(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 116780778 B
(45)授权公告日 2024.07.09

(21)申请号 202310814395.8
(22)申请日 2023.07.05
(65)同一申请的已公布的文献号
申请公布号 CN 116780778 A
(43)申请公布日 2023.09.19
(73)专利权人 西安天能软件科技有限责任公司
地址 710065 陕西省西安市高新区沣惠南路16号泰华金贸国际9号楼2303室
(72)发明人 刘晓玉 郭志雷 刘敏 张月斌 王君杰
(74)专利代理机构 北京鼎云升知识产权代理事务所(普通合伙) 11495
专利代理师 李梓栋
(51)Int.Cl.
H02J 13/00(2006.01)
H02H 7/22(2006.01)
G06F 8/30(2018.01)
(56)对比文件
CN 108600183 A,2018.09.28
审查员 李瑞梅

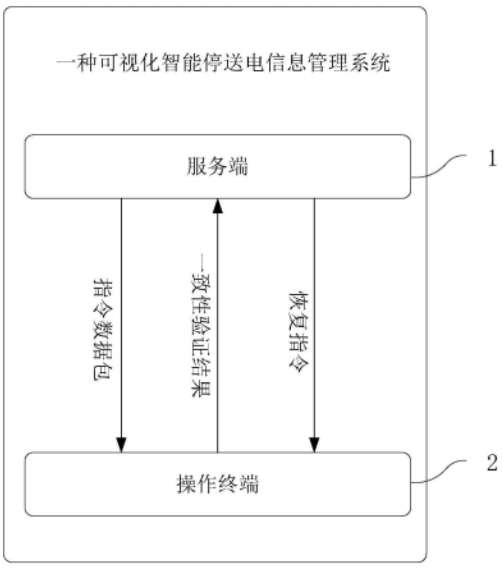
权利要求书4页 说明书20页 附图7页

(54)发明名称

一种能量隔离处理方法及可视化智能停送电信息管理系统

(57)摘要

本发明公开一种能量隔离处理方法及可视化智能停送电信息管理系统,涉及输变电技术领域。本发明包括,接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员;根据目标电气开关柜以及对应的目标操作人员生成指令数据包,指令数据包内包括目标电气开关柜和目标操作人员的验证信息;向目标操作人员对应的操作终端发送指令数据包;接收操作终端返回的一致性验证结果;判断是否通过身份验证和设备验证;若是,则将目标电器开关柜标记为能量隔离状态,若否,则向操作终端发送恢复指令并结束步骤。本发明通过对操作人员和设备进行多重验证,在避免误操作的前提下有效提高操作效率。



CN 116780778 B

1.一种能量隔离处理方法,其特征在于,包括,

接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员;

根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包,所述指令数据包内包括所述目标电气开关柜和目标操作人员的验证信息;

向所述目标操作人员对应的操作终端发送所述指令数据包;

接收所述操作终端返回的一致性验证结果;

判断是否通过身份验证和设备验证;

若是,则将所述目标电器开关柜标记为能量隔离状态,

若否,则向所述操作终端发送恢复指令并结束步骤;

其中,

所述根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包的步骤,包括,

根据所述目标电气开关柜的编号生成得到设备信息;

将所述设备验证指令和所述设备信息组合得到设备验证信息;

将目标操作人员的生物信息进行特征提取得到目标操作人员的身份特征;

根据目标操作人员的所述身份特征生成身份验证信息,并对所述设备验证信息进行加密得到设备验证加密信息;

根据所述目标电气开关柜的编号、所述设备验证加密信息和所述身份验证信息生成得到所述指令数据包;

所述根据目标操作人员的所述身份特征生成身份验证信息,并对所述设备验证信息进行加密得到设备验证加密信息的步骤,包括,

按照生成算法获取目标操作人员的所述身份特征的不可逆的数字摘要;

将目标操作人员的所述身份特征的数字摘要拆分为显性数字段和隐性数字段;

使用所述隐性数字段作为密钥对所述设备验证信息进行加密得到设备验证加密信息;

将所述生成算法和所述显性数字段组合得到身份验证信息;

所述按照生成算法获取目标操作人员的所述身份特征的不可逆的数字摘要的步骤,包括,

根据目标操作人员的所述身份特征得到目标操作人员的所述身份特征在多个特征维度上的数值分布;

对于每个特征维度,根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间;

将每个特征维度和对应的常规分布区间的数据信息组合为一个特征数据组;

将每个特征数据组根据对应的常规分布区间分配至显性数据块或隐性数据块;

按照生成算法获取显性数据块和所述隐性数据块的不可逆的数字摘要,组合得到所述身份特征的数字摘要;

所述对于每个特征维度,根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间的步骤,包括,

对于每个特征维度,

根据对应的数值分布获取对应数值的顺序排序序列作为数值序列;

获取所述数值序列内全部相邻的两个数值的差值；

计算获取所述数值序列内全部相邻的两个数值的差值的均值作为筛选标定值；

将所述数值序列内与相邻数值的差值小于所述筛选标定值的数值作为异常值进行剔除；

获取剔除异常值之后的所述数值序列内数值的分布区间作为目标操作人员在该特征维度的常规分布区间。

2.根据权利要求1所述的方法,其特征在于,所述将目标操作人员的所述身份特征的数字摘要拆分为显性数字段和隐性数字段的步骤,包括,

将所述显性数据块对应的数字摘要作为显性数字段；

将所述隐性数据块对应的数字摘要作为隐性数字段。

3.根据权利要求1所述的方法,其特征在于,所述方法还包括,

接收由所述操作终端发送的所述目标电器开关柜的状态锁的锁具信息；

将所述目标电器开关柜的编号与对应的所述状态锁的锁具信息进行关联,得到设备锁具关联关系；

接收所述操作终端发送的锁具关联查询请求,其中,所述锁具关联查询请求内包含待确认的电器开关柜的编号与所述状态锁的锁具信息；

根据所述锁具关联查询指令查询所述设备锁具关联关系,判断所述锁具关联查询请求内包含待确认的电器开关柜的编号与所述状态锁的锁具信息是否具有关联性；

若是,则向所述操作终端返回具有关联性；

若否,则向所述操作终端返回不具有关联性。

4.一种能量隔离处理方法,其特征在于,包括,

接收服务端发送的权利要求1至3任一项所述一种能量隔离处理方法中的指令数据包；

根据所述指令数据包内的目标电气开关柜的编号寻找到目标电气开关柜；

获取操作人员的身份特征；

根据所述指令数据包内的身份验证信息对操作人员的身份特征进行验证；

若未通过,则结束步骤或提醒重试；

若通过,则对目标电器开关柜执行能量隔离操作；

在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息,其中,所述隐藏设备信息包括位于所述目标电器开关柜内部的二维码标签；

根据指令数据包对所述隐藏设备信息进行验证；

若通过,则向所述服务端返回通过身份验证和设备验证的一致性验证结果；

若未通过,则向所述服务端返回身份验证和设备验证的不一致性验证结果。

5.根据权利要求4所述的方法,其特征在于,所述根据指令数据包对所述隐藏设备信息进行验证的步骤,包括,

根据所述指令数据包解析得到生成算法、显性数字段以及设备验证加密信息；

利用所述生成算法获取操作人员的身份特征的数字摘要；

根据所述数字摘要以及显性数字段得到隐性数字段；

根据所述隐性数字段作为密钥对所述设备验证加密信息进行解密得到所述设备验证信息内的设备验证指令和设备信息；

执行所述设备验证指令对所述设备验证信息内所述设备信息和所述隐藏设备信息进行比对；

若相同,则设备验证一致；

若不相同,则设备验证不一致。

6.一种可视化智能停送电信息管理系统,其特征在于,包括,

服务端,用于接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员；

根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包,所述指令数据包内包括所述目标电气开关柜和目标操作人员的验证信息；

向所述目标操作人员对应的操作终端发送所述指令数据包；

接收所述操作终端返回的一致性验证结果；

判断是否通过身份验证和设备验证；

若是,则将所述目标电器开关柜标记为能量隔离状态,

若否,则向所述操作终端发送恢复指令并结束步骤；

操作终端,用于接收服务端发送的指令数据包；

根据所述指令数据包内的目标电气开关柜的编号寻找到目标电气开关柜；

获取操作人员的身份特征；

根据所述指令数据包内的身份验证信息对操作人员的身份特征进行验证；

若未通过,则结束步骤或提醒重试；

若通过,则对目标电器开关柜执行能量隔离操作；

在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息,其中,所述隐藏设备信息包括位于所述目标电器开关柜内部的二维码标签；

根据指令数据包对所述隐藏设备信息进行验证；

若通过,则向所述服务端返回通过身份验证和设备验证的一致性验证结果；

若未通过,则向所述服务端返回身份验证和设备验证的不一致性验证结果；

其中,

所述根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包的步骤,包括,

根据所述目标电气开关柜的编号生成得到设备信息；

将所述设备验证指令和所述设备信息组合得到设备验证信息；

将目标操作人员的生物信息进行特征提取得到目标操作人员的身份特征；

根据目标操作人员的所述身份特征生成身份验证信息,并对所述设备验证信息进行加密得到设备验证加密信息；

根据所述目标电气开关柜的编号、所述设备验证加密信息和所述身份验证信息生成得到所述指令数据包；

所述根据目标操作人员的所述身份特征生成身份验证信息,并对所述设备验证信息进行加密得到设备验证加密信息的步骤,包括,

按照生成算法获取目标操作人员的所述身份特征的不可逆的数字摘要；

将目标操作人员的所述身份特征的数字摘要拆分为显性数字段和隐性数字段；

使用所述隐性数字段作为密钥对所述设备验证信息进行加密得到设备验证加密信息；

将所述生成算法和所述显性数字段组合得到身份验证信息；

所述按照生成算法获取目标操作人员的所述身份特征的不可逆的数字摘要的步骤，包括，

根据目标操作人员的所述身份特征得到目标操作人员的所述身份特征在多个特征维度上的数值分布；

对于每个特征维度，根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间；

将每个特征维度和对应的常规分布区间的数据信息组合为一个特征数据组；

将每个特征数据组根据对应的常规分布区间分配至显性数据块或隐性数据块；

按照生成算法获取显性数据块和所述隐性数据块的不可逆的数字摘要，组合得到所述身份特征的数字摘要；

所述对于每个特征维度，根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间的步骤，包括，

对于每个特征维度，

根据对应的数值分布获取对应数值的顺序排序序列作为数值序列；

获取所述数值序列内全部相邻的两个数值的差值；

计算获取所述数值序列内全部相邻的两个数值的差值的均值作为筛选标定值；

将所述数值序列内与相邻数值的差值小于所述筛选标定值的数值作为异常值进行剔除；

获取剔除异常值之后的所述数值序列内数值的分布区间作为目标操作人员在该特征维度的常规分布区间。

一种能量隔离处理方法及可视化智能停送电信息管理系统

技术领域

[0001] 本发明属于输变电技术领域,特别是涉及一种能量隔离处理方法及可视化智能停送电信息管理系统。

背景技术

[0002] 随着工业化和现代化的发展,能源的管理和利用成为了至关重要的问题。尤其是在电力系统中,准确有效地管理和调度电力资源对于保障供电的可靠性和安全性具有重要意义。在电力系统运营中,停送电作业是常见的一种操作,通常涉及到对电气设备进行检修、维护或其他处理,以确保系统的稳定运行。

[0003] 然而,传统的停送电管理方式往往依赖于人工操作和判断,这不仅效率低下,而且可能因为人为错误而导致安全隐患。此外,当电力系统出现故障时,能够迅速并准确地隔离问题区域,以防止故障的进一步扩散,对于保障系统稳定是至关重要的。然而,现有的能量隔离处理方法在故障检测、隔离和信息反馈方面存在不足,不能有效地应对复杂和大规模的电力系统。

[0004] 此外,在传统的停送电信息管理中,信息的展示和交互通常不够直观,导致操作人员难以快速理解和掌握系统的状态。在紧急情况下,这可能会影响到操作人员做出准确判断和快速响应。

发明内容

[0005] 本发明的目的在于提供一种能量隔离处理方法及可视化智能停送电信息管理系统,通过对操作人员和设备进行多重验证,在避免误操作的前提下有效提高操作效率。

[0006] 为解决上述技术问题,本发明是通过以下技术方案实现的:

[0007] 本发明提供一种能量隔离处理方法,包括,

[0008] 接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员;

[0009] 根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包,所述指令数据包内包括所述目标电气开关柜和目标操作人员的验证信息;

[0010] 向所述目标操作人员对应的操作终端发送所述指令数据包;

[0011] 接收所述操作终端返回的一致性验证结果;

[0012] 判断是否通过身份验证和设备验证;

[0013] 若是,则将所述目标电器开关柜标记为能量隔离状态,

[0014] 若否,则向所述操作终端发送恢复指令并结束步骤。

[0015] 本发明还公开了一种能量隔离处理方法,包括,

[0016] 接收服务端发送的指令数据包;

[0017] 根据所述指令数据包内的目标电气开关柜的编号寻找到目标电气开关柜;

[0018] 获取操作人员的身份特征;

[0019] 根据所述指令数据包内的身份验证信息对操作人员的身份特征进行验证；

[0020] 若未通过，则结束步骤或提醒重试；

[0021] 若通过，则对目标电器开关柜执行能量隔离操作；

[0022] 在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息，其中，所述隐藏设备信息包括位于所述目标电器开关柜内部的二维码标签；

[0023] 根据指令数据包对所述隐藏设备信息进行验证；

[0024] 若通过，则向所述服务端返回通过身份验证和设备验证的一致性验证结果；

[0025] 若未通过，则向所述服务端返回身份验证和设备验证的不一致性验证结果。

[0026] 本发明还公开了一种可视化智能停送电信息管理系统，其特征在于，包括，

[0027] 服务端，用于接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员；

[0028] 根据所述目标电气开关柜以及对应的目标操作人员生成指令数据包，所述指令数据包内包括所述目标电气开关柜和目标操作人员的验证信息；

[0029] 向所述目标操作人员对应的操作终端发送所述指令数据包；

[0030] 接收所述操作终端返回的一致性验证结果；

[0031] 判断是否通过身份验证和设备验证；

[0032] 若是，则将所述目标电器开关柜标记为能量隔离状态，

[0033] 若否，则向所述操作终端发送恢复指令并结束步骤；

[0034] 操作终端，用于接收服务端发送的指令数据包；

[0035] 根据所述指令数据包内的目标电气开关柜的编号寻找到目标电气开关柜；

[0036] 获取操作人员的身份特征；

[0037] 根据所述指令数据包内的身份验证信息对操作人员的身份特征进行验证；

[0038] 若未通过，则结束步骤或提醒重试；

[0039] 若通过，则对目标电器开关柜执行能量隔离操作；

[0040] 在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息，其中，所述隐藏设备信息包括位于所述目标电器开关柜内部的二维码标签；

[0041] 根据指令数据包对所述隐藏设备信息进行验证；

[0042] 若通过，则向所述服务端返回通过身份验证和设备验证的一致性验证结果；

[0043] 若未通过，则向所述服务端返回身份验证和设备验证的不一致性验证结果。

[0044] 本发明通过对操作人员和设备进行多重验证，能够在避免误操作的前提下有效提高操作效率。在实施过程中首先接收并解析获取到需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员信息。然后根据这些信息生成指令数据包。接下来将指令数据包发送到目标操作人员对应的操作终端。在接收到操作终端返回的一致性验证结果后进行身份验证和设备验证判断。如果通过验证则将目标电气开关柜标记为能量隔离状态。如果未通过验证则向操作终端发送恢复指令并结束操作步骤。通过以上流程，能够确保在操作过程中避免误操作，并提高操作效率。

[0045] 当然，实施本发明的任一产品并不一定需要同时达到以上所述的所有优点。

附图说明

[0046] 为了更清楚地说明本发明实施例的技术方案,下面将对实施例描述所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0047] 图1为本发明所述一种可视化智能停送电信息管理系统于一实施例的功能单元和信息流向示意图;

[0048] 图2为本发明所述服务端于一实施例的步骤流程示意图;

[0049] 图3为本发明所述操作终端于一实施例的步骤流程示意图;

[0050] 图4为本发明所述步骤S2于一实施例的步骤流程示意图;

[0051] 图5为本发明所述步骤S24于一实施例的步骤流程示意图;

[0052] 图6为本发明所述步骤S241于一实施例的步骤流程示意图;

[0053] 图7为本发明所述步骤S2412于一实施例的步骤流程示意图;

[0054] 图8为本发明所述步骤S15于一实施例的步骤流程示意图。

[0055] 附图中,各标号所代表的部件列表如下:

[0056] 1-服务端,2-操作终端。

具体实施方式

[0057] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其它实施例,都属于本发明保护的范围。

[0058] 为了对电气开关柜的操作进行准确定位,同时也对电气开关柜的状态进行监控,本发明提供以下方案。

[0059] 请参阅图1所示,本发明提供了一种可视化智能停送电信息管理系统,从功能单元上划分可以包括服务端1和操作终端2。服务端1可以位于主控制室,也可以部署在云端。操作终端2由操作人员持有。接下来进行分别介绍。

[0060] 请参阅图2所示,对于服务端1,在具体实施的过程中首先可以执行步骤S1接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人员,这部分信息可以是由电力调配系统产生。接下来可以执行步骤S2根据目标电气开关柜以及对应的目标操作人员生成指令数据包,指令数据包内包括目标电气开关柜和目标操作人员的验证信息。接下来可以执行步骤S3向目标操作人员对应的操作终端发送指令数据包。接下来可以执行步骤S4接收操作终端返回的一致性验证结果。接下来可以执行步骤S5判断是否通过身份验证和设备验证。若是则接下来可以执行步骤S6将目标电器开关柜标记为能量隔离状态,若否则接下来可以执行步骤S7向操作终端发送恢复指令并结束步骤。

[0061] 为了避免对能量隔离状态的电气开关柜进行误接通导致安全隐患,可以对目标电器开关柜进行物理锁定,并将目标电器开关柜的状态锁的锁具信息上传到服务端1。具体而言,首先接收由操作终端发送的目标电器开关柜的状态锁的锁具信息。之后将目标电器开关柜的编号与对应的状态锁的锁具信息进行关联,得到设备锁具关联关系。接下来接收操

作终端发送的锁具关联查询请求,在这其中,锁具关联查询请求内包含待确认的电器开关柜的编号与状态锁的锁具信息。接下来根据锁具关联查询指令查询设备锁具关联关系,判断锁具关联查询请求内包含待确认的电器开关柜的编号与状态锁的锁具信息是否具有关联性。若是则向操作终端返回具有关联性,若否则向操作终端返回不具有关联性。这样即可以避免对能量隔离状态的目标电器开关柜进行误操作。

[0062] 为了对上述的步骤S1至步骤S7的实施过程进行补充说明,提供部分功能模块的源代码,并在注释部分进行对照解释说明。为了符合相关法律法规对建筑物的数据安全要求,对不影响方案实施的部分数据进行脱敏处理,下同。

```
#include <iostream>
#include <string>
```

[0063]

```
// 模拟的指令数据包结构
struct InstructionPacket {
    std::string switchCabinet; // 目标电气开关柜
```

```
        std::string operatorName;   // 目标操作人员
        std::string verificationInfo; // 验证信息
    };
    // 模拟的一致性验证结果
    enum class VerificationResult {
        Passed,
        Failed
    };
    // 模拟向操作终端发送指令数据包的函数
    VerificationResult sendInstructionToTerminal(const InstructionPacket& packet) {
        // 此处可以加入真实的发送逻辑
        std::cout << "Sending instruction to terminal for operator: " << packet.operatorName <<
std::endl;
        return VerificationResult::Passed; // 总是通过验证
    }
    // 模拟向操作终端发送恢复指令的函数
[0064]  void sendRecoveryInstructionToTerminal(const std::string& operatorName) {
        // 此处可以加入真实的发送逻辑
        std::cout << "Sending recovery instruction to terminal for operator: " << operatorName <<
std::endl;
    }
    int main() {
        // 接收并解析获取需要执行能量隔离操作的目标电气开关柜以及对应的目标操作人
员
        std::string targetSwitchCabinet;
        std::string targetOperator;
        std::cout << "Enter target switch cabinet: ";
        std::cin >> targetSwitchCabinet;
        std::cout << "Enter target operator: ";
        std::cin >> targetOperator;
        // 根据目标电气开关柜以及对应的目标操作人员生成指令数据包
        // 指令数据包内包括目标电气开关柜和目标操作人员的验证信息
        InstructionPacket packet;
```

```
        packet.switchCabinet = targetSwitchCabinet;
        packet.operatorName = targetOperator;
        packet.verificationInfo = "some_verification_info"; // 的验证信息
        // 向目标操作人员对应的操作终端发送指令数据包
        VerificationResult result = sendInstructionToTerminal(packet);
        // 接收操作终端返回的一致性验证结果
        // 判断是否通过身份验证和设备验证
        if (result == VerificationResult::Passed) {
            // 若是，则将目标电器开关柜标记为能量隔离状态
            std::cout << "Verification passed. The switch cabinet " << targetSwitchCabinet << "
is marked as energy isolated." << std::endl;
        } else {
            // 若否，则向操作终端发送恢复指令并结束步骤
            sendRecoveryInstructionToTerminal(targetOperator);
        }
        return 0;
    }
```

[0065]

[0066] 请参阅图3所示,对于操作终端2,在具体实施的过程中首先可以执行步骤S8接收服务端1在步骤S3发送的指令数据包。接下来可以执行步骤S9根据指令数据包内的目标电气开关柜的编号寻找到目标电气开关柜。接下来可以执行步骤S10获取操作人员的身份特征。接下来可以执行步骤S11根据指令数据包内的身份验证信息对操作人员的身份特征进行验证。若未通过则接下来可以执行步骤S12结束步骤或提醒重试,若通过则接下来可以执行步骤S13对目标电器开关柜执行能量隔离操作。接下来可以执行步骤S14在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息,在实际应用中,可以在目标开关柜的抽屉壳体的外侧设置记载目标开关柜设备信息的二维码作为隐藏设备信息,只要在抽出目标电器开关柜进行能量隔离操作之后才能读取隐藏设备信息。接下来可以执行步骤S15根据指令数据包对隐藏设备信息进行验证。若通过则接下来可以执行步骤S16向服务端返回通过身份验证和设备验证的一致性验证结果。若未通过则最后可以执行步骤S17向服务端返回身份验证和设备验证的不一致性验证结果。

[0067] 以上步骤在实施的过程中服务端接收并解析获取目标电气开关柜和目标操作人员信息,生成包含验证信息的指令数据包,并发送至目标操作人员的操作终端。操作终端接收服务端指令数据包,根据编号找到目标电气开关柜,并验证操作人员身份特征。如果未通过验证,结束或提醒重试;如果通过验证,执行能量隔离操作,并获取隐藏设备信息。根据指令数据包验证隐藏设备信息,通过则返回一致性验证结果给服务端,未通过则返回不一致性验证结果。本方案通过多重验证操作人员和设备,在避免误操作的前提下有效提高操作效率。

[0068] 为了对上述的步骤S8至步骤S17的实施过程进行补充说明,提供部分功能模块的源代码,并在注释部分进行对照解释说明。

```
#include <iostream>
#include <string>
// 模拟的指令数据包结构
struct InstructionPacket {
    std::string switchCabinet; // 目标电气开关柜
    std::string operatorName;  // 目标操作人员
    std::string verificationInfo; // 验证信息
};
// 模拟的一致性验证结果
enum class VerificationResult {
    Passed,
    Failed
};
```

[0069]

```
// 模拟获取操作人员身份特征的函数
std::string getOperatorIdentity(const std::string& operatorName) {
    // 此处可以加入真实的获取逻辑
    return "identity_feature"; // 的身份特征
}
// 模拟根据指令数据包内的身份验证信息对操作人员的身份特征进行验证的函数
VerificationResult verifyOperatorIdentity(const std::string& verificationInfo, const std::string&
identityFeature) {
    // 此处可以加入真实的验证逻辑
    return VerificationResult::Passed; // 总是通过验证
}
// 模拟获取目标电器开关柜的隐藏设备信息的函数
std::string getHiddenDeviceInfo(const std::string& switchCabinet) {
    // 此处可以加入真实的获取逻辑
```

```
        return "hidden_device_info"; // 的隐藏设备信息
    }
    // 模拟根据指令数据包对隐藏设备信息进行验证的函数
    VerificationResult verifyHiddenDeviceInfo(const std::string& verificationInfo, const
std::string& hiddenDeviceInfo) {
        // 此处可以加入真实的验证逻辑
        return VerificationResult::Passed; // 总是通过验证
    }
    // 模拟向服务端返回一致性验证结果的函数
    void sendVerificationResultToServer(VerificationResult result) {
        // 此处可以加入真实的发送逻辑
        if (result == VerificationResult::Passed) {
            std::cout << "Sending passed verification result to server." << std::endl;
        } else {
            std::cout << "Sending failed verification result to server." << std::endl;
        }
[0070]  }
    int main() {
        // 已经接收到了服务端发送的指令数据包
        InstructionPacket receivedPacket;
        receivedPacket.switchCabinet = "target_switch_cabinet";
        receivedPacket.operatorName = "target_operator";
        receivedPacket.verificationInfo = "received_verification_info";
        // 获取操作人员的身份特征
        std::string identityFeature = getOperatorIdentity(receivedPacket.operatorName);
        // 根据指令数据包内的身份验证信息对操作人员的身份特征进行验证
        VerificationResult operatorVerificationResult =
verifyOperatorIdentity(receivedPacket.verificationInfo, identityFeature);
        if (operatorVerificationResult == VerificationResult::Failed) {
            // 若未通过，则结束步骤或提醒重试
            std::cout << "Operator identity verification failed." << std::endl;
            return 0;
        }
```

```
        // 若通过，则对目标电器开关柜执行能量隔离操作
        // 请在这里添加执行能量隔离操作的代码
        // 在目标电器开关柜的隐藏区域获取目标电器开关柜的设备信息作为隐藏设备信息
        std::string hiddenDeviceInfo = getHiddenDeviceInfo(receivedPacket.switchCabinet);
        // 根据指令数据包对隐藏设备信息进行验证
        VerificationResult deviceVerificationResult =
verifyHiddenDeviceInfo(receivedPacket.verificationInfo, hiddenDeviceInfo);
        if (deviceVerificationResult == VerificationResult::Failed) {
[0071]      // 若未通过，则向服务端返回身份验证和设备验证的不一致性验证结果
            std::cout << "Hidden device information verification failed." << std::endl;
            sendVerificationResultToServer(VerificationResult::Failed);
            return 0;
        }
        // 若通过，则向服务端返回通过身份验证和设备验证的一致性验证结果
        sendVerificationResultToServer(VerificationResult::Passed);
        return 0;
    }
```

[0072] 请参阅图4所示，在操作人员对目标电气开关柜进行操作的过程中进行二次验证，同时也避免操作人员在发现错误后拒不上报，可以将二次验证中使用的设备验证信息进行加密。有鉴于此，上述的步骤S2在具体实施的过程中首先可以执行步骤S21根据目标电气开关柜的编号生成得到设备信息。接下来可以执行步骤S22将设备验证指令和设备信息组合得到设备验证信息。接下来可以执行步骤S23将目标操作人员的生物信息进行特征提取得到目标操作人员的身份特征。接下来可以执行步骤S24根据目标操作人员的身份特征生成身份验证信息，并对设备验证信息进行加密得到设备验证加密信息。最后可以执行步骤S25根据目标电气开关柜的编号、设备验证加密信息和身份验证信息生成得到指令数据包。

[0073] 为了对上述步骤的实施过程进行补充说明，提供部分功能模块的源代码，并在注释部分进行对照解释说明。

```
#include <iostream>
#include <string>
```

[0074]

```
// 模拟的指令数据包结构
struct InstructionPacket {
    std::string switchCabinet;          // 目标电气开关柜
```

```
    std::string encryptedDeviceVerificationInfo; // 加密的设备验证信息
    std::string identityVerificationInfo; // 身份验证信息
};
// 模拟根据目标电气开关柜的编号生成设备信息的函数
std::string generateDeviceInfo(const std::string& switchCabinetId) {
    // 此处可以加入真实的设备信息生成逻辑
    return "device_info_" + switchCabinetId; // 的设备信息
}
// 模拟将设备验证指令和设备信息组合得到设备验证信息的函数
std::string generateDeviceVerificationInfo(const std::string& deviceInfo) {
    // 此处可以加入真实的设备验证信息生成逻辑
    return "device_verification_info_" + deviceInfo; // 的设备验证信息
}
// 模拟将目标操作人员的生物信息进行特征提取得到目标操作人员的身份特征的函数
std::string extractIdentityFeature(const std::string& biometricInfo) {
    // 此处可以加入真实的特征提取逻辑
    return "identity_feature_" + biometricInfo; // 的身份特征
}
// 模拟根据目标操作人员的身份特征生成身份验证信息的函数
std::string generateIdentityVerificationInfo(const std::string& identityFeature) {
    // 此处可以加入真实的身份验证信息生成逻辑
    return "identity_verification_info_" + identityFeature; // 的身份验证信息
}
// 模拟对设备验证信息进行加密的函数
std::string encryptDeviceVerificationInfo(const std::string& deviceVerificationInfo) {
    // 此处可以加入真实的加密逻辑
    return "encrypted_" + deviceVerificationInfo; // 的加密设备验证信息
}
int main() {
    // 输入：目标电气开关柜的编号和目标操作人员的生物信息
    std::string targetSwitchCabinetId = "123";
    std::string targetOperatorBiometricInfo = "biometric_info";
```

[0075]

```
        // 根据目标电气开关柜的编号生成得到设备信息
        std::string deviceInfo = generateDeviceInfo(targetSwitchCabinetId);
        // 将设备验证指令和设备信息组合得到设备验证信息
        std::string deviceVerificationInfo = generateDeviceVerificationInfo(deviceInfo);
        // 将目标操作人员的生物信息进行特征提取得到目标操作人员的身份特征
        std::string identityFeature = extractIdentityFeature(targetOperatorBiometricInfo);
        // 根据目标操作人员的身份特征生成身份验证信息
        std::string identityVerificationInfo = generateIdentityVerificationInfo(identityFeature);
        // 对设备验证信息进行加密得到设备验证加密信息
        std::string encryptedDeviceVerificationInfo =
encryptDeviceVerificationInfo(deviceVerificationInfo);
        // 根据目标电气开关柜的编号、设备验证加密信息和身份验证信息生成得到指令数
据包
```

[0076]

```
        InstructionPacket packet;
        packet.switchCabinet = targetSwitchCabinetId;
        packet.encryptedDeviceVerificationInfo = encryptedDeviceVerificationInfo;
        packet.identityVerificationInfo = identityVerificationInfo;
        // 输出：指令数据包
        std::cout << "Instruction packet generated: "
                  << "switchCabinet = " << packet.switchCabinet << ", "
                  << "encryptedDeviceVerificationInfo = " <<
packet.encryptedDeviceVerificationInfo << ", "
                  << "identityVerificationInfo = " << packet.identityVerificationInfo <<
std::endl;
        return 0;
    }
```

[0077] 请参阅图5所示，由于电气设备操作间的电磁环境复杂，服务端1与操作终端2的连接并不稳定，可能导致通讯延迟，为了提高操作效率，可以将身份验证信息和设备验证加密信息都发送到操作终端2。但是为了避免目标操作人员的身份特征泄露，上述的步骤S24在实施的过程中首先可以执行步骤S241按照生成算法获取目标操作人员的身份特征的不可逆的数字摘要。接下来可以执行步骤S242将目标操作人员的身份特征的数字摘要拆分为显性数字段和隐性数字段。接下来可以执行步骤S243使用隐性数字段作为密钥对设备验证信息进行加密得到设备验证加密信息。最后可以执行步骤S244将生成算法和显性数字段组合得到身份验证信息。

[0078] 为了对上述步骤的实施过程进行补充说明，提供部分功能模块的源代码，并在注释部分进行对照解释说明。

```
#include <iostream>
#include <string>
// 模拟的指令数据包结构
struct InstructionPacket {
    std::string switchCabinet;              // 目标电气开关柜
    std::string encryptedDeviceVerificationInfo; // 加密的设备验证信息
    std::string identityVerificationInfo; // 身份验证信息
};
// 模拟按照生成算法获取目标操作人员的身份特征的不可逆的数字摘要的函数
std::string getIdentityFeatureDigest(const std::string& identityFeature) {
    // 此处可以加入真实的生成算法
    return "digest_" + identityFeature; // 的数字摘要
}
// 模拟将目标操作人员的身份特征的数字摘要拆分为显性数字段和隐性数字段的函数
std::pair<std::string, std::string> splitDigest(const std::string& digest) {
    // 此处可以加入真实的拆分逻辑
    return {"explicit_" + digest, "implicit_" + digest}; // 的显性数字段和隐性数字段
}
// 模拟使用隐性数字段作为密钥对设备验证信息进行加密的函数
std::string encryptDeviceVerificationInfo(const std::string& deviceVerificationInfo, const std::string& key) {
    // 此处可以加入真实的加密逻辑
    return "encrypted_" + deviceVerificationInfo + "_with_" + key; // 的加密设备验证信息
}
// 模拟将生成算法和显性数字段组合得到身份验证信息的函数
std::string generateIdentityVerificationInfo(const std::string& algorithm, const std::string& explicitNumberField) {
    // 此处可以加入真实的生成逻辑
```

[0079]

```
        return "identity_verification_info_" + algorithm + "_" + explicitNumberField; // 的身份
验证信息
    }
    int main() {
        // 输入：目标操作人员的身份特征和设备验证信息
        std::string targetOperatorIdentityFeature = "identity_feature";
        std::string deviceVerificationInfo = "device_verification_info";
        // 按照生成算法获取目标操作人员的身份特征的不可逆的数字摘要
        std::string identityFeatureDigest =
getIdentityFeatureDigest(targetOperatorIdentityFeature);
        // 将目标操作人员的身份特征的数字摘要拆分为显性数字段和隐性数字段
        std::pair<std::string, std::string> splitResult = splitDigest(identityFeatureDigest);
        std::string explicitNumberField = splitResult.first;
        std::string implicitNumberField = splitResult.second;
        // 使用隐性数字段作为密钥对设备验证信息进行加密得到设备验证加密信息
        std::string encryptedDeviceVerificationInfo =
encryptDeviceVerificationInfo(deviceVerificationInfo, implicitNumberField);
        // 将生成算法和显性数字段组合得到身份验证信息
        std::string identityVerificationInfo = generateIdentityVerificationInfo("algorithm",
explicitNumberField);
        // 输出：设备验证加密信息和身份验证信息
        std::cout << "Device Verification Info (Encrypted): " << encryptedDeviceVerificationInfo
<< "\n";
        std::cout << "Identity Verification Info: " << identityVerificationInfo << std::endl;
        return 0;
    }
```

[0080]

[0081] 请参阅图6所示，为了有效降低身份特征的数字摘要的数据量，同时也避免不同操作人员的身份特征的数字摘要出现雷同，需要提取目标操作人员最具有代表性的特征。有鉴于此，上述的步骤S241在实施的过程中能够首先可以执行步骤S2411根据目标操作人员的身份特征得到目标操作人员的身份特征在多个特征维度上的数值分布。接下来可以执行步骤S2412对于每个特征维度，根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间。接下来可以执行步骤S2413将每个特征维度和对应的常规分布区间的数据信息组合为一个特征数据组。接下来可以执行步骤S2414将每个特征数据组根据对应的常规分布区间分配至显性数据块或隐性数据块。将常规分布区间大的特征数据组分配至显性数据块，将常规分布区间小的特征数据组分配至隐性数据块。最后可以执行步骤S2415按照生成算法获取显性数据块和隐性数据块的不可逆的数字摘要，组合得到身份特征的数字摘要。

[0082] 根据以上步骤，将显性数据块对应的数字摘要作为显性数字段，将隐性数据块对

应的数字摘要作为隐性数字段。

[0083] 为了对上述步骤的实施过程进行补充说明,提供部分功能模块的源代码,并在注释部分进行对照解释说明。

[0084]

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <algorithm>
// 特征维度的数据结构
struct FeatureDimension {
    std::string name;                    // 特征维度的名称
    std::vector<int> distribution; // 特征维度上的数值分布
};
// 特征数据组的数据结构
struct FeatureDataGroup {
    std::string name;                  // 特征维度的名称
    std::vector<int> interval;   // 常规分布区间
};
// 的生成算法
std::string generateDigest(const std::vector<int>& block) {
    int sum = std::accumulate(block.begin(), block.end(), 0);
    return std::to_string(sum);
}
int main() {
    // 输入：目标操作人员的身份特征在多个特征维度上的数值分布
    std::vector<FeatureDimension> identityFeature = {
        {"feature1", {1, 2, 3, 4, 5}},
```

```
            {"feature2", {6, 7, 8, 9, 10}}
        };
        std::vector<FeatureDataGroup> dataGroups;
        std::vector<int> explicitBlock, implicitBlock;
        // 对于每个特征维度
        for (const auto& dimension : identityFeature) {
            // 根据对应的数值分布得到目标操作人员在对应特征维度上的常规分布区间
            // 常规分布区间是数值分布的最小值和最大值
            int minValue = *std::min_element(dimension.distribution.begin(),
dimension.distribution.end());
            int maxValue = *std::max_element(dimension.distribution.begin(),
dimension.distribution.end());
            // 将每个特征维度和对应的常规分布区间的数据信息组合为一个特征数据组
            dataGroups.push_back({dimension.name, {minValue, maxValue}});
            // 将每个特征数据组根据对应的常规分布区间分配至显性数据块或隐性数据块
[0085]      // 奇数特征维度分配到显性数据块，偶数特征维度分配到隐性数据块
            if ((maxValue - minValue) % 2 == 1) {
                explicitBlock.push_back(maxValue - minValue);
            } else {
                implicitBlock.push_back(maxValue - minValue);
            }
        }
        // 按照生成算法获取显性数据块和隐性数据块的不可逆的数字摘要
        std::string explicitDigest = generateDigest(explicitBlock);
        std::string implicitDigest = generateDigest(implicitBlock);
        // 组合得到身份特征的数字摘要
        std::string identityFeatureDigest = explicitDigest + "_" + implicitDigest;
        // 输出：身份特征的数字摘要
        std::cout << "Identity feature digest: " << identityFeatureDigest << std::endl;
        return 0;
    }
```

[0086] 请参阅图7所示,为了计算得到目标操作人员在对应特征维度上的常规分布区间,在提高身份识别效率的前提下不降低识别准确度,对于每个特征维度,上述的步骤S2412在具体实施的过程中首先可以执行步骤S24121根据对应的数值分布获取对应数值的顺序排序序列作为数值序列。接下来可以执行步骤S24122获取数值序列内全部相邻的两个数值的差值。接下来可以执行步骤S24123计算获取数值序列内全部相邻的两个数值的差值的均值作为筛选标定值。接下来可以执行步骤S24124将数值序列内与相邻数值的差值小于筛选标定值的数值作为异常值进行剔除。最后可以执行步骤S24125获取剔除异常值之后的数值序

列内数值的分布区间作为目标操作人员在该特征维度的常规分布区间。

[0087] 为了对上述步骤的实施过程进行补充说明,提供部分功能模块的源代码,并在注释部分进行对照解释说明。

[0088]

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <numeric>
// 特征维度的数据结构
struct FeatureDimension {
    std::string name;                    // 特征维度的名称
    std::vector<int> distribution; // 特征维度上的数值分布
};
// 特征数据组的数据结构
struct FeatureDataGroup {
    std::string name;                  // 特征维度的名称
    std::vector<int> interval;   // 常规分布区间
};
int main() {
    // 输入：目标操作人员的身份特征在多个特征维度上的数值分布
    std::vector<FeatureDimension> identityFeature = {
        {"feature1", {1, 2, 3, 4, 5}},
        {"feature2", {6, 7, 8, 9, 10}}
    };
    std::vector<FeatureDataGroup> dataGroups;
    // 对于每个特征维度
    for (const auto& dimension : identityFeature) {
```

```
                // 根据对应的数值分布获取对应数值的顺序排序序列作为数值序列
                std::vector<int> valueSequence = dimension.distribution;
                std::sort(valueSequence.begin(), valueSequence.end());
                // 获取数值序列内全部相邻的两个数值的差值
                std::vector<int> diffSequence(valueSequence.size() - 1);
                std::adjacent_difference(valueSequence.begin(), valueSequence.end(),
diffSequence.begin());
                diffSequence.erase(diffSequence.begin());
                // 计算获取数值序列内全部相邻的两个数值的差值的均值作为筛选标定值
                double diffMean = std::accumulate(diffSequence.begin(), diffSequence.end(), 0.0) /
diffSequence.size();
                // 将数值序列内与相邻数值的差值小于筛选标定值的数值作为异常值进行剔除
                std::vector<int> cleanedSequence;
                for (int i = 1; i < valueSequence.size(); ++i) {
                    if (std::abs(valueSequence[i] - valueSequence[i - 1]) >= diffMean) {
                        cleanedSequence.push_back(valueSequence[i]);
                    }
                }
                // 获取剔除异常值之后的数值序列内数值的分布区间作为目标操作人员在该特
征维度的常规分布区间
                int minValue = *std::min_element(cleanedSequence.begin(), cleanedSequence.end());
                int maxValue = *std::max_element(cleanedSequence.begin(),
cleanedSequence.end());
                // 将每个特征维度和对应的常规分布区间的数据信息组合为一个特征数据组
                dataGroups.push_back({dimension.name, {minValue, maxValue}});
            }
            // 输出：常规分布区间
            for (const auto& group : dataGroups) {
                std::cout << "Feature: " << group.name << ", Normal distribution interval: [" <<
group.interval[0] << ", " << group.interval[1] << "]" << std::endl;
            }
            return 0;
        }
```

[0089]

[0090] 请参阅图8所示,为了对隐藏设备信息进行解密,上述的步骤S15在具体实施的过程中首先可以执行步骤S151根据指令数据包解析得到生成算法、显性数字段以及设备验证加密信息。接下来可以执行步骤S152利用生成算法获取操作人员的身份特征的数字摘要。接下来可以执行步骤S153根据数字摘要以及显性数字段得到隐性数字段。接下来可以执行步骤S154根据隐性数字段作为密钥对设备验证加密信息进行解密得到设备验证信息内的

设备验证指令和设备信息。接下来可以执行步骤S155执行设备验证指令对设备验证信息内设备信息和隐藏设备信息进行比对。若相同则接下来可以执行步骤S156设备验证的一致，若不相同则最后可以执行步骤S157设备验证的不一致。

[0091] 为了对上述步骤的实施过程进行补充说明，提供部分功能模块的源代码，并在注释部分进行对照解释说明。

[0092]

```
#include <iostream>
#include <string>
#include <vector>
// 数据包的数据结构
struct DataPacket {
    std::string generatorAlgorithm; // 生成算法
    std::string explicitNumericalField; // 显性数字段
    std::string encryptedDeviceValidationInfo; // 设备验证加密信息
};
// 设备验证信息的数据结构
struct DeviceValidationInfo {
    std::string deviceValidationInstruction; // 设备验证指令
    std::string deviceInfo; // 设备信息
};
// 设备验证结果的数据结构
struct DeviceValidationResult {
    bool isConsistent; // 是否一致
};
// 解密设备验证信息的函数
DeviceValidationInfo DecryptDeviceValidationInfo(const std::string& encryptedInfo, const std::string& key) {
    // TODO: 实现具体的解密算法
```

```
        return DeviceValidationInfo();
    }
    // 执行设备验证的函数
    bool ExecuteDeviceValidation(const std::string& instruction, const std::string& deviceInfo, const std::string& hiddenDeviceInfo) {
        // TODO: 实现具体的验证算法
        return deviceInfo == hiddenDeviceInfo;
    }
    int main() {
        // 输入：指令数据包和隐藏设备信息
        DataPacket packet;
        std::string hiddenDeviceInfo;
        // 根据指令数据包解析得到生成算法、显性数字段以及设备验证加密信息
        std::string generatorAlgorithm = packet.generatorAlgorithm;
        std::string explicitNumericalField = packet.explicitNumericalField;
        std::string encryptedDeviceValidationInfo = packet.encryptedDeviceValidationInfo;
        // 利用生成算法获取操作人员的身份特征的数字摘要
        // TODO: 实现具体的生成算法
        // 根据数字摘要以及显性数字段得到隐性数字段
        // TODO: 实现具体的算法
        // 根据隐性数字段作为密钥对设备验证加密信息进行解密得到设备验证信息内的设备验证指令和设备信息
        DeviceValidationInfo validationInfo = DecryptDeviceValidationInfo(encryptedDeviceValidationInfo, hiddenNumericalField);
        // 执行设备验证指令对设备验证信息内设备信息和隐藏设备信息进行比对
        bool isConsistent = ExecuteDeviceValidation(validationInfo.deviceValidationInstruction, validationInfo.deviceInfo, hiddenDeviceInfo);
        // 输出：设备验证结果
        DeviceValidationResult validationResult = { isConsistent };
        std::cout << "Device validation result: " << (validationResult.isConsistent ? "Consistent" : "Inconsistent") << std::endl;
        return 0;
    }
```

[0093]

[0094] 综上所述,本方案采用多重验证的方法,有效提高操作效率的同时避免误操作。在实施过程中首先获取需要执行能量隔离操作的目标电气开关柜及对应的目标操作人员信息,并进行解析。随后,根据这些信息生成指令数据包,并将其发送到目标操作人员的操作终端。接着,接收操作终端返回的一致性验证结果,并进行身份验证和设备验证。若验证通

过,将目标电气开关柜标记为能量隔离状态。若验证未通过,则向操作终端发送恢复指令并结束操作。通过以上流程,可确保操作过程中避免误操作,并有效提高操作效率。

[0095] 附图中的流程图和框图显示了根据本申请的多个实施例的装置、系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分,所述模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中,方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如,两个连续的方框实际上可以基本并行地执行,它们有时也可以按相反的顺序执行,这依所涉及的功能而定。

[0096] 也要注意的是,框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合,可以用执行相应的功能或动作的硬件,例如电路或ASIC(专用集成电路,Application Specific Integrated Circuit)来实现,或者可以用硬件和软件的组合,如固件等来实现。

[0097] 尽管在此结合各实施例对本发明进行了描述,然而,在实施所要求保护的本发明过程中,本领域技术人员通过查看所述附图、公开内容、以及所附权利要求书,可理解并实现所述公开实施例的其它变化。在权利要求中,“包括”(comprising)一词不排除其他组成部分或步骤,“一”或“一个”不排除多个的情况。单个处理器或其它单元可以实现权利要求中列举的若干项功能。相互不同的从属权利要求中记载了某些措施,但这并不表示这些措施不能组合起来产生良好的效果。

[0098] 以上已经描述了本申请的各实施例,上述说明是示例性的,并非穷尽性的,并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围的情况下,对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择,旨在最好地解释各实施例的原理、实际应用或对市场中的技术的改进,或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

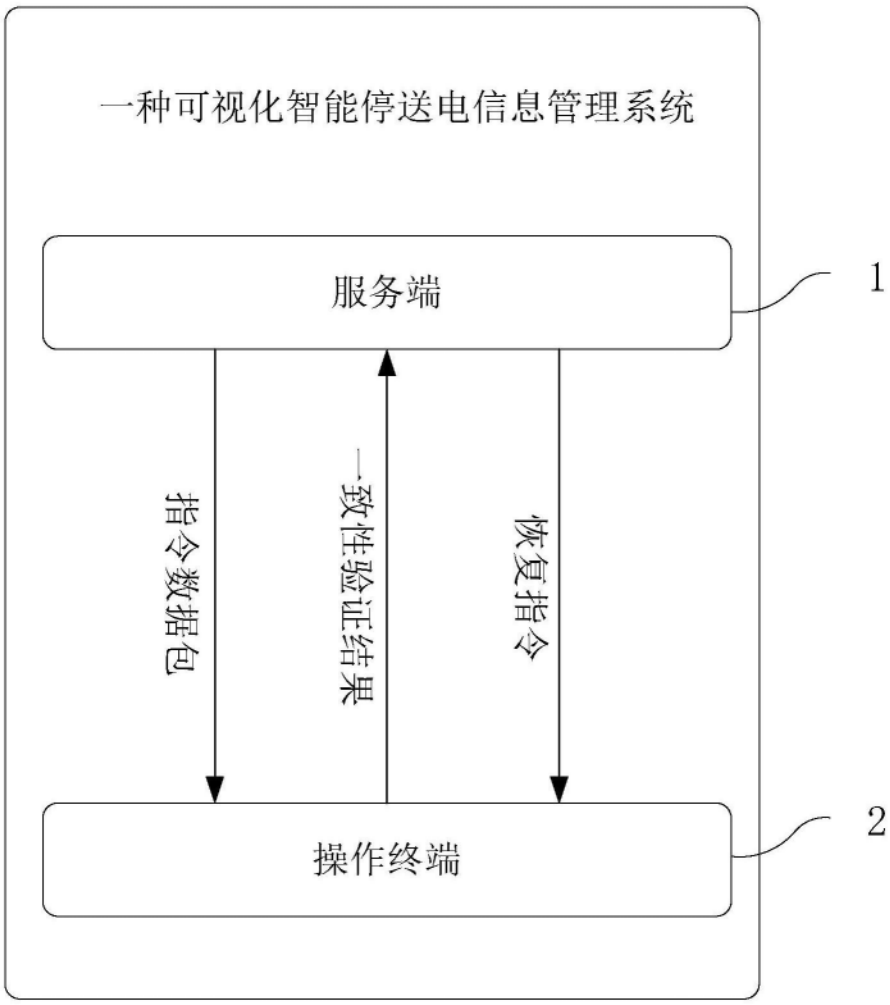


图1

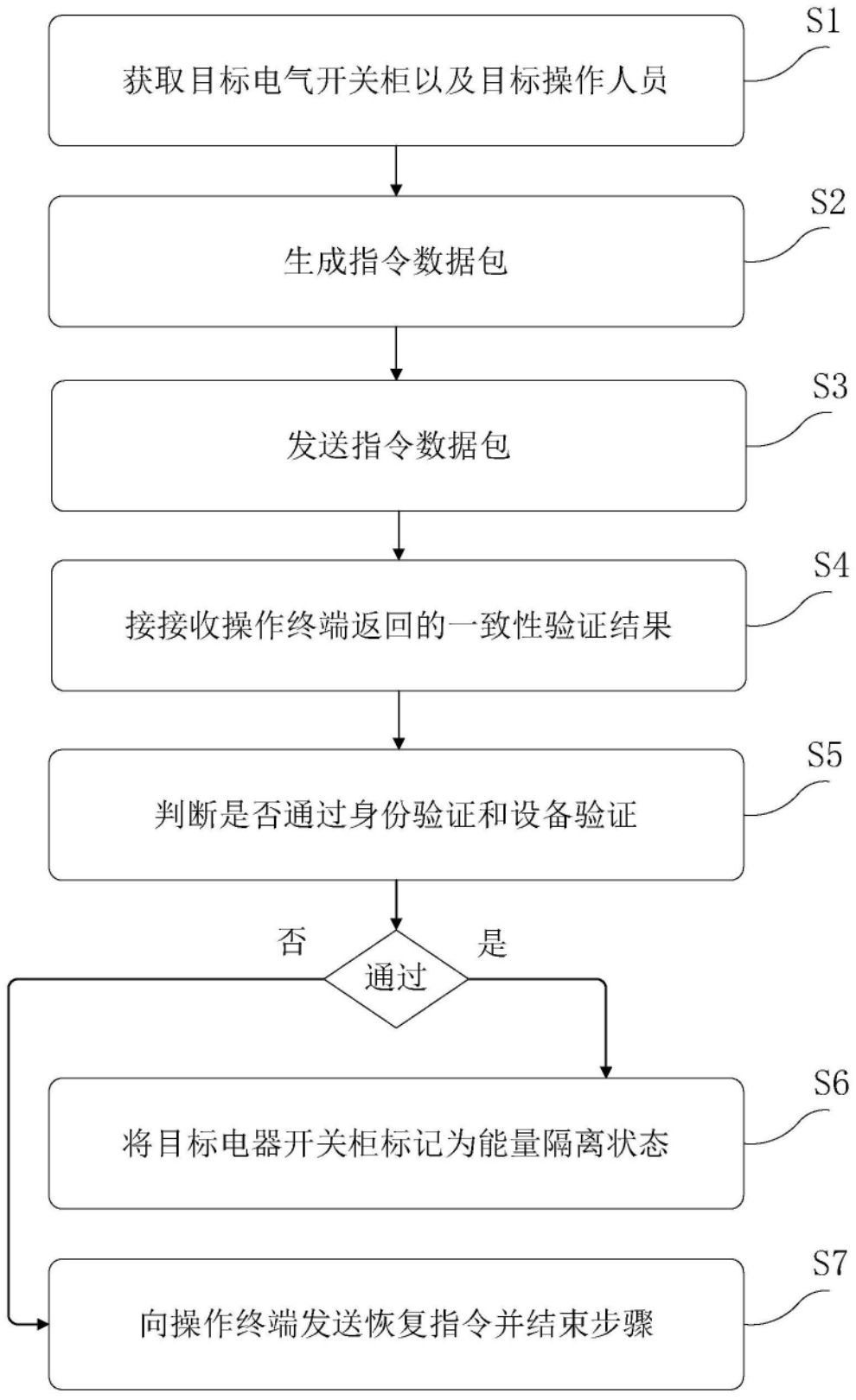

S1
获取目标电气开关柜以及目标操作人员
S2
生成指令数据包
S3
发送指令数据包
S4
接接收操作终端返回的一致性验证结果
S5
判断是否通过身份验证和设备验证
否
通过
是
S6
将目标电器开关柜标记为能量隔离状态
S7
向操作终端发送恢复指令并结束步骤

图2

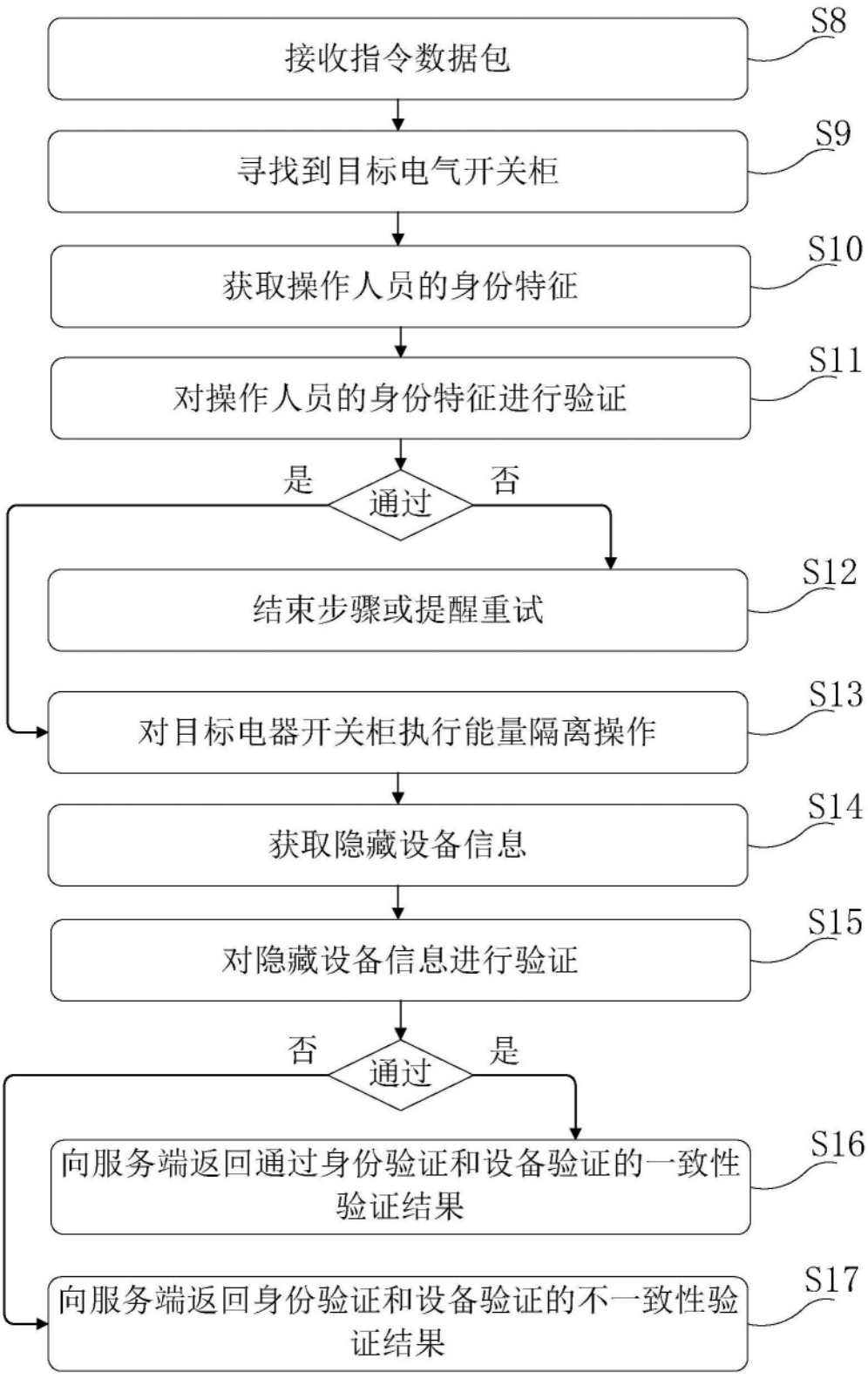

接收指令数据包
S8
寻找到目标电气开关柜
S9
获取操作人员的身份特征
S10
对操作人员的身份特征进行验证
S11
是
通过
否
结束步骤或提醒重试
S12
对目标电器开关柜执行能量隔离操作
S13
获取隐藏设备信息
S14
对隐藏设备信息进行验证
S15
否
通过
是
向服务端返回通过身份验证和设备验证的一致性验证结果
S16
向服务端返回身份验证和设备验证的不一致性验证结果
S17

图3

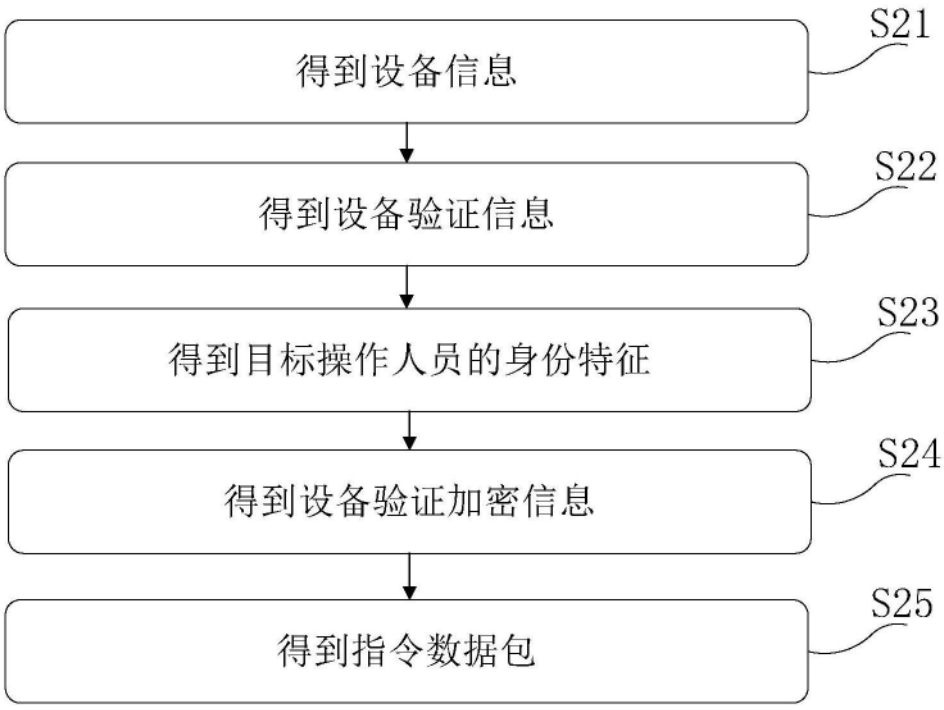

S21
得到设备信息
S22
得到设备验证信息
S23
得到目标操作人员的身份特征
S24
得到设备验证加密信息
S25
得到指令数据包

图4

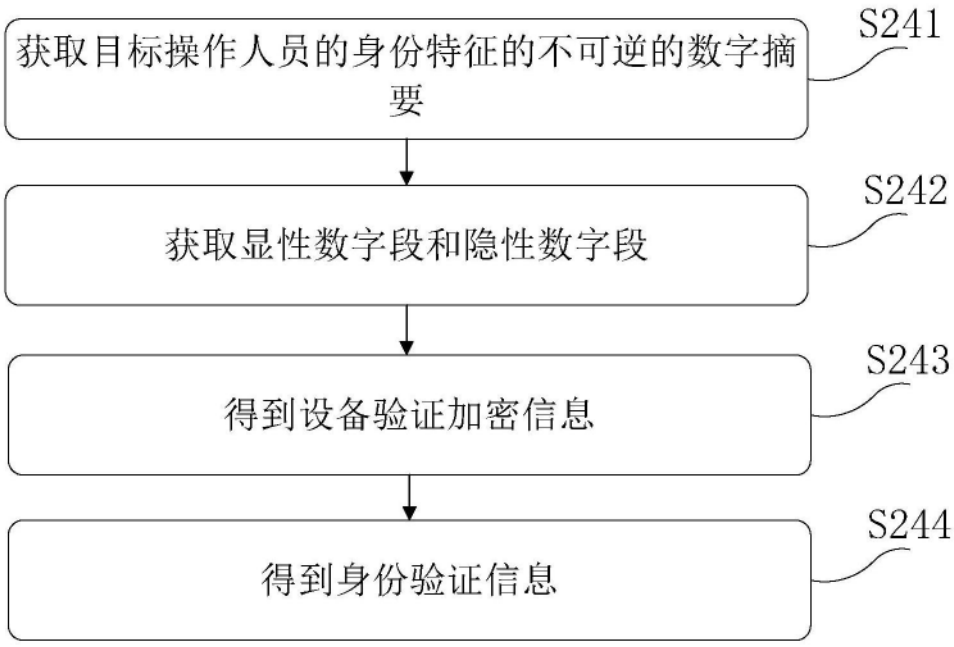

S241
获取目标操作人员的身份特征的不可逆的数字摘要
S242
获取显性数字段和隐性数字段
S243
得到设备验证加密信息
S244
得到身份验证信息

图5

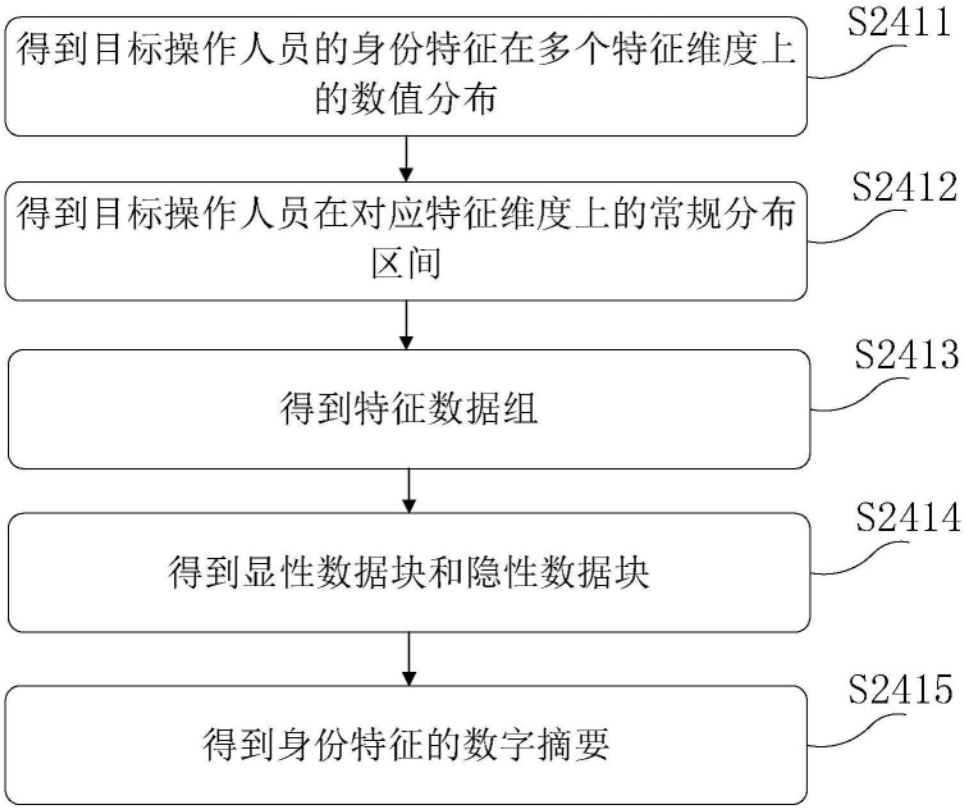

得到目标操作人员的身份特征在多个特征维度上的数值分布
S2411
得到目标操作人员在对应特征维度上的常规分布区间
S2412
得到特征数据组
S2413
得到显性数据块和隐性数据块
S2414
得到身份特征的数字摘要
S2415

图6

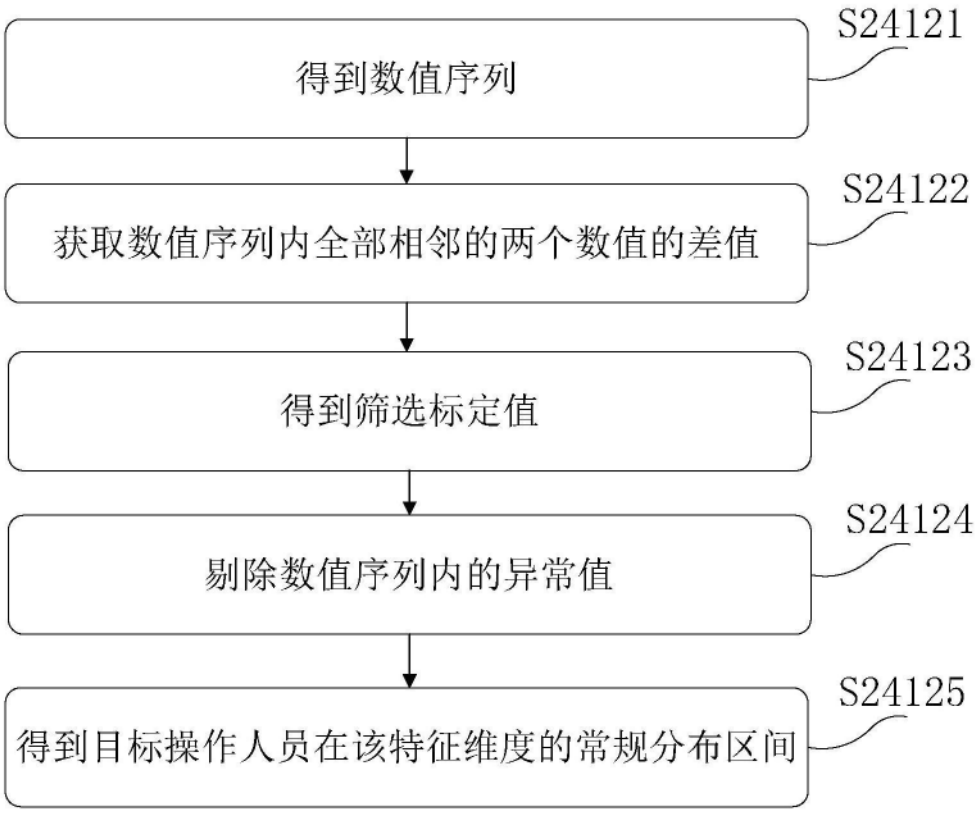

S24121
得到数值序列
S24122
获取数值序列内全部相邻的两个数值的差值
S24123
得到筛选标定值
S24124
剔除数值序列内的异常值
S24125
得到目标操作人员在该特征维度的常规分布区间

图7

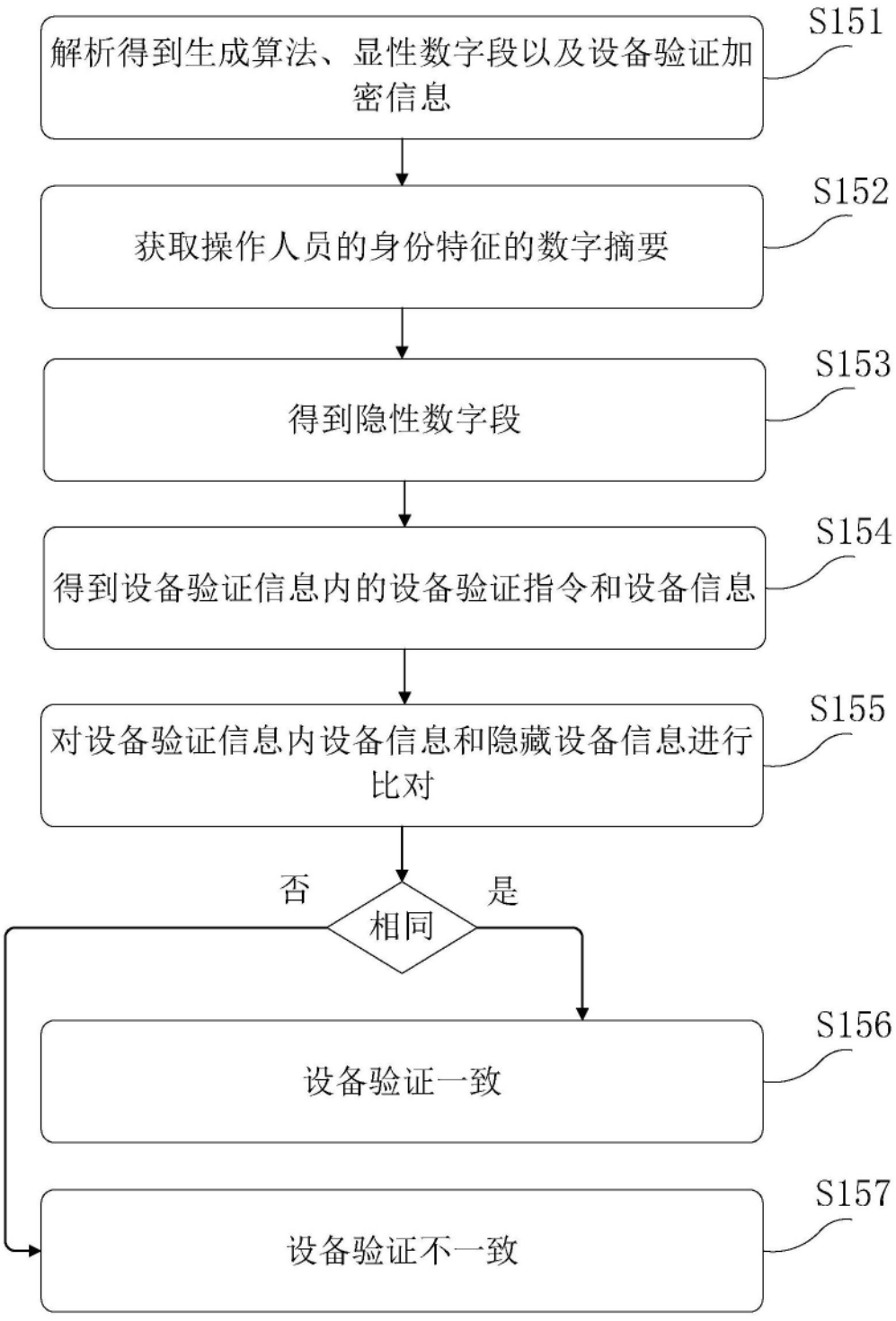

解析得到生成算法、显性数字段以及设备验证加密信息
S151
获取操作人员的身份特征的数字摘要
S152
得到隐性数字段
S153
得到设备验证信息内的设备验证指令和设备信息
S154
对设备验证信息内设备信息和隐藏设备信息进行比对
S155
否
相同
是
设备验证一致
S156
设备验证不一致
S157

图8