```
define: HandleInstForCollection(inst, lock_set, alias_graph, key_fields)
alias_graph′ := UpdateAliasGraph(alias_graph, inst);
lock_set′ := UpdateLockSet(inst, lock_set, alias_graph′);
var := GetOperand(inst);
var_node := GetAliasNode(var, alias_graph′);
switch typeof(inst):
    case write:
    case read:
        foreach lock_node in lock_set′ do
            <var_access_path, lock_access_path> :=
                        GetProtectedFieldAccess(
                        var_node, lock_node, alias_graph′);
            if var_access_path is not NULL then
                key_fields := key_fields $\cup$ {var_access_path};
            end if
        end foreach
        break;
    case call:
        called_func := GetCalledFunc(inst);
        < lock_set′, alias_graph′, key_fields′> :=
                        HandleFuncForCollection(
                        lock_set′, alias_graph′, key_fields, called_func);
        key_fields := key_fields $\cup$ key_fields′;
    break;
end switch
return <lock_set′, alias_graph′, key_fields>;
```

```
define: HandleFuncForCollection(func, lock_set, alias_graph, key_fields)
foreach code_path in GetCodePath(func) do
    foreach inst in GetInstructions(code_path) do
        <lock_set′, alias_graph′, key_fields′> :=
                        HandleInstForCollection(inst, lock_set, alias_graph);
    end foreach
end foreach
return <lock_set′, alias_graph′, key_fields′>;
```

```
define: CollectKeyField()
lock_set := $\emptyset$;
alias_graph := $\emptyset$;
key_fields := $\emptyset$;
foreach func in OS code without a caller function do
    <lock_set, alias_graph, key_fields> :=
                        HandleFuncForCollection(
                        func, lock_set, alias_graph, key_fields);
end foreach
return key_fields;
```