(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101399627 B
(45)授权公告日 2012.08.29

(21)申请号 200810171686.5

(22)申请日 2008.10.23

(66)本国优先权数据
200810223214.X 2008.09.27 CN

(73)专利权人 北京数字太和科技有限责任公司
地址 100083 北京市海淀区花园路2号牡丹创业楼三层

(72)发明人 王兴军 闫峰冰 雷大明 陈晨 胡坚珉 梅红兵

(74)专利代理机构 北京德琦知识产权代理有限公司 11018
代理人 宋志强 麻海明

(51)Int.Cl.
H04J 3/06(2006.01)
H04L 7/00(2006.01)
H04L 9/18(2006.01)
H04L 29/06(2006.01)

(56)对比文件
CN 101002420 A,2007.07.18,
CN 1849774 A,2006.10.18,
CN 1784899 A,2006.06.07,
CN 101213839 A,2008.07.02,
CN 1993920 A,2007.07.04,
US 2008019504 A1,2008.01.24,
WO 0163832 A1,2001.08.30,
CN 101030849 A,2007.09.05,

审查员 郑文潇

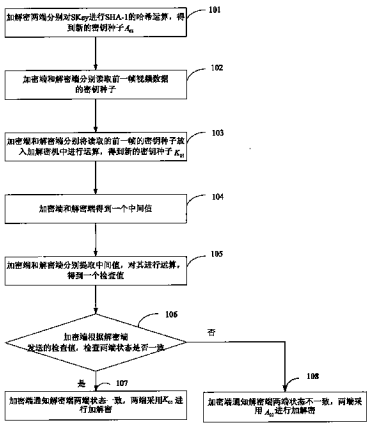
权利要求书 2 页 说明书 7 页 附图 2 页

(54)发明名称

一种同步恢复的方法和系统

(57)摘要

本发明公开了一种同步恢复的方法和系统，加解密两端预先生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别生成表示自身状态的检查值，如果两端生成的检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。本发明解决了加密过程中加解密端因为噪声等的原因造成的两端生成密钥不一致的问题，且在传输过程中对密钥种子的不断更新也提高了系统的安全性。



CN 101399627 B

1. 一种同步恢复的方法，其特征在于：生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别提取其内部产生的能表征加密状态和解密状态的值，将该提取的值直接作为表示自身状态的检查值，或者对该提取的值进行设定算法的运算后，将运算得到的结果作为表示自身状态的检查值，如果两端生成的检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。

2. 如权利要求 1 所述的方法，其特征在于：触发条件发生包括：每传输完一个预先指定长度的中等数据帧，或者，接收到用户输入的同步恢复指令，或者，加密端设备检测到自身的加密处理过程出现异常。

3. 如权利要求 1 所述的方法，其特征在于：对于每一次同步恢复分别生成该次同步恢复使用的密钥种子；

所述生成同步恢复时使用的密钥种子的步骤包括：在生成第一次同步恢复使用的密钥种子时，对初始密钥种子 SKey 进行运算，取运算结果的指定位数作为第一次同步恢复时使用的密钥种子；

在生成后续同步恢复使用的密钥种子时，对上一次同步恢复使用的密钥种子进行运算，取运算结果的指定位数作为本次同步恢复时使用的密钥种子。

4. 如权利要求 3 所述的方法，其特征在于：所述能表征加密状态和解密状态的值为：加密端和解密端中钟控线性反馈移位寄存器 LFSR_CC 的输出值的指定位数，或者，为钟控线性反馈移位寄存器 LFSR_CC 在运行过程中其内部寄存器的传递值的指定位数，或者，为加密端和解密端中序列加扰模块的输出值的指定位数，或者，为钟控线性反馈移位寄存器 LFSR_CC 的值的指定位数与序列加扰模块输出值的指定位数的结合；

所述 LFSR_CC 和所述序列加扰模块是加解密两端密钥种子的生成模块。

5. 如权利要求 4 所述的方法，其特征在于：所述设定算法为哈希算法，或为与运算，或者为或运算；

其中，当为或运算时，所述对该值进行设定算法的运算包括：将提取出的值与加密端和解密端中序列加扰模块的输出值的指定位数进行异或得到一个值，再将该值分为 N 份，并将这 N 份做异或运算，将得到的最后一个值作为检查值，其中 N 为自然数。

6. 如权利要求 1 至 5 中任意一项所述的方法，其特征在于：所述方法进一步包括：加密端每向解密端传输一个指定长度的短数据帧时，加密端和解密端进行一次自同步，该自同步包括：

a、加密端和解密端分别读取前一指定长度短数据帧加解密使用的密钥种子；

b、加密端和解密端分别将读取的前一指定长度短数据帧加解密使用的密钥种子放入加解密机中，进行运算，得到一个新的密钥种子；

c、加密端和解密端采用新的密钥种子作为当前指定长度短数据帧加解密使用的密钥种子。

7. 如权利要求 1 至 5 中任意一项所述的方法，其特征在于：所述方法进一步包括：加密端每向解密端传输一个指定长度的长数据帧时，加密端和解密端进行一次再同步，该再同步包括：

加密端和解密端之间进行认证，协商出一个密钥种子，作为后续数据帧加解密使用的

密钥种子。

8. 如权利要求 7 所述的方法，其特征在于：该方法进一步包括：加密端和解密端在进行再同步后，对该再同步过程中协商出的密钥种子进行运算，将运算结果强制作为下次同步恢复时使用的密钥种子。

9. 一种同步恢复的系统，所述系统包括：加密端、解密端，其特征在于：所述加密端和解密端分别生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别提取其内部产生的能表征加密状态和解密状态的值，将该提取的值直接作为表示自身状态的检查值，或者对该提取的值进行设定算法的运算后，将运算得到的结果作为表示自身状态的检查值，并对两端生成的检查值进行检查，如果检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。

10. 如权利要求 9 所述的系统，其特征在于：

所述加密端和解密端对初始密钥种子 SKey 进行运算，取运算结果的指定位数作为第一次同步恢复使用的密钥种子；

所述加密端和解密端对上一次同步恢复使用的密钥种子进行运算，取运算结果的指定位数作为第一次之后同步恢复使用的密钥种子。

11. 如权利要求 9 至 10 中任意一项所述的系统，其特征在于：所述加密端每向解密端传输一个指定长度的短数据帧时，加密端和解密端分别读取前一指定长度的短数据帧加解密使用的密钥种子，并对其进行运算，将得到的新的密钥种子作为下一指定长度短数据帧加解密使用的密钥种子；

和 / 或，所述加密端每向解密端传输一个指定长度的长数据帧时，加密端和解密端之间进行认证，协商出一个密钥种子，作为后续数据帧加解密使用的密钥种子。

一种同步恢复的方法和系统

技术领域

[0001] 本发明涉及加密领域，特别是一种在加密过程中同步恢复的方法和系统。

背景技术

[0002] 目前，在流加密过程中，通常是加密端设备和解密端设备经过认证确认了对方的合法性之后，加密端和解密端生成一致的密钥种子，用以给所要保护的内容加解密。但由于噪声等的问题，容易造成加解密两端利用密钥种子同时生成的密钥不一致，导致解密端无法正确解密内容。对于正常的加密解密过程，如果这种情况经常出现，会造成密钥不同步，从而造成数据的失步，影响加解密两端正常的工作。而在目前的流加密过程中，并没有涉及到对加解密所使用的密钥进行同步恢复的功能，所以，在两边密钥不一致的情况下，无法检测出这种错误，更无法恢复这种错误，结果会造成解密数据错误，无法解密。

发明内容

[0003] 有鉴于此，本发明的主要目的在于提供一种同步恢复的方法，使其能在加解密过程中及时发现加解密两端失去同步的状态，并尽快使之恢复同步。

[0004] 本发明的另一目的在于提供一种同步恢复的系统，使其能在加解密过程中及时发现加解密两端失去同步的状态，并尽快恢复同步。

[0005] 为达到上述目的，本发明的技术方案具体是这样实现的：

[0006] 一种同步恢复的方法，加解密两端生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别提取其内部产生的能表征加密状态和解密状态的值，将该提取的值直接作为表示自身状态的检查值，或者对该提取的值进行设定算法的运算后，将运算得到的结果作为表示自身状态的检查值，如果两端生成的检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。

[0007] 触发条件发生包括：每传输完一个预先指定长度的中等数据帧，或者，接收到用户输入的同步恢复指令，或者，加密端设备检测到自身的加密处理过程出现异常。

[0008] 对于每一次同步恢复分别生成该次同步恢复使用的密钥种子；

[0009] 所述生成同步恢复时使用的密钥种子的步骤包括：在生成第一次同步恢复使用的密钥种子时，对初始密钥种子 SKey 进行运算，取运算结果的指定位数得到第一次同步恢复时使用的密钥种子；

[0010] 在生成后续同步恢复使用的密钥种子时，对上一次同步恢复使用的密钥种子进行运算，取运算结果的指定位数得到本次同步恢复时使用的密钥种子。

[0011] 所述能表征加密状态和解密状态的值为加密端和解密端中钟控线性反馈移位寄存器 LFSR_CC 的输出值的指定位数，或者，为钟控线性反馈移位寄存器 LFSR_CC 在运行过程中其内部寄存器的一些传递值的指定位数，或者，为加密端和解密端中序列加扰模块的输出值的指定位数，或者，为钟控线性反馈移位寄存器 LFSR_CC 的值的指定位数与序列加扰

模块输出值的指定位数的结合；

[0012] 所述 LFSR_CC 和所述序列加扰模块是加解密两端密钥种子的生成模块。

[0013] 所述设定算法为哈希算法，或为与运算，或者为或运算；

[0014] 其中，当为或运算时，所述对该值进行设定算法的运算包括：将所述中间值与加密端和解密端中钟控线性反馈移位寄存器 LFSR_CC 和 / 或序列加扰模块的输出值的指定位数进行异或得到一个值，再将该值分为 N 份，并将这 N 份做异或运算，将得到的最后一个值作为检查值，其中 N 为自然数。

[0015] 所述方法进一步包括：加密端每向解密端传输一个指定长度的短数据帧时，加密端和解密端进行一次自同步，该自同步包括：

[0016] a、加密端和解密端分别读取前一指定长度短数据帧加解密使用的密钥种子；

[0017] b、加密端和解密端分别将读取的前一指定长度短数据帧加解密使用的密钥种子放入加解密机中，进行运算，得到一个新的密钥种子；

[0018] c、加密端和解密端采用新的密钥种子作为当前指定长度短数据帧加解密使用的密钥种子。

[0019] 所述方法进一步包括：加密端每向解密端传输一个指定长度的长数据帧时，加密端和解密端进行一次再同步，该再同步包括：

[0020] 加密端和解密端之间进行认证，协商出一个密钥种子，作为后续数据帧加解密使用的密钥种子。

[0021] 该方法进一步包括：加密端和解密端在进行再同步后，对该再同步过程中协商出的密钥种子进行运算，将运算结果强制作为下次同步恢复时使用的密钥种子。

[0022] 一种同步恢复的系统，包括：加密端、解密端，所述加密端和解密端生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别提取其内部产生的能表征加密状态和解密状态的值，将该提取的值直接作为表示自身状态的检查值，或者对该提取的值进行设定算法的运算后，将运算得到的结果作为表示自身状态的检查值，并对两端生成的检查值进行检查，如果检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。

[0023] 所述加密端和解密端对初始密钥种子 SKey 进行运算，取运算结果的指定位数作为第一次同步恢复使用的密钥种子；

[0024] 所述加密端和解密端对上一次同步恢复使用的密钥种子进行运算，取运算结果的指定位数作为第一次之后同步恢复使用的密钥种子。

[0025] 所述加密端每向解密端传输一个指定长度的短数据帧时，加密端和解密端分别读取前一指定长度的短数据帧加解密使用的密钥种子，并对其进行运算，将得到的新的密钥种子作为下一指定长度短数据帧加解密使用的密钥种子；

[0026] 和 / 或，所述加密端每向解密端传输一个指定长度的长数据帧时，加密端和解密端之间进行认证，协商出一个密钥种子，作为后续数据帧加解密使用的密钥种子。

[0027] 由此可见，本发明通过传输一定的数据帧之后，对加解密两端进行状态的检查，能够及时得知两端状态是否同步，并且在失步之后及时生成新的密钥种子，使之恢复同步。在本发明的较佳实施例中，能够传输每一个短数据帧之后进行自同步，及时保证两端状态一

致，同时密钥种子能够及时更改，减少了被窃取之后破解系统的可能性，增加了安全性；每传输一个长数据帧之后，周期性的强制重新同步能够进一步提高安全性；并且本发明具有很好的兼容性，可以在多种快速流加密算法中使用。

附图说明

[0028] 图 1 为本发明实施例中中等数据帧传输时同步恢复流程图；

[0029] 图 2 为本发明实施例中短数据帧传输时自同步流程图。

具体实施方式

[0030] 为使本发明的目的、技术方案、及优点更加清楚明白，以下参照附图并举实施例，对本发明做进一步详细说明。

[0031] 本发明提出了一种同步恢复方法，该方法的核心思想是，生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别生成表示自身状态的检查值，如果两端生成的检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。这样，加解密两端能够及时得知两端失去同步的状态，并尽快恢复同步，从而解决了由于噪声等造成加解密两端生成的密钥不一致，导致解密端无法正确解密内容的问题。

[0032] 相应地，本发明还提出了一种同步恢复的系统，该系统包括加密端和解密端，加密端和解密端分别生成同步恢复时使用的密钥种子，该同步恢复时使用的密钥种子与当前加解密利用的密钥种子不同；在触发条件发生时，加密端和解密端分别生成表示自身状态的检查值，并对两端生成的检查值进行检查，如果检查值不一致，则加密端和解密端利用同步恢复时使用的密钥种子进行后续数据的加解密处理。该系统中，加解密两端能够及时得知两端失去同步的状态，并尽快恢复同步，从而解决了由于噪声等造成加解密两端生成的密钥不一致，导致解密端无法正确解密内容的问题。

[0033] 其中，较佳地，上述触发条件发生可以举例为：每传输完一个预先指定长度的中等数据帧，或者，接收到用户输入的同步恢复指令，或者，加密端检测到自身的加密处理过程出现异常等等。

[0034] 另外，为了进一步优化本发明的技术方案，从而对加密过程进行进一步地安全保证处理，在本发明中，还可以进一步包括自同步过程，该自同步过程包括：在每传输一个短数据帧后，加密端和解密端分别利用自身重新生成的一致的密钥种子来对数据进行加解密。另外，本发明中，还可以进一步包括再同步的过程，该再同步的过程包括：在每传输一个长数据帧后，加密端和解密端重新进行认证，并重新协商出双方使用的密钥种子。

[0035] 在以上描述中，短数据帧、中等数据帧和长数据帧的长度可以根据实际业务需要和特点而灵活地指定。比如，在数字电视领域中，将一行数据帧指定为一个短数据帧，将一帧数据指定为一个中等数据帧，将多帧如三帧或四帧数据指定为一个长数据帧。当然，这只是一些举例，在数字电视领域或其他领域，可以根据需要对短数据帧、中等数据帧和长数据帧的长度进行任意指定。

[0036] 本发明提出的同步恢复方法的过程可以与上述自同步过程和再同步过程中的任意一个过程组合使用，或者与自同步过程和再同步过程同时组合使用。

[0037] 本发明可以应用于任何一种加密领域中，比如，需要对普通流加密的领 域，或者需要对快速流加密的领域等。其中，需要对快速流加密的领域可以举例为数字电视、视频共享、多方在线游戏等领域。

[0038] 下面以数字电视领域中的快速流加密过程为例，分别对同步恢复过程、自同步过程和再同步过程的方法以及本发明系统中各个设备的功能及连接关系进行详细说明。

[0039] 首先，与现有技术中相同的过程是：在加密端设备与解密端设备初始连接完成后，加密端设备和解密端设备之间经过认证确认了对方的合法性，之后，加密端设备和解密端设备生成一致的初始密钥种子 SKey，然后使用该密钥种子对传输的数据进行加解密处理。

[0040] 数据在传输了一定量之后，加解密两端需要对双方状态进行检查，以检查两端是否丧失同步，如果丧失同步，需要做相应处理，使两端状态恢复同步。具体工作流程如下：

[0041] 步骤 101：加密端和解密端预先分别对 SKey 进行安全散列计算 (SHA-1) 的哈希运算，并取运算结果的指定位数作为同步恢复时使用的密钥种子，为便于以后描述，将该同步恢复时使用的密钥种子称为 A01。

[0042] 本步骤中，指定位数例如可以取运算结果最高有效位的 128 位或者，取运算结果最低或中间有效位的 128 位等。

[0043] 另外，在本步骤中，对 SKey 进行的是安全散列计算 (SHA-1) 的哈希运算，在实际的业务实现中，也可以采用其他任何可行的算法对 SKey 进行运算。

[0044] 另外，在本步骤中，取运算结果的指定位数作为同步恢复时使用的密钥种子，在本发明的其他实施例中，也可以直接将运算结果作为同步恢复时使用的密钥种子，或者采用其他变换方法，例如移位等方法，来通过运算结果得到同步恢复时使用的密钥种子。

[0045] 步骤 102：加密端每向解密端传输一帧视频数据之后，加密端和解密端分别读取前一帧数据加解密使用的密钥种子，即如果所传为第 N 帧视频数 据，则加密端和解密端分别读取第 N-1 帧视频数据的密钥种子，如果该行视频数据是第一帧数据，则加密端设备和解密端设备读取 SKey。

[0046] 步骤 103：加密端和解密端分别将读取的前一帧的密钥种子放入加解密机中，进行运算，得到加解密数据使用的新的密钥种子，为区别于密钥种子 A_{01}，将该密钥种子表示为 K_{01}。

[0047] 步骤 104：加密端每向解密端传输一帧视频数据之后，加密端和解密端中的加解密机分别得到一个中间值。

[0048] 在本步骤中，所得到的中间值代表了加密端和解密端当前加解密的自身运行状态。

[0049] 步骤 105：加密端和解密端分别提取中间值，并对其进行运算得到一个检查值。

[0050] 总的说来，在本发明中，在生成检查值时，具体原则是：加密端和解密端分别提取其内部产生的能表征加密状态和解密状态的值，并将该值作为中间值，可以将该中间值直接作为两端进行状态检查的检查值，但出于安全性考虑，也可以对该值进行设定算法的运算后，将运算得到的结果作为所述检查值。

[0051] 其中，较佳地，所述能表征加密端状态或解密端状态的值，即中间值可以是加密端和解密端中钟控线性反馈移位寄存器 (LFSR_CC) 模块的值的指定位数，例如可以是该模块最终的输出值的指定位数，也可以是该模块在运行过程中其内部寄存器的一些传递值的指

定位数；所述中间值还可以是加密端和解密端中序列加扰模块的输出值的指定位数或钟控线性反馈移位寄存器（LFSR_CC）的值的指定位数与序列加扰模块输出值的指定位数的相应结合等多种形式。之所以采取钟控线性反馈移位寄存器（LFSR_CC）和/或序列加扰模块的值作为中间值，是因为该两模块原本就是加解密两端密钥种子的生成模块，所以利用其输出值的指定位数作为中间值能够准确反映加解密两端的运行状态。

[0052] 上述对中间值进行设定算法的运算过程可以举例为：将中间值自身进行设定算法的运算，或者，将中间值与钟控线性反馈移位寄存器（LFSR_CC）和/或序列加扰模块运行一定周期后的输出值的指定位数进行设定算法的运算等多种运算方式。

[0053] 上述设定算法可以为任意一种可行的算法，比如为与运算，或者为或运算，或者为哈希运算等等。

[0054] 下面对钟控线性反馈移位寄存器（LFSR_CC）的输出值的指定位数作为中间值，并将该中间值与序列加扰模块的输出值的指定位数进行异或运算从而得到检查值的具体实现方式进行详细说明。

[0055] 在上述步骤104和步骤105中，加解密两端提取钟控线性反馈移位寄存器（LFSR_CC）的输出，并取其输出值的指定位数作为中间值。

[0056] 检查值的计算过程是：得到中间值以后，加解密端中的序列加扰模块继续运行一定周期，同步恢复模块将所得到的中间值与序列加扰模块此时的输出值的指定位数进行异或得到一个值，之后再将该值分为N份（N为自然数），再将这N份互相异或，得到最后一个值，称为检查值。其中，在该步骤中的互相异或，可以是将第1份与第2份进行异或，其结果再与第3份进行异或，以此类推，直到第N份，也可以是将N份进行分组，首先进行组内各份的异或，然后将各组的运算结果再进行异或运算，直至得出最后结果，当然还可以有其他能实现目的的异或方法。

[0057] 下面给出一种更为具体的得到中间值和检查值的实现过程。

[0058] 在现有技术中，加密端和解密端中包括钟控线性反馈移位寄存器（LFSR_CC）模块和序列加扰模块，在加密过程中，密钥种子、中间值以及检测值的生成都由上述模块来完成。

[0059] 在加解密两端得到新的密钥种子K_{01}之后，LFSR_CC模块继续运行34个周期，提取其输出的128位，该值称为中间值。

[0060] 中间值是一个可以表示加解密端运行状态的数值，鉴于系统安全性，需 对中间值进行变换，但变换结果不丧失其代表性，所以，在得到中间值后，可对其做如下变换：序列加扰模块继续运行6个周期，将其输出值的高128位与中间值进行异或运算，得到一个128位的值，然后将该值分成8份，并进行编号，每份16位，将第1份与第2份进行异或，其结果再与第3份进行异或，以此类推，直至到第16份，得到一个16位的值，称之为检查值。

[0061] 需要说明的是，上述步骤104至步骤105中所描述的生成检查值的过程只是本实施例所列举的一种生成检查值的较佳实现过程，在本发明的其他实施例中，也可以利用其他方法来生成能够反映加解密设备的运行状态的检查值，比如，直接将步骤104中得到的中间值的指定位数作为检查值，或者，将步骤104中得到的中间值的指定位数进行运算比如哈希运算后，将运算得到的结果作为检查值等等。

[0062] 步骤106：解密端将其产生的检查值发送给加密端，加密端判断自身得到的检查

值与接收到的检查值是否相同，如果检查结果相同，则执行步骤 107，如果检查结果不同，表明两端状态不一致，则执行步骤 108。

[0063] 本步骤中，如果检查结果相同，则表明加密端设备和解密端设备两端状态一致，也就是说，其使用的密钥种子一致，因此，利用该密钥种子生成的对数据帧进行加解密所使用的密钥是同步的，无需进行同步恢复处理，可以继续进行后续的与现有技术中相同处理的数据加解密过程，如果检查结果不相同，则表明加密端设备和解密端设备两端状态不一致，也就是说，其使用的密钥种子不一致，因此，利用该密钥种子生成的对数据帧进行加解密所使用的密钥是不同步的，需要进行同步恢复处理，即两端需要使用新的一致的密钥种子，执行步骤 108。

[0064] 步骤 107 ：加密端通知解密端两端状态一致，并且两端采用 K_{01} 进行加解密，结束当前流程。

[0065] 步骤 108 ：加密端通知解密端两端状态不一致，加密端和解密端采用密钥种子 A_{01} 进行加解密。

[0066] 本步骤中，加密端和解密端采用密钥种子 A_{01} 进行加解密的过程是指利用密钥种子 A_{01} 作为后续数据中第一帧数据使用的初始密钥种子来执行步骤 102 和步骤 103 所示原理的数据加解密过程，即利用 A_{01} 来对后续第一帧数据进行加解密，然后利用 A_{01} 进行运算后得到的结果对后续第二帧数据进行加解密，以此类推。

[0067] 此后，可以对 A_{01} 进行 SHA-1 的哈希运算，并取运算结果的指定位数，例如可以取其最高有效位的 128 位等，并将其称之为 A_{02}，作为下次同步恢复所使用的加解密的密钥种子。

[0068] 需要说明的是，在上述步骤 106 至步骤 108 中，是由解密端将自身生成的检查值发送给加密端，由加密端判断两个检查值是否相同，并通知解密端状态是否一致。在本发明的其他实施例中，也可以由加密端将自身生成的检查值发送给解密端，由解密端判断两个检查值是否相同，并通知加密端状态是否一致，从而触发由加密端和解密端在状态一致时使用 K_{01} 且在状态不一致时使用 A_{01} 作为密钥种子。

[0069] 至此，实现了视频数据加密过程中加解密两端状态检测以及恢复同步的过程。

[0070] 在实际的视频数据传输过程中，为进一步提高系统的安全性和灵活性，在加解密两端每传输一个短视频数据时，加解密两端可以直接进行自同步而无需做检测处理。

[0071] 图 2 是短数据帧传输时自同步流程图，如图所示，具体步骤如下 ：

[0072] 步骤 201 ：加密端每向解密端传输一行视频数据之后，加密端和解密端分别读取前一行的密钥种子，即如果所传为第 N 行视频数据，则加密端和解密端分别读取第 N-1 行视频数据的密钥种子，如果该行视频数据是第一行数据，则加密端和解密端读取 SKey。

[0073] 步骤 202 ：加密端和解密端分别将读取的前一行密钥种子放入加解密机中，进行运算，得到一个新的密钥种子。

[0074] 步骤 203 ：两端采用新的密钥种子作为下一行视频数据的加密种子。

[0075] 至此，实现了视频数据加密过程中短数据帧传输时的自同步。

[0076] 同样，当加解密两端已传输了一个较长视频数据后，加解密两端也无需进行状态检测而直接进行两端的强制再同步，以提高系统的安全性。

[0077] 加解密两端进行再同步的具体操作如下 ：

[0078] 加密端每向解密端传输多帧如三帧视频数据后，需执行 ：加密端和解密端进行认

证，协商出一个密钥种子，作为后续数据帧加解密的密钥种子。

[0079] 至此，完成了视频数据加密过程中长数据帧传输时的再同步。

[0080] 在完成该再同步过程后，出于安全性考虑，对再同步过程中所生成的新的密钥种子做 SHA-1 的哈希运算，得到另外一个密钥种子 A_{11}，该密钥种子的功能如下：当加解密两端已进行了 M-1 次同步恢复 (M 为一个大于 1 的自然数)，并且已生成了下次同步恢复所要使用的密钥种子 A_{0M}，此时，由于加解密两端传输的数据量已达到了一个指定长度的长数据帧，所以加解密两端要进行一次再同步，随着再同步过程的结束，也产生了一个密钥种子 A_{11}，那么该密钥种子将在下次的同步恢复过程中强制性取代 A_{0M} 作为同步恢复的密钥种子。

[0081] 由以上实施例可以得出：本发明通过传输一定的数据帧之后，对加解密两端进行状态的检查，能够及时得知两端状态是否同步，并且在失步之后及时生成新的密钥种子，使之恢复同步。在本发明的较佳实施例中，能够传输每一个短数据帧之后进行自同步，及时保证两端状态一致，同时密钥种子能够及时更改，减少了被窃取之后破解系统的可能性，增加了安全性；每传输一个场数据帧之后，周期性的强制重新同步能够进一步提高安全性；并且本发明具有很好的兼容性，可以在多种快速流加密算法中使用。

[0082] 总之，以上所述仅为本发明的较佳实施例而已，并非用于限定本发明的保护范围，凡在本发明的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

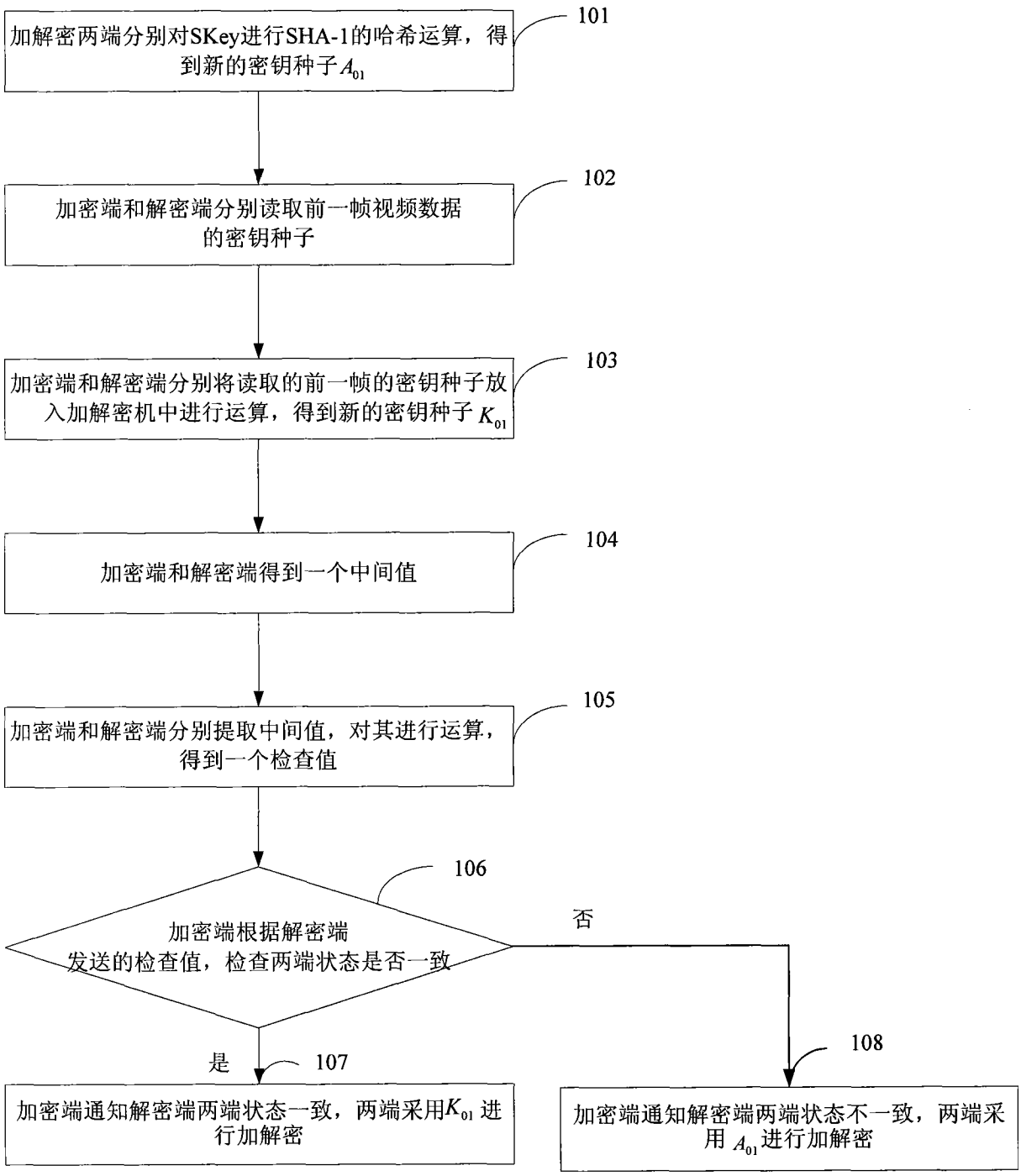

加解密两端分别对SKey进行SHA-1的哈希运算，得到新的密钥种子A_{01}
101
加密端和解密端分别读取前一帧视频数据的密钥种子
102
加密端和解密端分别将读取的前一帧的密钥种子放入加解密机中进行运算，得到新的密钥种子K_{01}
103
加密端和解密端得到一个中间值
104
加密端和解密端分别提取中间值，对其进行运算，得到一个检查值
105
加密端根据解密端发送的检查值，检查两端状态是否一致
106
否
是
107
加密端通知解密端两端状态一致，两端采用K_{01}进行加解密
加密端通知解密端两端状态不一致，两端采用A_{01}进行加解密
108

图 1

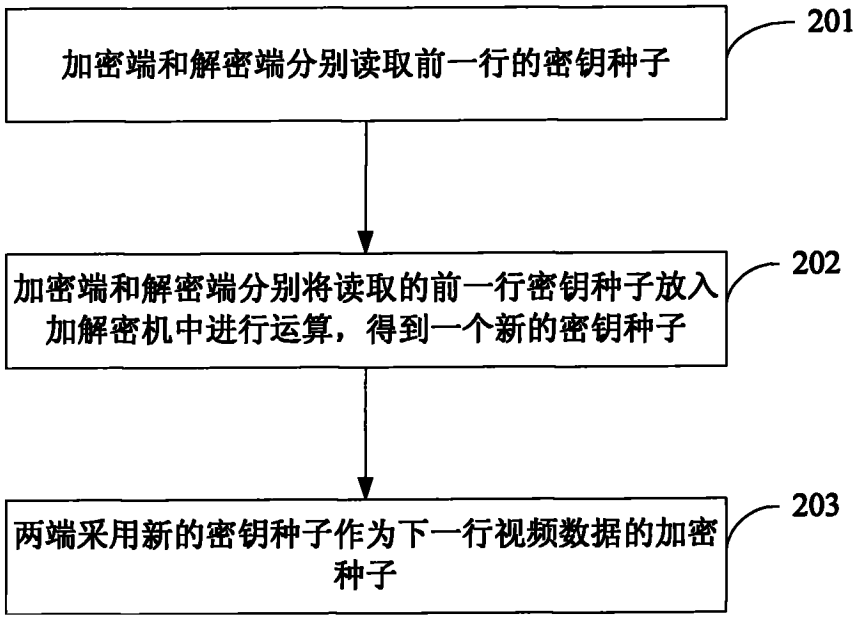

加密端和解密端分别读取前一行的密钥种子
201
加密端和解密端分别将读取的前一行密钥种子放入加解密机中进行运算，得到一个新的密钥种子
202
两端采用新的密钥种子作为下一行视频数据的加密种子
203

图 2