Formalisation of the key manager

This Event-B model is an attempt to formalise the requirements of the VerifyThis Long Term Challenge 2020[1]. It is deliberately kept on a very high level of abstraction, but should have enough detail to clarify the behaviour of the system.

The following operations (aka events) are permitted:

getExisting Retrieve an existing key from a database. The email must be associated to at least one key. (Note that there is an indetermistic result if more than one key is registered for an e-mail address.)

requestAdd Request adding an email+key pair that is not yet in the database. A non-existing confirmation code is returned, but the pair is not added yet to the database.

confirmAdd Once the request has been issued, an email with the code is sent out to the issuer. If they confirm this code, the addition to the database will take place.

requestDel An existing email+key pair can also be removed from the database by the owner. When requesting a removal, the pair is not yet removed from the database, but a confirmation code is issued.

confirmDel Once the request has been issued, an email with the code is sent out to the issuer. If they confirm this code, the deletion from the database will take place.

rerequestDel An open removal request can be reissued, resulting in the same confirmation code.

Author: Mattias Ulbrich `<ulbrich@kit.edu>`.
Version: 1.0, 2019-09-10

[1] see `https://verifythis.github.io/`

CONTEXT Data
SETS
KEY
EMAIL
CONFIRMATION_CODE
END

MACHINE KeyServer

SEES Data

VARIABLES

database

openAdds

openDels

INVARIANTS

databaseType: $database \in EMAIL \leftrightarrow KEY$

openAddsType: $openAdds \in CONFIRMATION\_CODE \pfun (EMAIL \times KEY)$

openDelsType: $openDels \in CONFIRMATION\_CODE \pfun (EMAIL \times KEY)$

noSpuriousDels: $ran(openDels) \subseteq database$

disjointConfirms: $dom(openDels) \cap dom(openAdds) = \varnothing$

uniqueDels: $\forall x, y \cdot x \in dom(openDels) \land y \in dom(openDels) \Rightarrow openDels(x) \neq openDels(y) \lor x = y$

EVENTS

Initialisation

begin

act1: $database := \varnothing$

act2: $openAdds, openDels := \varnothing, \varnothing$

end

Event getExisting ⟨ordinary⟩ $\widehat{=}$

any

email IN

result OUT

where

guard: $email \in dom(database)$

result: $result \in database[\{email\}]$

then

skip

end

Event requestAdd ⟨ordinary⟩ $\widehat{=}$

any

email IN

key IN

conf_code OUT

where

grd1: $email \mapsto key \notin database$

grd2: $conf\_code \notin dom(openAdds) \cup dom(openDels)$

then

act1: $openAdds(conf\_code) := email \mapsto key$

end

Event confirmAdd ⟨ordinary⟩ $\widehat{=}$

any

conf_code IN

where

grd1: $conf\_code \in dom(openAdds)$

then

act1: $openAdds := \{conf\_code\} \domsub openAdds$

act2: $database := database \cup openAdds[\{conf\_code\}]$

end

Event requestDel ⟨ordinary⟩ $\widehat{=}$

any

email IN
key IN
conf_code OUT

where

grd1: $email \mapsto key \in database$
grd2: $conf\_code \notin dom(openAdds) \cup dom(openDels)$
grd3: $email \mapsto key \notin ran(openDels)$

then

act1: $openDels(conf\_code) := email \mapsto key$

end

Event confirmDel ⟨ordinary⟩ $\widehat{=}$

any

conf_code IN

where

grd1: $conf\_code \in dom(openDels)$

then

act1: $openDels := \{conf\_code\} \lhd openDels$
act2: $database := database \setminus openDels[\{conf\_code\}]$

end

Event rerequestDel ⟨ordinary⟩ $\widehat{=}$

any

email IN
key IN
conf_code OUT

where

grd1: $email \mapsto key \in database$
grd2: $email \mapsto key \in ran(openDels)$
grd3: $conf\_code \mapsto (email \mapsto key) \in openDels$

then

skip

end

END