https://doi.org/10.1051/jnwpu/20224051188

电力物联网终端存储设备身份认证与数据保护方法研究

何佩[1], 郑文斌[2], 池晓金[2], 蔡怡挺[2], 姚红静[1]

(1.西北工业大学 计算机学院, 陕西 西安 710072; 2.国网浙江省电力有限公司温州供电公司, 浙江 温州 325000)

摘 要:针对电力物联网智能终端与数据中心通信链路低带宽、高延迟的特点,结合泛在物联的网络环境中数据同步和共享的安全需求,以实现通信资源受限情况下尽量减少更新文件传输代价为目标,设计了一种实现轻量级信息动态安全存储与验证的方法,确保了终端存储设备的信息安全。该方法通过动态证明存储技术对数据中心存储密态文件完整性进行验证,并且支持对密态文件的动态修改处理和增量更新。对于修改的文件,仅传输修改部分的数据块到数据中心完成更新,无需重新上传整个文件,有效降低了验证的计算开销和传输的通信开销。

关 键 词:电力物联网;终端;身份认证;数据保护;动态证明存储;安全性

中图分类号:TP309G **文献标志码**:A **文章编号**:1000-2758(2022)05-1188-07

随着电力物联网发展及能源数字化转型,电力边缘智能终端的适用场景不断拓展,构建具备边缘侧智能感知及分析处理能力的电力边缘智能终端已成为数字化电网的迫切需要和必然发展趋势[1]。电网作为国家重大信息安全基础设施,此时面向终端数据存储和交换的信息安全成为用户日益关注的焦点之一。

应用中大容量存储设备存在着重大的安全隐患,例如非授权用户能够随意非法读取并复制存储于该大容量设备中的明文数据。此外,大容量设备与外部设备连接过程中,攻击者能够轻易截获此过程中的数据。因此,结合当前电力物联网智能终端设备操作系统采用文件系统进行信息存储和共享的典型应用场景,研究文件同步和共享过程中的安全性问题,即开展用户身份的认证以及存储设备中明文数据的保护至关重要。

文件同步和共享过程的安全性包含机密性和完整性两方面,机密性涉及密态文件的同步和统一访问控制等技术,完整性涉及对数据中心存储文件完整性的验证技术[2]。传统的消息认证码(MAC)、数字签名以及确定一个最新版本文件而直接全覆盖、对文件进行无损压缩等方法存在带宽占用量大、压缩程度有限等不足使其均不可取。为此,论文针对电力物联网智能终端设备通信资源受限的特点,研究一种面向电力物联网终端设备数据存储和交换行为的轻量级身份认证与数据保护方法。通过设计动态证明存储系统(DyPoS)[3-6]建立了基于文件系统实现信息的安全存储与共享的机制。即在文件同步到中心服务器时保证文件的机密性与完整性。为了防止加解密机制占用过多边缘计算资源,对于文件的修改、再次同步时仅加密传输修改部分,而不需要加密和传输整个文件,从而降低计算和传输开销。

1 证明存储原理

证明存储的目的是要由物联网终端发起验证数据中心存储文件的完整性,即验证所有的分块都如实存储[7]。证明存储的做法是终端随机选择一些分块索引发送给数据中心,选择的分块索引数量称为挑战块数。数据中心将这些挑战的数据块和验证

收稿日期:2021-12-18 **基金项目**:国网浙江省电力有限公司科技项目(5211WZ1900RU)、JCKY 项目和中央高校基本科研业务费(D500022035)资助

作者简介:何佩(1991—),西北工业大学博士研究生,主要从事数字化系统智能运维研究。e-mail:1059293443@qq.com

这些数据块会用到的认证结构信息作为认证器返回给终端。终端验证这些数据块,如果是完整的,则以一定的概率认为文件是如实存储的[8]。

将文件 A 以 S 字节分成一系列没有重叠的数据块(最后的块可能比 S 字节短)。如文件 $F=\{m_1,m_2,m_3,m_4\}$,分块加密后为$\{c_1,c_2,c_3,c_4\}$。图1所示的认证结构和文件的密文分块保存在数据中心,终端随机选择一个分块索引作为挑战发送给数据中心(挑战块数为1),这里假设选择的索引为2,即验证分块 c_2 的完整性。数据中心返回存储的分块 c_2 和 ν_1,ν_3,ν_4,终端由数据块 c_2 计算 ν_5',由 ν_4 和 ν_5' 计算 ν_2',最后计算出 ν_1'。如果 v_1' 和 ν_1 一致,则说明 c_2 被完整存储。因为 c_2 是随机选取的,所以终端可以以一定的概率认为文件 F 的密文块在数据中心上是如实存储的。

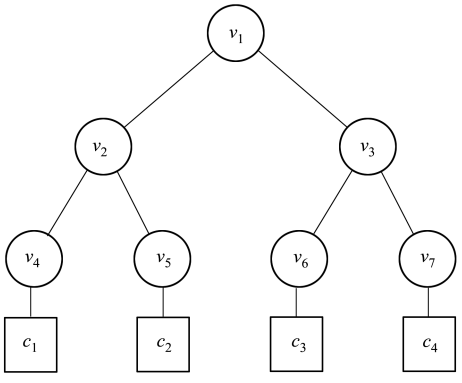
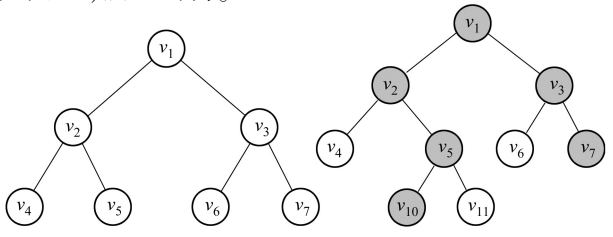


图 1　证明存储过程示例

由以上描述可知,实现高效证明存储的关键是建立一个新的认证结构,即同态认证树 HAT(homomorphic authenticated tree)。HAT 是一个二叉树,其中每个叶节点对应一个数据块。

尽管 HAT 对数据块的数量没有任何限制,为了描述简单,假设数据块的数量 n 等于完整的二叉树中的叶子节点的数量。因此,对于文件 $F=(m_1,m_2,m_3,m_4)$(其中,m_i 表示文件的第 i 块),可以构建如图 2a)所示的树。



a)　初始文件 F　　b)　文件 F 更新 F'

图 2　基于树的认证结构

HAT 中的每个节点都由一个四元组 $v_i=(i,l_i,v_i,t_i)$ 表示。i 是节点的唯一索引,根节点的索引是1,索引从上到下,从左到右增加。l_i 表示可以从第 i 个节点到达的叶节点的数量。v_i 是第 i 个节点的版本号。t_i 代表第 i 个节点的标签。

终端随机选择分块索引发送给数据中心。同时,终端采用路径搜索[9]和兄弟搜索[10]获取数据块和验证这些数据块用到的认证结构信息。

1)路径搜索过程(见附录算法 1)

定义路径搜索 $\rho_t\leftarrow P(T,t)$,即以文件的 HAT 标记 T 和块索引 t 为输入,输出从根节点到该文件第 t 个块对应的叶子节点的路径中的节点的索引集合。T 中的第 i 个节点的唯一索引表示为一个四元组 $v_i=(i,l_i,v_i,t_i)$。

为每个合法块索引 l 初始化2个辅助变量,其中 i_t 定义其根是 T 中第 i_t 个节点的子树,ord_t 指示该子树中相应叶子节点的位置。支持多路径搜索的过程如下:

(1)初始化路径 ρ 和状态 S。

(2)对于 T 的每个级别,通过广度优先搜索循环计算每个块索引 l 在 ρ 中的节点。

2)兄弟搜索过程(见附录算法 2)

定义兄弟搜索 $\boldsymbol{\psi}\leftarrow\mathrm{Sibling}(\rho)$,将路径 ρ 作为输入,输出路径 ρ 中所有节点的兄弟节点的索引集,即输出其余兄弟中最左边的节点集合。

兄弟搜索的过程如下:

(1)初始化兄弟集合 $\boldsymbol{\psi}$ 和辅助集合 $\boldsymbol{Q}$。

(2)确定 ρ 中节点的多少个“孩子”出现在 ρ 行。

a)若为2个,将2个“孩子”从 ρ 中移除,并将右侧的“孩子”插入 $\boldsymbol{Q}$。

b)若为1个,则从 ρ 删除这个“孩子”,并将其插入兄弟集合 $\boldsymbol{\psi}$。

由此,数据中心向终端返回存储的数据块和认证结构信息,终端将采用2种搜索方法获得的数据块和认证结构信息对比验证,如果这些数据块是完整的,则以一定的概率认为文件是如实存储的。

2　动态证明存储系统设计

2.1　DyPoS 系统构造工具[11]

1)防碰撞散列函数

对于散列函数 $H:\{0,1\}^*\rightarrow\{0,1\}^*$,如果找到

满足 $H(x)=H(y)$ 的 2 个不同值 x 和 y 的概率是可以忽略的,那么它是抗碰撞的。

2) 确定性对称加密

加密算法将密钥 k 和明文 m 作为输入,并输出密文。使用符号 $\mathrm{Enc}_k(m)$ 来表示加密算法。

3) 基于散列的消息认证码

基于散列的消息认证码 $\mathrm{HMAC}:\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ 是一个确定性函数,它取一个密钥 k 和一个输入 x,输出一个值 y。定义 $\mathrm{HMAC}_k(x) \overset{\mathrm{def}}{=} \mathrm{HMAC}(k,x)$。

4) 伪随机函数

伪随机函数 $f:\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ 是一个确定性函数,它取一个密钥 k 和一个值 x,并输出一个与同一输入 x 的真正随机函数无法区分的值 y。定义 $f_k(x) \overset{\mathrm{def}}{=} f(k,x)$。

5) 伪随机置换

伪随机置换 $\pi:\{0,1\}^* \times [1,n] \rightarrow [1,n]$ 是一个确定性函数,它取一个密钥 k 和一个整数 x,其中 $1 \leqslant x \leqslant n$,并输出一个值 y(其中 $1 \leqslant y \leqslant n$)与同一输入 x 的真正随机排列无法区分。定义 $\pi_k(x) \overset{\mathrm{def}}{=} \pi(k,x)$。

6) 密钥导出函数

密钥导出函数 $\mathrm{KDF}:\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ 是一个确定性函数,可以从 2 个秘密值中派生出一个密钥。

2.2 动态证明存储算法

1) 终端运行初始化算法 $(\mathrm{id},e) \leftarrow \mathrm{Init}(1^\lambda,F)$ 计算:

$$e \leftarrow H(F),\ \mathrm{id} \leftarrow H(e)$$

然后,终端发送 id 给数据中心作为文件的标识。

2) 假设文件 $F=(m_1,\cdots,m_n)$,终端首先调用编码算法 $(C,T) \leftarrow \mathrm{Encode}(e,F)$,生成分块密文和认证结构,过程如下:

① 生成一个随机密钥 $k \leftarrow \{0,1\}^{|e|}$,并计算 $r \leftarrow k \oplus e$。

② 计算加密密钥 $k_e \leftarrow \mathrm{KDF}(k,0)$。对于每个块 $m_t(1 \leqslant t \leqslant n)$ 计算 $c_t \leftarrow \mathrm{Enc}_{k_e}(m_t)$。

③ 对 F 构建一个 HAT,此时 HAT 中的标签未被赋值。

④ 计算 $\alpha_s \leftarrow \mathrm{KDF}(k,1)$,$k_c \leftarrow \mathrm{KDF}(k,2)$ 和 $\alpha_c \leftarrow \mathrm{KDF}(k,3)$。通过算法 3,用 $(c_1,\cdots,c_n)$ 计算 HAT 中所有叶节点的标签。

⑤ 基于叶节点标签,通过算法 4 计算 HAT 中所有非叶节点的标签。

⑥ 计算 $\omega \leftarrow \mathrm{HMAC}_e(t_1)$。

⑦ 令 $\boldsymbol{C}=\{c_1,\cdots,c_n\}$ 和 $\boldsymbol{T}=(r,\alpha_s,\boldsymbol{\Gamma},\omega,\boldsymbol{N})$,其中 $\boldsymbol{\Gamma}=\{\tau_1,\cdots,\tau_n\}$,$\boldsymbol{N}=\{\nu_1,v_2,\cdots\}$ 是所有 HAT 节点的集合。

上传阶段结束时,终端将 $\boldsymbol{C}$ 和 $\boldsymbol{T}$ 上传到数据中心,并仅在本地存储 e。

3) 检测下载文件之前数据中心保存文件的完整性。终端和数据中心 S 交互地运行检查协议 $\mathrm{res} \in \{0,1\} \leftarrow \mathrm{Check}\langle S(\boldsymbol{C},\boldsymbol{T}),U(e)\rangle$ 检查数据中心文件的完整性,过程如下:

① U 选择 $b \in [1,n]$,$\kappa \in \{0,1\}^\lambda$,并发送 (b,κ) 到 S。

② 对于每个 $j(1 \leqslant j \leqslant b)$,$S$ 计算 $t_j \leftarrow \pi_k(b)$。然后,数据中心调用算法 5 中的响应算法,其中 $\boldsymbol{I}=\{t_1,\cdots,t_b\}$,并将文件证明 resp 和 (r,v_1,ω) 发送给 U。

③ U 计算 $k \leftarrow r \oplus e$,$\alpha_s \leftarrow \mathrm{KDF}(k,1)$,$k_c \leftarrow \mathrm{KDF}(k,2)$ 和 $\alpha_c \leftarrow \mathrm{KDF}(k,3)$。然后验证 v_1,并调用算法 6 中的验证算法来完成验证。若验证成功则输出 1,否则输出 0。

4) 终端通过调用更新协议 $\mathrm{res} \in \{\langle e^*,(C^*,T^*)\rangle,\perp\} \leftarrow \mathrm{Update}\langle U(e,\iota,m,O_P),S(C,T)\rangle$ 来任意更新文件。如修改块、插入一些块、删除一些块等。

5) 终端将文件的更新块和基于 HAT 更新节点上传到数据中心,终端计算更新后的元数据 e^* 并通过检查协议验证更新的块。

3 实例测试与分析

结合某地方电力公司的电力物联网平台,建立一个由智能终端模拟开发板和数据中心服务器构成的测试系统,如图 3 所示。基于该系统的测试过程如下:

1) 启动 2 个服务程序,运行在服务器上的 2 个服务程序分别监听 4 000 端口和 8 000 端口,如图 4 所示。遍历文件系统,选择一个要同步的文件 shangchuan.txt,文件内容图 5 所示。

2) 服务程序 1 收到文件名,查询数据库未找到该文件,通知终端上传整个文件,如图 6 所示。终端上传原文件的整个文件给服务程序 2,服务程序 2 收到原文件,保存并更新数据库,如图 7 所示。

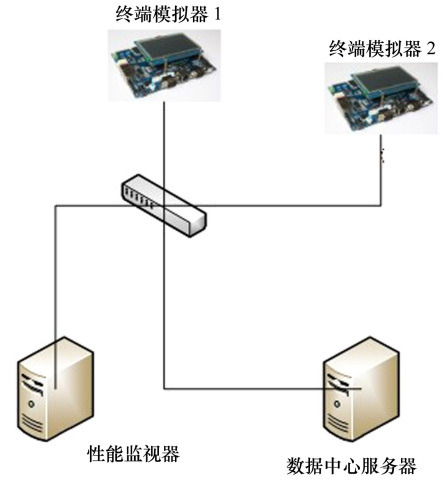


图 3　测试环境

```
itachi@itachi-K46CM:~/clion/server1/cmake-build-debug$ ./server_1
[*]--------Server_1 Started.Listening at Port:4000---------

itachi@itachi-K46CM:~/clion/server1/cmake-build-debug$ ./server_2
[*]--------Server_2 Started.Listening at Port:8000---------
```

图 4　启动服务程序

```
48      fclose(sig_file);
49      result=rs_build_hash_table(sunset);
50      result=rs_delta_file(sunset,new_file,delta_file,&stats);
51      fclose(delta_file);
52      fclose(new_file);
53      return;
54  }
55
56  void test_patch(){
57      rs_result result;
58      rs_stats_t stats;
59      FILE *delta_file=fopen("delta.txt","r");
60      FILE *basis_file=fopen("dst.txt","r");
61      FILE *new_file=fopen("dst.txt","a");
62      result=rs_patch_file(basis_file,delta_file,new_file,&stats);
63      fclose(delta_file);
64      fclose(basis_file);
65      fclose(new_file);
66      return;
67  }
```

图 5　文件内容

```
itachi@itachi-K46CM:~/clion/server1/cmake-build-debug$ ./server_1
[*]--------Server_1 Started.Listening at Port:4000---------

[*]Server:来自 16777343:1739 的连接。
        Receive File Name:shangchuan.txt
        [Database]Opened database successfully
        [Database]Operation done successfully
```

图 6　通知终端上传文件

```
itachi@itachi-K46CM:~/clion/server1/cmake-build-debug$ ./server_2
[*]--------Server_2 Started.Listening at Port:8000---------

[*]Server:来自 16777343:697 的连接。
        收到文件名: shangchuan.txt
        -------处理原文件-----
        Receive File:shangchuan.txt From Client IP Successful!
        [Database]Opened database successfully
        [Database]Operation done successfully
```

图 7　保存并更新数据库

3）首次同步完成,终端上修改文件 shangchuan.txt,即在文件后加入一些内容,再次选择 shangchuan.txt。服务程序 1 收到文件名,查询数据库发现本地已有 shangchuan.txt 文件并对文件分块计算签名,生成签名文件 sig 并发送给终端,如图 8 所示。

```
itachi@itachi-K46CM:~/clion/server1/cmake-build-debug$ ./server_1
[*]--------Server_1 Started.Listening at Port:4000---------

[*]Server:来自 16777343:1739 的连接。
        Receive File Name:shangchuan.txt
        [Database]Opened database successfully
        [Database]Operation done successfully

[*]Server:来自 16777343:2763 的连接。
        Receive File Name:shangchuan.txt
        [Database]Opened database successfully
        [File Found]文件ID: shangchuan.txt     文件目录: ./File/
        [File Found]文件路径: ./File/shangchuan.txt
        --Compute Sig for File:shangchuan.txt--
                [Sig]分块数量: 8
                [Sig]分块长度: 256
        --Compute Sig Successful!--
        File:sig_shangchuan.txt Transfer Successful!
        [Database]Operation done successfully
```

图 8　已有文件计算过程

4）终端收到 sig 文件,利用本地的 shangchuan.txt 文件计算轻量,并将轻量文件上传给服务程序 2。服务程序 2 收到终端发来的轻量文件,与本地 shangchuan.txt 的旧版本文件一起计算新文件,如图 9 所示。

```
[*]Server:来自 16777343:1721 的连接。
        收到文件名: del_shangchuan.txt
        -------处理增量------
        Receive File:del_shangchuan.txt From Client IP Successful!
server_2: (rs_job_new) start patch job
server_2: (rs_scoop_readahead) got 4 bytes from input buffer
server_2: (rs_scoop_advance) advance over 4 bytes from input buffer
server_2: (rs_patch_s_header) got patch magic 0x72730236
server_2: (rs_scoop_readahead) got 1 bytes from input buffer
server_2: (rs_scoop_advance) advance over 1 bytes from input buffer
server_2: (rs_patch_s_cmdbyte) got command byte 0x46 (COPY), len_1=1
server_2: (rs_scoop_readahead) got 3 bytes from input buffer
server_2: (rs_scoop_readahead) got 1 bytes from input buffer
server_2: (rs_scoop_advance) advance over 1 bytes from input buffer
server_2: (rs_scoop_readahead) got 2 bytes from input buffer
server_2: (rs_scoop_advance) advance over 2 bytes from input buffer
server_2: (rs_patch_s_run) running command 0x46, kind 1003
server_2: (rs_patch_s_copy) COPY(where=0, len=1792)
server_2: (rs_infilebuf_fill) seen end of file on input
server_2: (rs_patch_s_copying) copy 1792 bytes from basis at offset 0
server_2: (rs_patch_s_copying) copy callback returned OK
server_2: (rs_patch_s_copying) got 1792 bytes back from basis callbac
k
```

图 9　计算新文件过程

5）轻量更新与验证完成。更新的性能展示通过终端将性能数据发送到服务器的数据库中,由界面展示出更新的各项数据,如更新类型、文件名、文件大小、sig 文件大小、传输文件大小以及更新时间。32M 文件更新 8M(25%)时的性能如图 10 所示。

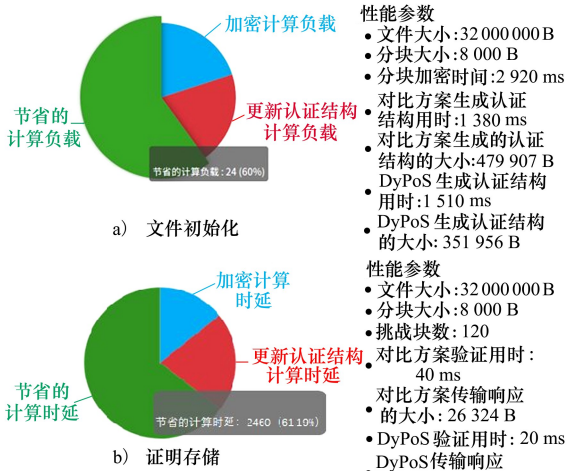


图 10　性能展示

由此可见：

1）32 MB 文件进行 8 kB 分块时，认证结构大小较 Merkle 树[12]对比方案减少 26.7%；挑战 120 个块，DyPoS 较 Merkle 树[12]对比方案响应用时减少 57.3%，验证用时减少 50%。

2）加密文件修改内容 18%时，更新该加密文件的计算负载降低 68.4%，时延降低 68.9%；加密文件修改内容 25%时，更新该加密文件的计算负载降低 60%，时延降低 61.2%。

4 结 论

文件同步技术是电力物联网应用的关键技术之一。基于同步时传输 2 个文件版本之间不同的文件数据，而重用相同部分的思想，通过对密文形成压缩标签的方式对数据中心存储内容进行效验，无需下载完整密文文件，在保障安全的前提下可以有效降低文件内容完整性效验的性能开销。针对标签的验证问题，设计了轻量快速更新密文的存储方法，提出了同态认证树 HAT 的新型认证结构，与传统的认证结构相比，有效地减少了终端生成认证结构时间，降低了校验数据内容时的通信开销和计算开销。

附录

算法 1 路径搜索算法

1: procedure Path(T,I)
2: for $\iota \in I$ do
3: if $\iota > l_1$ then
4: return 0
5: $i_\iota \leftarrow 1$, $ord_\iota \leftarrow \iota$
6: $\rho \leftarrow \{1\}$, $S \leftarrow$TRUE
7: while S do
8: $S \leftarrow$FALSE
9: for $\iota \in I$ do
10: if $l_{i_\iota} = 1$ then
11: continue
12: else if $ord_\iota \leqslant l_{2i_\iota}$ then
13: $i_\iota \leftarrow 2i_\iota$
14: else
15: $ord_\iota \leftarrow ord_\iota \leftarrow l_{2i_\iota}, i_\iota \leftarrow 2i_\iota + 1$
16: $\rho \leftarrow \rho \cup \{i_\iota\}$
17: if $\iota_{i_\iota} > 1$ then
18: $S \leftarrow$TRUE
19: return ρ

算法 2 兄弟搜索算法

1: procedure Sibling(ρ)
2: $\psi \leftarrow \phi, \rho \leftarrow \rho \setminus \{1\}, \rho \leftarrow \phi, i \leftarrow 1$
3: while $\rho \neq \phi \vee \rho \neq \phi$ do
4: if $2i \in \rho$ then
5: $i \leftarrow 2i, \rho \leftarrow \rho \setminus \{i\}$
6: if $i+1 \in \rho$ then
7: $\rho \leftarrow \rho \cup \{(i+1, \text{FALSE})\}, \rho \leftarrow \rho \setminus \{i+1\}$
8: else
9: $\rho \leftarrow \rho \cup \{(i+1, \text{TRUE})\}$
11: else if $2i+1 \in \rho$ then
12: $i \leftarrow 2i+1, \rho \leftarrow \rho \setminus \{i\}, \psi \leftarrow \psi \cup \{i-1\}$
13: else if $\rho \neq \phi$ then
14: pop the last inserted (α, β) in ρ
15: $i \leftarrow \alpha$
16: if $\beta =$TRUE then
17: $\psi \leftarrow \psi \cup \{i\}$
21: return ψ

算法 3 叶子节点标签生成算法

1: procedure LeafTag($\alpha_s, k_c, \alpha_c, c_\iota, i_\iota, l_{i_\iota}, \nu_{i_\iota}$)
2: $\tau_\iota \leftarrow \alpha_s c_\iota$
3: $t_{i_\iota} \leftarrow f_{k_c}(i_\iota \parallel l_{i_\iota} \parallel \nu_{i_\iota}) + \alpha_c \tau_\iota$
4: return τ_ι, t_{i_ι}

算法 4 非叶子节点标签生成算法

1: procedure NonLeafTag(k_c, i, l_i, ν_i)
2: $\tau_{2i} \leftarrow t_{2i} - f_{k_c}(2i \parallel l_{2i} \parallel \nu_{2i})$
3: $\tau_{2i+1} \leftarrow t_{2i+1} - f_{k_c}(2i+1 \parallel l_{2i+1} \parallel \nu_{2i+1})$
4: return $t_i \leftarrow f_{k_c}(i \parallel l_i \parallel \nu_i) + \tau_{2i} + \tau_{2i+1}$

算法 5 响应算法

1: procedure Response(I)
2: $\rho \leftarrow P_{\text{ATH}} T, I, \psi \leftarrow$Sibling ρ
3: $c \leftarrow 0, t \leftarrow 0$
4: for $\iota \in I$ do
5: $c \leftarrow c + c_\iota$
6: for $i \in \psi$ do
7: $t \leftarrow t + t_i$
8: return $resp \leftarrow (c, t, \{\nu_{i_\iota} \mid \iota \in I\}, \{(i, l_i, u_i) \mid i \in \psi\})$

算法 6 验证算法

1: procedure Verify($\alpha_s, k_c, \alpha_c, \nu_1, I, resp$)

2: $ctr \leftarrow 1, \tau \leftarrow 0$
3: for $\iota \in I$ do
4: while $ctr < \iota$ do
5: pop the first element in $\{(i, l_i, \nu_i) \mid i \in \psi\}$
6: $ctr \leftarrow ctr + l_i, \tau \leftarrow \tau + f_{k_c}(i \| l_i \| \nu_i)$
7: if $ctr \neq \iota$ then
8: return 0
9: else
10: $ctr \leftarrow ctr + 1$
11: for $(i, l_i, \nu_i) \in \{(i, l_i, \nu_i) \mid i \in \psi\}$ do
12: $ctr \leftarrow ctr + l_i, \tau \leftarrow \tau + f_{k_c}(i \| l_i \| \nu_i)$
13: if $ctr \neq n+1$ then
14: return 0
15: else if $t + f_{k_c}(1 \| l_1 \| \nu_1) - \tau + \alpha_s \alpha_c c \neq t_1$ then
16: return 0
17: else
18: return 1

参考文献:

[1] LI Xin, LAI Ji, CHEN Zhongtao, et al. Intelligent fault detection system for state grid IoT equipment based on edge computing [C]//2019 IEEE 3rd International Electrical and Energy Conference, 2019: 1434-1439

[2] ATENIESE G, BURNS R, CURTMOLA R, et al. Provable data possession at untrusted stores[C]//Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007: 598-609

[3] ARMKNECHT F, BOHLI J M, KARAME G O, et al. Outsourced proofs of retrievability[C]//Proceedings of the 21st ACM Conference on Computer and Communications Security, 2014: 831 - 843

[4] SHACHAM H, WATERS B. Compact proofs of retrievability[C]//Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, 2008

[5] DODIS Y, VADHAN S, WICHS D. Proofs of retrievability via hardness amplification[C]//Proceedings of the 6th Theory of Cryptography Conference, 2009

[6] BOWERS K D, JUELS A, OPREA A. HAIL: A high-availability and integrity layer for cloud storage[C]//Proceedings of 14th ACM Conference on Computer and Communications Security, 2009: 187-198

[7] WANG C, WANG Q, REN K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]//IEEE International Conference on Computer Communications, 2010: 1-9

[8] ATENIESE G, BURNS R, CURTMOLA R, et al. Remote data checking using provable data possession[J]. ACM Transactions on Information and System Security, 2011, 14(1): 1-34

[9] XU J, CHANG E C. Towards efficient proofs of retrievability[C]//Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 2012: 79-80

[10] CHEN J, PENG Y, DU R, et al. Regeneratingcodes-based efficient remote data checking and repairing in cloud storage[C]//Proceedings of 2015 IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2015: 143-150

[11] SHI E, STEFANOV E, PAPAMANTHOU C. Practical dynamic proofs of retrievability[C]//Proceedings of 20th ACM Conference on Computer and Communications Security, 2013: 325 - 336

[12] REN Z, WANG L, WANG Q, et al. Dynamic proofs of retrievability for coded cloud storage systems[J]. IEEE Trans on Serv Comput, 2018, 11(4): 685-698

A method for identity authentication and data protection of terminal storage devices of power internet of things

HE Pei[1], ZHENG Wenbin[2], CHI Xiaojin[2], CAI Yiting[2], YAO Hongjing[1]

(1.School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China;
2.State Grid Wenzhou Electric Power Co., Ltd, Wenzhou 325000, China)

Abstract: The communication link between the intelligent terminal of the power internet of things and its data center has the characteristics of low bandwidth and high delay. Combined with the security requirements of data synchronization and sharing in the ubiquitous internet of things network environment, based on the goal of minimizing the transmission cost of updated files under the conditions of limited communication resources, this paper designs a method for dynamic secure storage and verification of lightweight information to realize the information security of a terminal storage device. The method verifies the integrity of secret files stored in the data center through dynamic proof of storage (DyPoS) technology and supports their dynamic modification. At the same time, it supports the incremental update of secret files. For modified files, only modified data blocks are transmitted to the data center to complete the update without the need to upload the whole file again, thus effectively reducing the calculation overhead of verification and the communication overhead of transmission.

Keywords: power internet of things; terminal; identity authentication; data protection; dynamic proof of storage; security

引用格式: 何佩, 郑文斌, 池晓金, 等. 电力物联网终端存储设备身份认证与数据保护方法研究[J]. 西北工业大学学报, 2022,40(5): 1188-1194

HE Pei, ZHENG Wenbin, CHI Xiaojin, et al. A method for identity authentication and data protection of terminal storage devices of power internet of things[J]. Journal of Northwestern Polytechnical University, 2022, 40(5): 1188-1194 (in Chinese)

© 2022 Journal of Northwestern Polytechnical University.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.