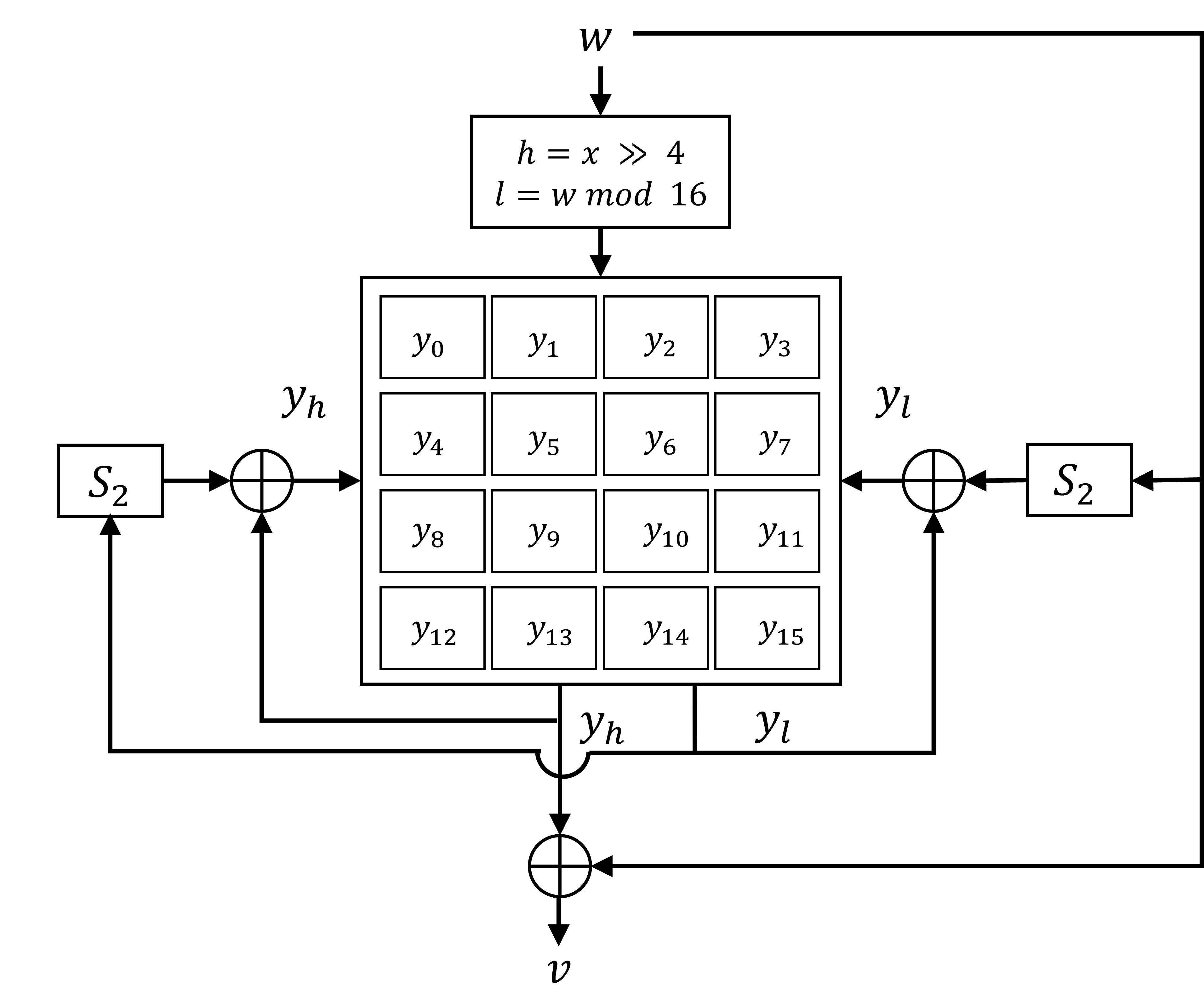


图 1-2 算法结构示意图

**1.1.1 线性反馈移位寄存器**

的由个位的寄存器组成，共字节，算法定义在域上。的包含一个函数，它的定义如下：

其中是本源多项式的一个根，也就是。

假设为在时刻的状态，为时刻的状态，则的反馈计算如下：

线性反馈移位寄存器结构如下图所示：

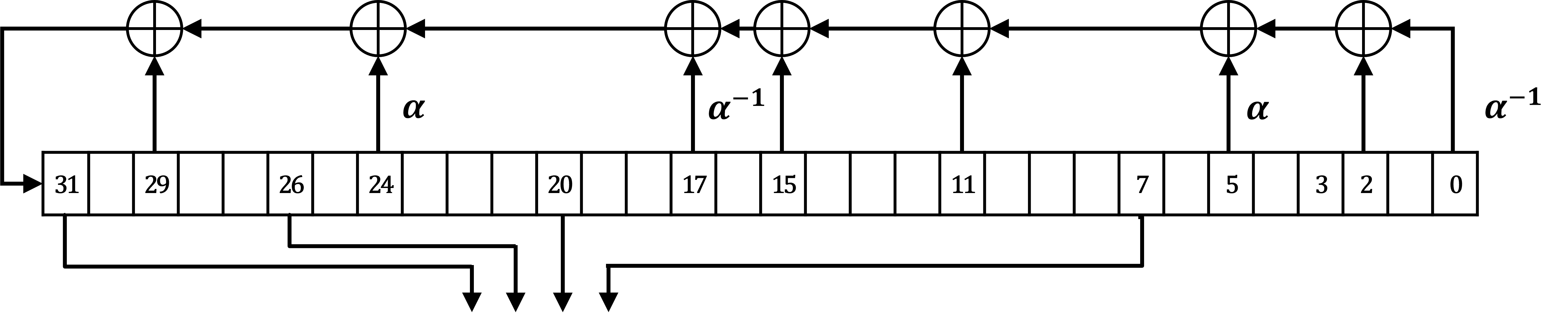


图 1-3 线性反馈移位寄存器结构示意图

**1.1.2 非线性函数**

非线性函数结构如下图所示：

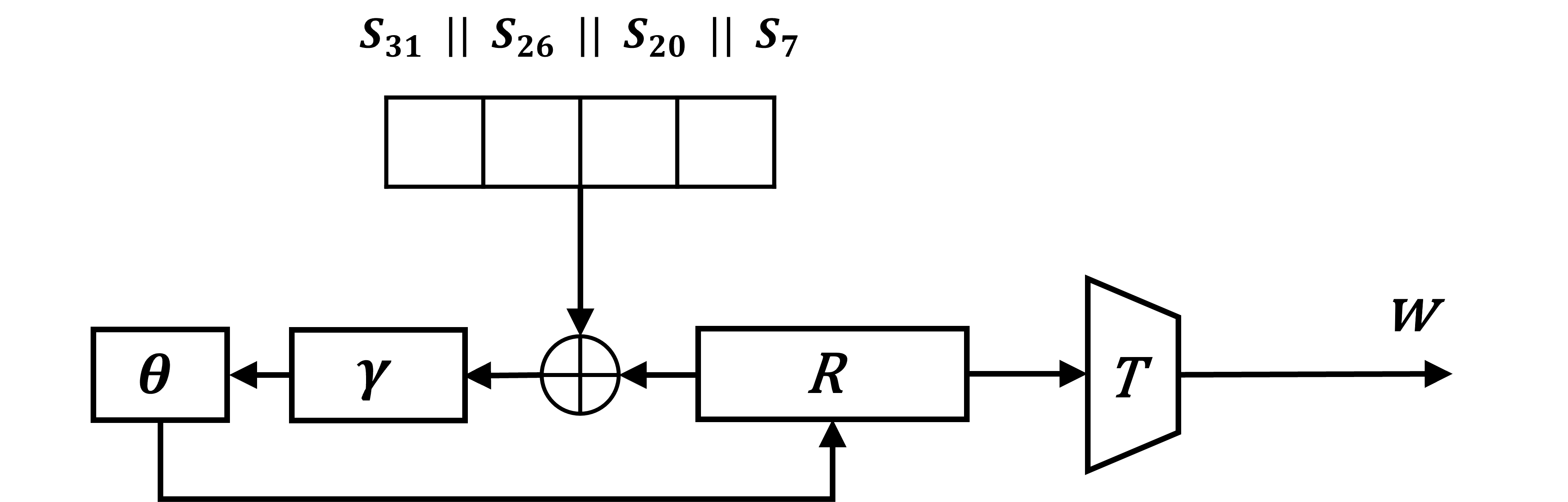


图 1-4 非线性函数结构示意图

非线性函数的功能是将线性反馈移位寄存器的比特的输出截断成比特，非线性函数拥有一个比特的记忆单元，线性反馈移位寄存器的个存储单元容作为非线性函数的输入，经过截断函数截断之后，最后输出一个比特的值，截断函数是将记忆单元中输出的比特的输出截取其前位，并将其输出，剩余的位并不保留。具体流程如下：

**1.1.3**

算法是一种基于字节操作的混合型带记忆逻辑算法，对输入的字节序列与自身的内部记忆单元进行叠加和搅拌等操作，输出一个具有更优统计性质的字节序列。叠加和混淆是密码算法设计中常用的两种操作，具有实现代价小、运行速度快等特点。

由于算法具有这些优越性，面向字节的序列密码算法中使用了算法作为主要的组件

算法的设计者认为，该算法在抗击相关攻击、分别征服攻击、相关密钥攻击和区分攻击等方面都有很好的性能。此外，还具备如下性质：

（1）平衡性

将算法的记忆单元以及输入字节看作上独立均匀分布的随机变量。称是平衡的，如果对任意都有成立。

（2）状态更新和产生输出字节的过程可以用代数（布尔）方程描述

算法的状态更新的过程可以表示成个次方程、个次的布尔方程。同时引入个中间变元。算法的输出过程可以表示为个次方程和个线性方程。引入了个中间变量。

（3）具有周期性的输入序列对应的输出序列表现出一定的线性关系

假设算法的输入字节序列是周期为的周期序列，那么其输出字节序列有一组线性关系：对任意整数，且这组线性关系成立的概率不小于。

的功能是将上一步非线性函数的输出进一步加密，它包含个字节的存储单元，它的输出与的当前时刻的，进行异或运算得到结果为最终的密钥流。详细流程介绍如下：

算法每经过一轮的初始化便更新中的一个记忆单元，更新的单元则是由截断函数经过上述流程而确定的。

**1.2 算法的运行**

算法的运行过程分为初始化阶段和运行阶段。

初始化阶段是将初始密钥和初始向量进行充分的隐藏，从而使算法攻击者不易获取初始密钥及初始向量的信息。算法在运行阶段便开始向外输出密钥流，用算法输出的密钥流与明文进行逐位异或运算，由此便得到经过加密好了的密文。

算法详细介绍如下：

**1.2.1 初始化阶段**

算法的初始化阶段分为两步，第一步是将比特的初始向量和比特的初始密钥分别添加到线性反馈移位寄存器和中，并将寄存器的状态置，算法的输出结果直接参与线性反馈移位寄存器的更新运算。

具体过程如下：

第二步是继第一步之后空运行轮，即不输出结果，而是的输出直接参与线性反馈移位寄存器的反馈计算。

令为时刻的输出，那么初始化状态下线性反馈移位寄存器的更新单元可以用如下公式表示：

序列密码算法初始化状态的结构如下图所示：

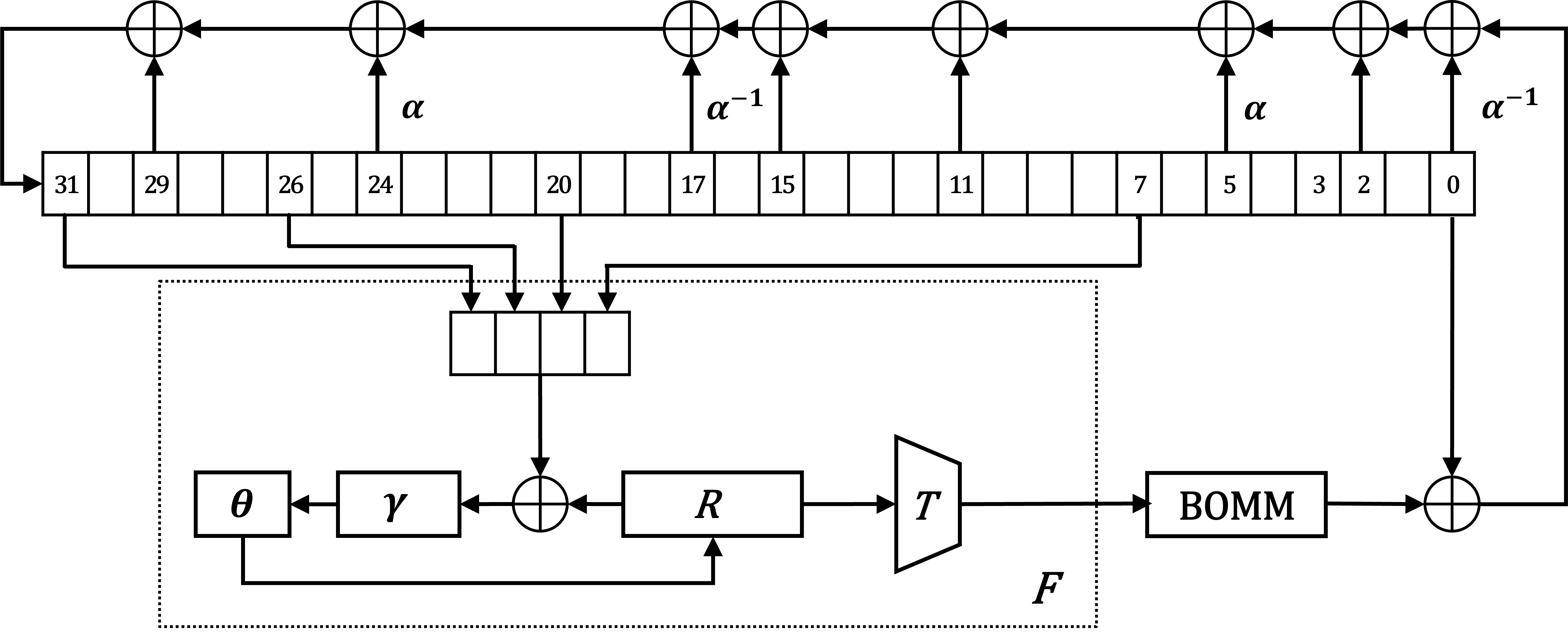


图 1-5 算法初始化结构图

**1.2.2 运行阶段**

在经过轮初始化之后，算法便进入算法运行阶段。在初始化阶段，算法的线性反馈移位寄存器记忆单元和算法记忆单元便得到了一定的更新替换。经过初始化阶段的反复更新后，算法达到了一定的安全性，之后算法便进入密钥流输出阶段，即正常的运行阶段。

算法的运行阶段即为密钥流的生成阶段，此时的输出不再参与线性反馈移位寄存器的反馈计算，而是直接与进行逐位的异或运算，从而产生密钥流，即：

在输出了密钥流之后，将密钥流与明文再进行简单的逐位异或运算，由此便得到了加密好了的密文。

解密端接到密文之后，运用算法的输出密钥流与得到的密文再次进行逐位的异或运算，由此便得到了明文。