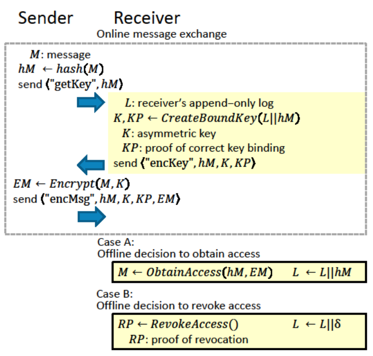
Inserting Intentional Bugs for Model Checking Assurance

Thomas L. Rodeheffer and Ramakrishna Kotla

Microsoft Research, Silicon Valley

The Problem

- Complicated protocol ("Pasture")



Sender | Receiver

Online message exchange

M: message
$hM \leftarrow hash(M)$
send ("getKey", hM)

L: receiver's append–only log
$K, KP \leftarrow CreateBoundKey(L||hM)$
K: asymmetric key
KP: proof of correct key binding
send ("encKey", hM, K, KP)

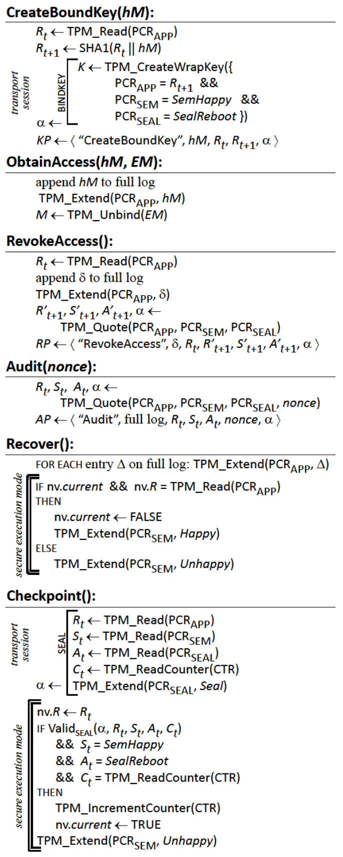
$EM \leftarrow Encrypt(M, K)$
send ("encMsg", hM, K, KP, EM)

Case A:
Offline decision to obtain access

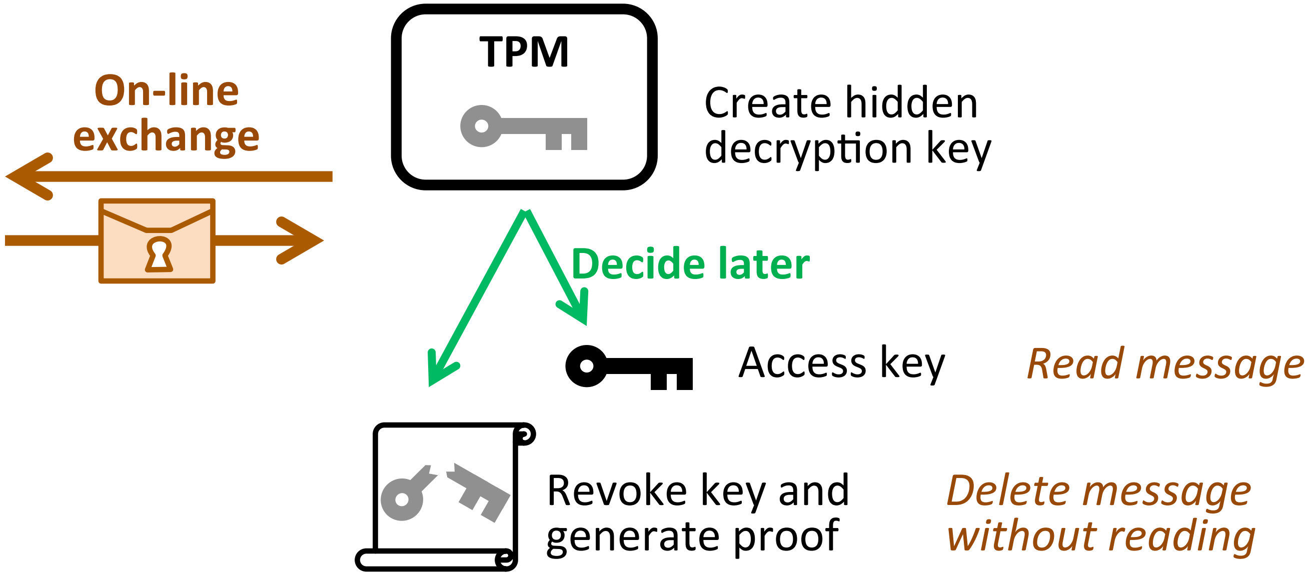
$M \leftarrow ObtainAccess(hM, EM)$ $\quad L \leftarrow L||hM$

Case B:
Offline decision to revoke access

$RP \leftarrow RevokeAccess()$ $\quad L \leftarrow L||\delta$
RP: proof of revocation

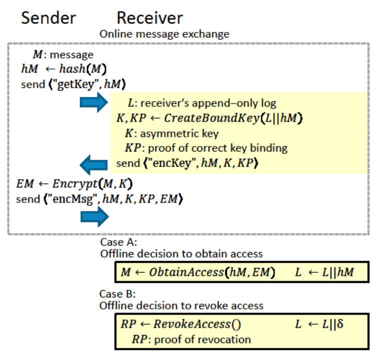


```
CreateBoundKey(hM):
    $R_t \leftarrow$ TPM_Read(PCR$_{APP}$)
    $R_{t+1} \leftarrow$ SHA1($R_t$ || $hM$)
  transport session
        BINDKEY [ $K \leftarrow$ TPM_CreateWrapKey({
                        PCR$_{APP}$ = $R_{t+1}$  &&
                        PCR$_{SEM}$ = $SemHappy$  &&
                        PCR$_{SEAL}$ = $SealReboot$ })
    $\alpha \leftarrow$
    $KP \leftarrow \langle$ "CreateBoundKey", $hM$, $R_t$, $R_{t+1}$, $\alpha$ $\rangle$
ObtainAccess(hM, EM):
    append $hM$ to full log
    TPM_Extend(PCR$_{APP}$, $hM$)
    $M \leftarrow$ TPM_Unbind($EM$)
RevokeAccess():
    $R_t \leftarrow$ TPM_Read(PCR$_{APP}$)
    append $\delta$ to full log
    TPM_Extend(PCR$_{APP}$, $\delta$)
    $R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, $\alpha \leftarrow$
        TPM_Quote(PCR$_{APP}$, PCR$_{SEM}$, PCR$_{SEAL}$)
    $RP \leftarrow \langle$ "RevokeAccess", $\delta$, $R_t$, $R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, $\alpha$ $\rangle$
Audit(nonce):
    $R_t$, $S_t$, $A_t$, $\alpha \leftarrow$
        TPM_Quote(PCR$_{APP}$, PCR$_{SEM}$, PCR$_{SEAL}$, $nonce$)
    $AP \leftarrow \langle$ "Audit", full log, $R_t$, $S_t$, $A_t$, $nonce$, $\alpha$ $\rangle$
Recover():
    FOR EACH entry $\Delta$ on full log: TPM_Extend(PCR$_{APP}$, $\Delta$)
  secure execution mode
    [ IF nv.current && nv.R = TPM_Read(PCR$_{APP}$)
      THEN
          nv.current $\leftarrow$ FALSE
          TPM_Extend(PCR$_{SEM}$, $Happy$)
      ELSE
          TPM_Extend(PCR$_{SEM}$, $Unhappy$)
Checkpoint():
  transport session
        SEAL [ $R_t \leftarrow$ TPM_Read(PCR$_{APP}$)
               $S_t \leftarrow$ TPM_Read(PCR$_{SEM}$)
               $A_t \leftarrow$ TPM_Read(PCR$_{SEAL}$)
               $C_t \leftarrow$ TPM_ReadCounter(CTR)
    $\alpha \leftarrow$ TPM_Extend(PCR$_{SEAL}$, $Seal$)
  secure execution mode
    [ nv.R $\leftarrow R_t$
      IF Valid$_{SEAL}$($\alpha$, $R_t$, $S_t$, $A_t$, $C_t$)
          && $S_t$ = $SemHappy$
          && $A_t$ = $SealReboot$
          && $C_t$ = TPM_ReadCounter(CTR)
      THEN
          TPM_IncrementCounter(CTR)
          nv.current $\leftarrow$ TRUE
      TPM_Extend(PCR$_{SEM}$, $Unhappy$)
```


On-line exchange
TPM
Create hidden decryption key
Decide later
Access key
Read message
Revoke key and generate proof
Delete message without reading

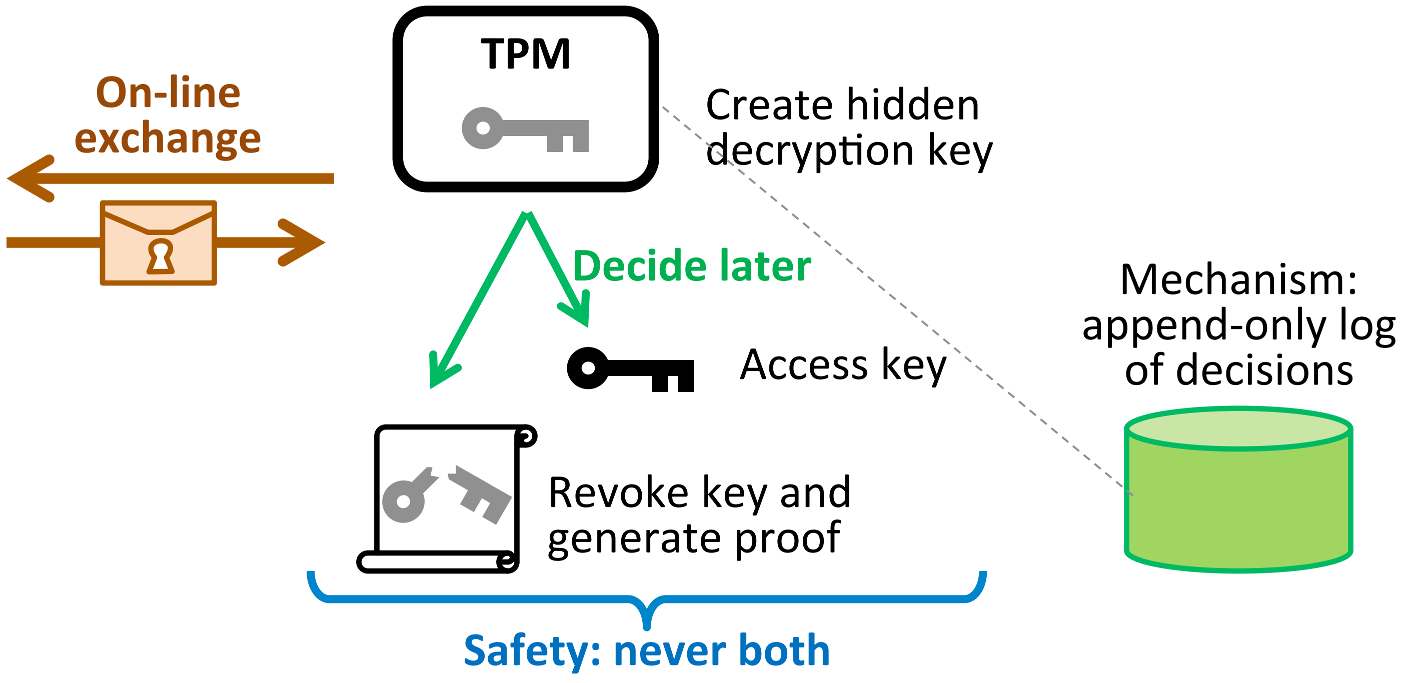
The Problem

- Complicated protocol (“Pasture”)
- Important safety properties



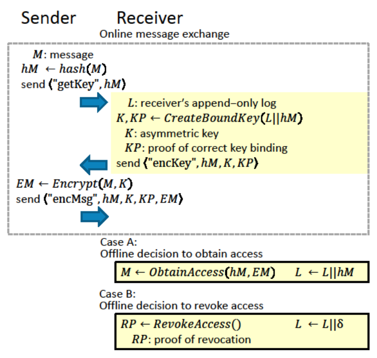
Sender
Receiver
Online message exchange
M: message
$hM \leftarrow hash(M)$
send ("getKey", hM)
L: receiver's append–only log
$K, KP \leftarrow CreateBoundKey(L||hM)$
K: asymmetric key
KP: proof of correct key binding
send ("encKey", hM, K, KP)
$EM \leftarrow Encrypt(M, K)$
send ("encMsg", hM, K, KP, EM)
Case A:
Offline decision to obtain access
$M \leftarrow ObtainAccess(hM, EM)$ $L \leftarrow L||hM$
Case B:
Offline decision to revoke access
$RP \leftarrow RevokeAccess()$ $L \leftarrow L||\delta$
RP: proof of revocation

```
CreateBoundKey(hM):
    R_t ← TPM_Read(PCR_APP)
    R_{t+1} ← SHA1(R_t || hM)
  transport session
        BINDKEY
        [ K ← TPM_CreateWrapKey({
                PCR_APP = R_{t+1}  &&
                PCR_SEM = SemHappy  &&
                PCR_SEAL = SealReboot })
    α ← 
    KP ← ⟨ “CreateBoundKey”, hM, R_t, R_{t+1}, α ⟩
ObtainAccess(hM, EM):
    append hM to full log
     TPM_Extend(PCR_APP, hM)
    M ← TPM_Unbind(EM)
RevokeAccess():
    R_t ← TPM_Read(PCR_APP)
    append δ to full log
    TPM_Extend(PCR_APP, δ)
    R'_{t+1}, S'_{t+1}, A'_{t+1}, α ←
        TPM_Quote(PCR_APP, PCR_SEM, PCR_SEAL)
    RP ← ⟨ “RevokeAccess”, δ, R_t, R'_{t+1}, S'_{t+1}, A'_{t+1}, α ⟩
Audit(nonce):
    R_t, S_t, A_t, α ←
        TPM_Quote(PCR_APP, PCR_SEM, PCR_SEAL, nonce)
    AP ← ⟨ “Audit”, full log, R_t, S_t, A_t, nonce, α ⟩
Recover():
    FOR EACH entry Δ on full log: TPM_Extend(PCR_APP, Δ)
  secure execution mode
  [ IF nv.current  &&  nv.R = TPM_Read(PCR_APP)
    THEN
        nv.current ← FALSE
        TPM_Extend(PCR_SEM, Happy)
    ELSE
        TPM_Extend(PCR_SEM, Unhappy)
Checkpoint():
  transport session
        SEAL
        [ R_t ← TPM_Read(PCR_APP)
          S_t ← TPM_Read(PCR_SEM)
          A_t ← TPM_Read(PCR_SEAL)
          C_t ← TPM_ReadCounter(CTR)
    α ←   TPM_Extend(PCR_SEAL, Seal)
  secure execution mode
  [ nv.R ← R_t
    IF Valid_SEAL(α, R_t, S_t, A_t, C_t)
        &&  S_t = SemHappy
        &&  A_t = SealReboot
        &&  C_t = TPM_ReadCounter(CTR)
    THEN
        TPM_IncrementCounter(CTR)
        nv.current ← TRUE
    TPM_Extend(PCR_SEM, Unhappy)
```



TPM
On-line exchange
Create hidden decryption key
Decide later
Access key
Revoke key and generate proof
Mechanism: append-only log of decisions
Safety: never both

The Problem



Sender Receiver

Online message exchange

```
M: message
hM ← hash(M)
send ("getKey", hM)
        L: receiver's append–only log
        K, KP ← CreateBoundKey(L||hM)
        K: asymmetric key
        KP: proof of correct key binding
        send ("encKey", hM, K, KP)
EM ← Encrypt(M, K)
send ("encMsg", hM, K, KP, EM)
```

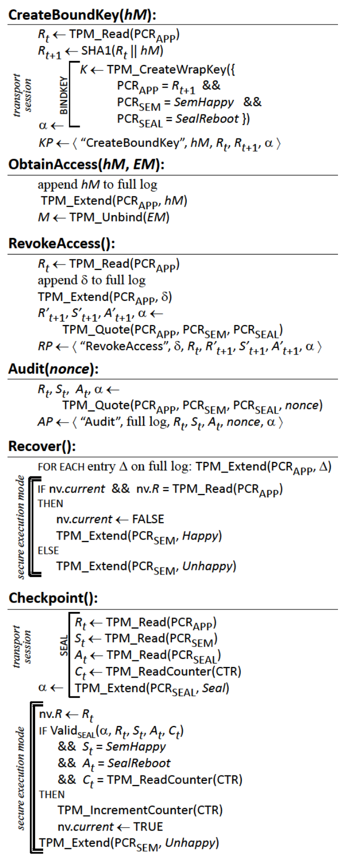
Case A:
Offline decision to obtain access

$M \leftarrow ObtainAccess(hM, EM)$ $L \leftarrow L||hM$

Case B:
Offline decision to revoke access

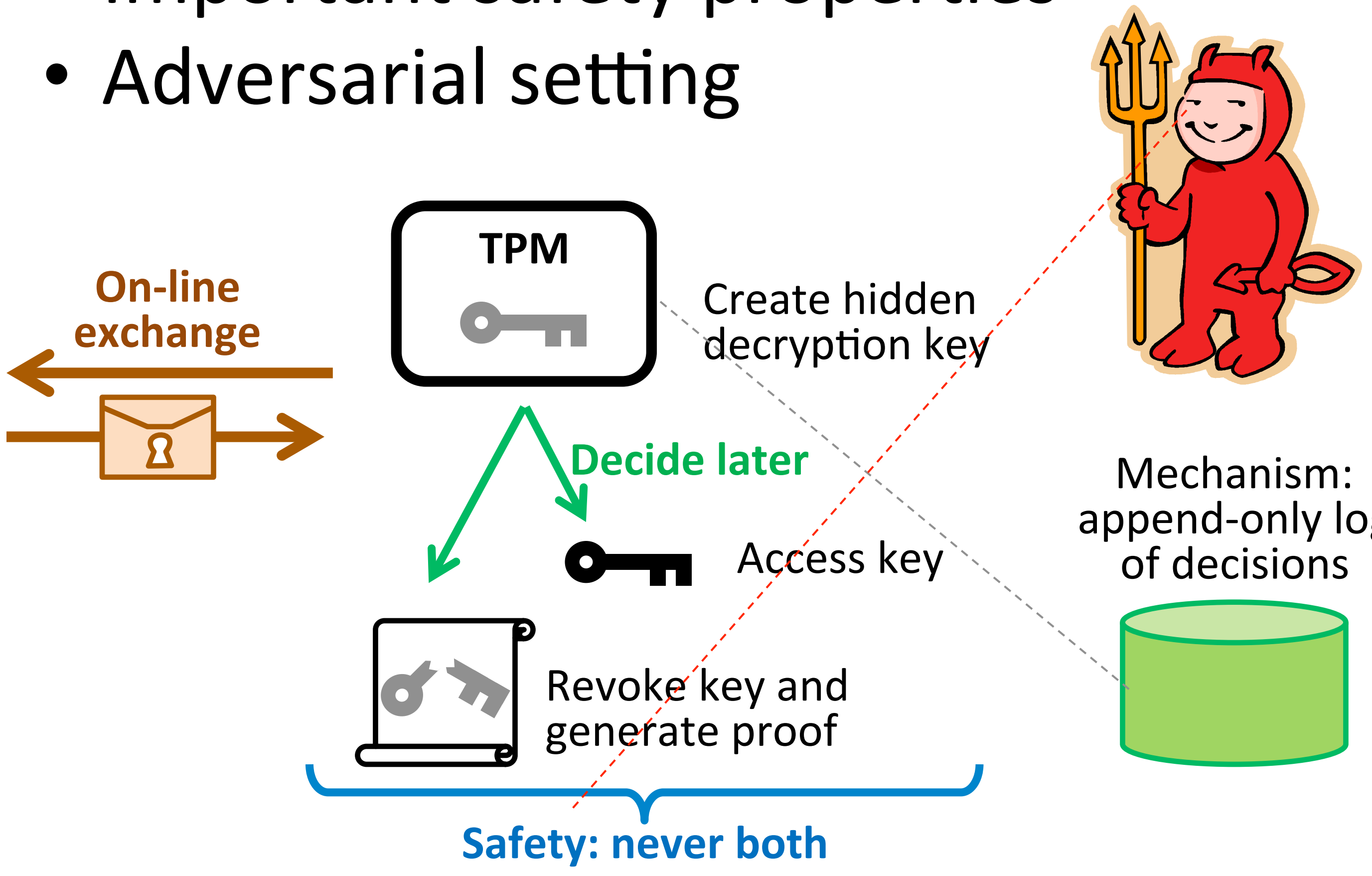
$RP \leftarrow RevokeAccess()$ $L \leftarrow L||\delta$

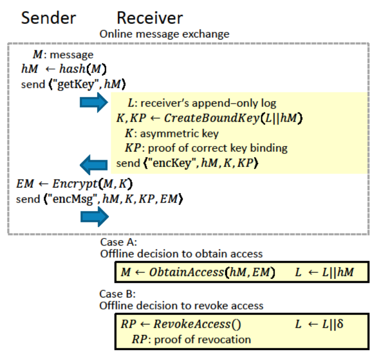
RP: proof of revocation



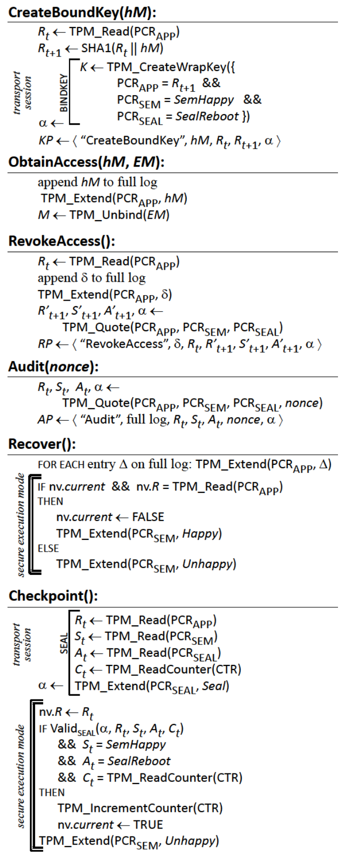
```
CreateBoundKey(hM):
    $R_t$ ← TPM_Read($PCR_{APP}$)
    $R_{t+1}$ ← SHA1($R_t$ || hM)
    transport session
        BINDKEY
            K ← TPM_CreateWrapKey({
                    $PCR_{APP} = R_{t+1}$ &&
                    $PCR_{SEM}$ = SemHappy &&
                    $PCR_{SEAL}$ = SealReboot })
    α ←
    KP ← ⟨ "CreateBoundKey", hM, $R_t$, $R_{t+1}$, α ⟩

ObtainAccess(hM, EM):
    append hM to full log
    TPM_Extend($PCR_{APP}$, hM)
    M ← TPM_Unbind(EM)

RevokeAccess():
    $R_t$ ← TPM_Read($PCR_{APP}$)
    append δ to full log
    TPM_Extend($PCR_{APP}$, δ)
    $R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, α ←
        TPM_Quote($PCR_{APP}$, $PCR_{SEM}$, $PCR_{SEAL}$)
    RP ← ⟨ "RevokeAccess", δ, $R_t$, $R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, α ⟩

Audit(nonce):
    $R_t$, $S_t$, $A_t$, α ←
        TPM_Quote($PCR_{APP}$, $PCR_{SEM}$, $PCR_{SEAL}$, nonce)
    AP ← ⟨ "Audit", full log, $R_t$, $S_t$, $A_t$, nonce, α ⟩

Recover():
    FOR EACH entry Δ on full log: TPM_Extend($PCR_{APP}$, Δ)
    secure execution mode
        IF nv.current && nv.R = TPM_Read($PCR_{APP}$)
        THEN
            nv.current ← FALSE
            TPM_Extend($PCR_{SEM}$, Happy)
        ELSE
            TPM_Extend($PCR_{SEM}$, Unhappy)

Checkpoint():
    transport session
        SEAL
            $R_t$ ← TPM_Read($PCR_{APP}$)
            $S_t$ ← TPM_Read($PCR_{SEM}$)
            $A_t$ ← TPM_Read($PCR_{SEAL}$)
            $C_t$ ← TPM_ReadCounter(CTR)
    α ←     TPM_Extend($PCR_{SEAL}$, Seal)
    secure execution mode
        nv.R ← $R_t$
        IF $Valid_{SEAL}$(α, $R_t$, $S_t$, $A_t$, $C_t$)
            && $S_t$ = SemHappy
            && $A_t$ = SealReboot
            && $C_t$ = TPM_ReadCounter(CTR)
        THEN
            TPM_IncrementCounter(CTR)
            nv.current ← TRUE
        TPM_Extend($PCR_{SEM}$, Unhappy)
```

- Complicated protocol ("Pasture")
- Important safety properties
- Adversarial setting


On-line exchange
TPM
Create hidden decryption key
Decide later
Access key
Revoke key and generate proof
Safety: never both
Mechanism: append-only log of decisions



Sender
Receiver
Online message exchange
M: message
$hM \leftarrow hash(M)$
send ("getKey", hM)
L: receiver's append–only log
$K, KP \leftarrow CreateBoundKey(L||hM)$
K: asymmetric key
KP: proof of correct key binding
send ("encKey", hM, K, KP)
$EM \leftarrow Encrypt(M, K)$
send ("encMsg", hM, K, KP, EM)
Case A:
Offline decision to obtain access
$M \leftarrow ObtainAccess(hM, EM)$ $L \leftarrow L||hM$
Case B:
Offline decision to revoke access
$RP \leftarrow RevokeAccess()$ $L \leftarrow L||\delta$
RP: proof of revocation



CreateBoundKey(*hM*):

```
$R_t \leftarrow$ TPM_Read($PCR_{APP}$)
$R_{t+1} \leftarrow$ SHA1($R_t$ || $hM$)
transport session
    BINDKEY
    $K \leftarrow$ TPM_CreateWrapKey({
            $PCR_{APP} = R_{t+1}$ &&
            $PCR_{SEM} = SemHappy$ &&
            $PCR_{SEAL} = SealReboot$ })
$\alpha \leftarrow$
$KP \leftarrow \langle$ "CreateBoundKey", $hM$, $R_t$, $R_{t+1}$, $\alpha \rangle$
```

ObtainAccess(*hM*, *EM*):

```
append $hM$ to full log
TPM_Extend($PCR_{APP}$, $hM$)
$M \leftarrow$ TPM_Unbind($EM$)
```

RevokeAccess():

```
$R_t \leftarrow$ TPM_Read($PCR_{APP}$)
append $\delta$ to full log
TPM_Extend($PCR_{APP}$, $\delta$)
$R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, $\alpha \leftarrow$
    TPM_Quote($PCR_{APP}$, $PCR_{SEM}$, $PCR_{SEAL}$)
$RP \leftarrow \langle$ "RevokeAccess", $\delta$, $R_t$, $R'_{t+1}$, $S'_{t+1}$, $A'_{t+1}$, $\alpha \rangle$
```

Audit(*nonce*):

```
$R_t$, $S_t$, $A_t$, $\alpha \leftarrow$
    TPM_Quote($PCR_{APP}$, $PCR_{SEM}$, $PCR_{SEAL}$, $nonce$)
$AP \leftarrow \langle$ "Audit", full log, $R_t$, $S_t$, $A_t$, $nonce$, $\alpha \rangle$
```

Recover():

```
FOR EACH entry $\Delta$ on full log: TPM_Extend($PCR_{APP}$, $\Delta$)
secure execution mode
    IF nv.$current$ && nv.$R$ = TPM_Read($PCR_{APP}$)
    THEN
        nv.$current \leftarrow$ FALSE
        TPM_Extend($PCR_{SEM}$, $Happy$)
    ELSE
        TPM_Extend($PCR_{SEM}$, $Unhappy$)
```

Checkpoint():

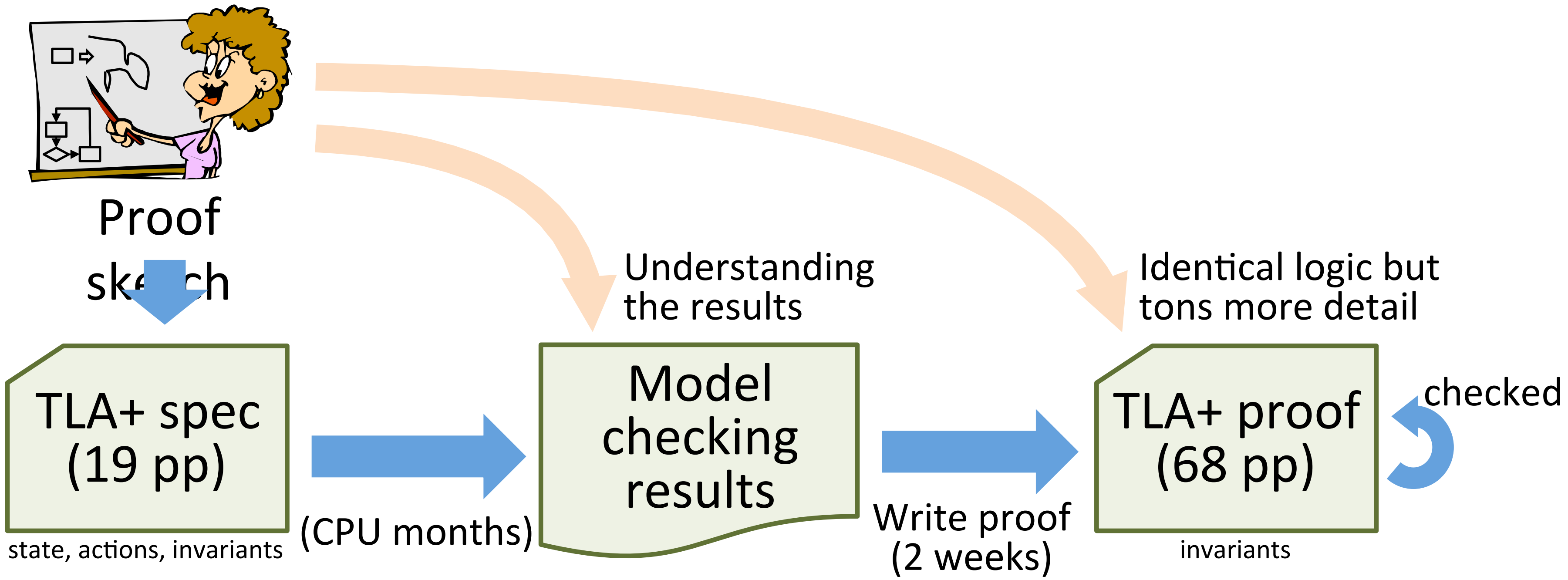
```
transport session
    SEAL
    $R_t \leftarrow$ TPM_Read($PCR_{APP}$)
    $S_t \leftarrow$ TPM_Read($PCR_{SEM}$)
    $A_t \leftarrow$ TPM_Read($PCR_{SEAL}$)
    $C_t \leftarrow$ TPM_ReadCounter(CTR)
$\alpha \leftarrow$ TPM_Extend($PCR_{SEAL}$, $Seal$)
secure execution mode
    nv.$R \leftarrow R_t$
    IF $Valid_{SEAL}$($\alpha$, $R_t$, $S_t$, $A_t$, $C_t$)
        && $S_t = SemHappy$
        && $A_t = SealReboot$
        && $C_t$ = TPM_ReadCounter(CTR)
    THEN
        TPM_IncrementCounter(CTR)
        nv.$current \leftarrow$ TRUE
    TPM_Extend($PCR_{SEM}$, $Unhappy$)
```

The Solution

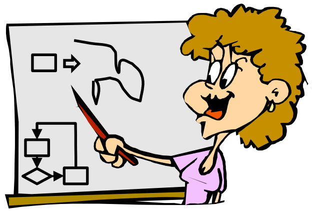
- Complicated protocol ("Pasture")
- Important safety properties
- Adversarial setting

- Solution: Use formal methods
  - Specification – *is it correct?*
  - Model checking – *was it enough?*
  - Formal proof – *too hard?*

What we did for Pasture

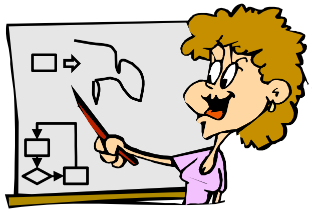


Pretty sure it is correct



Proof sketch

Pretty sure we covered everything

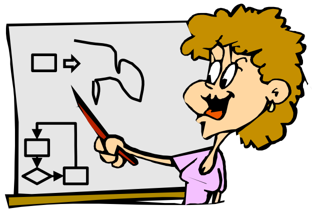


Proof sketch

TLA+ spec
(19 pp)

state, actions, invariants

Model checking – was it enough?



Proof sketch

TLA+ spec
(19 pp)

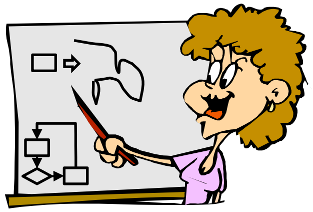
state, actions, invariants

Several CPU months later…

"no errors found"

Cannot model check any larger configurations using TLC because such configurations have more than 2^{32} distinct states – making state fingerprint collision a near certainty.

Insert some intentional bugs



Proof sketch

Easy methodology: Find an action that seems important and omit it

TLA+ spec (19 pp)

state, actions, invariants

Several CPU months later…

"no errors found"

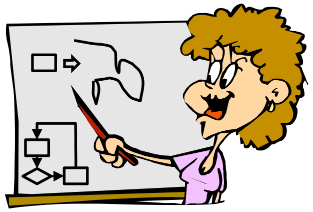
+ intentional bugs

TLA+ spec (19 pp)

A few CPU minutes…

Bug1	violation example
Bug2	violation example
Bug3	violation example
…	…
Bug12	violation example
Bug13	violation example
Bug14	no error
Bug15	no error
Bug16	no error

Not all bugs violate safety



Proof sketch

TLA+ spec (19 pp)

state, actions, invariants

Several CPU months later…

“no errors found”

+ intentional bugs

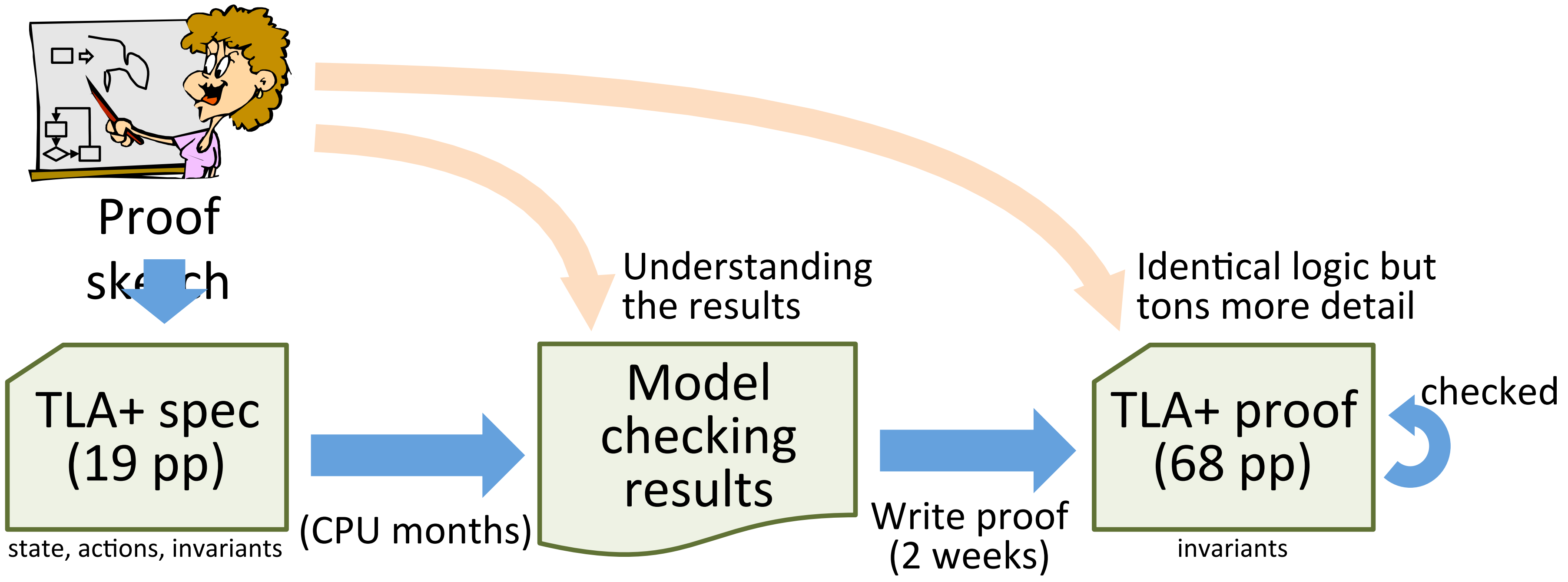
TLA+ spec (19 pp)

A few CPU minutes…

Bug1	violation example	
Bug2	violation example	
Bug3	violation example	
…	…	
Bug12	violation example	
Bug13	violation example	
Bug14	no error	violates liveness
Bug15	no error	violates append-only log
Bug16	no error	violates append-only log

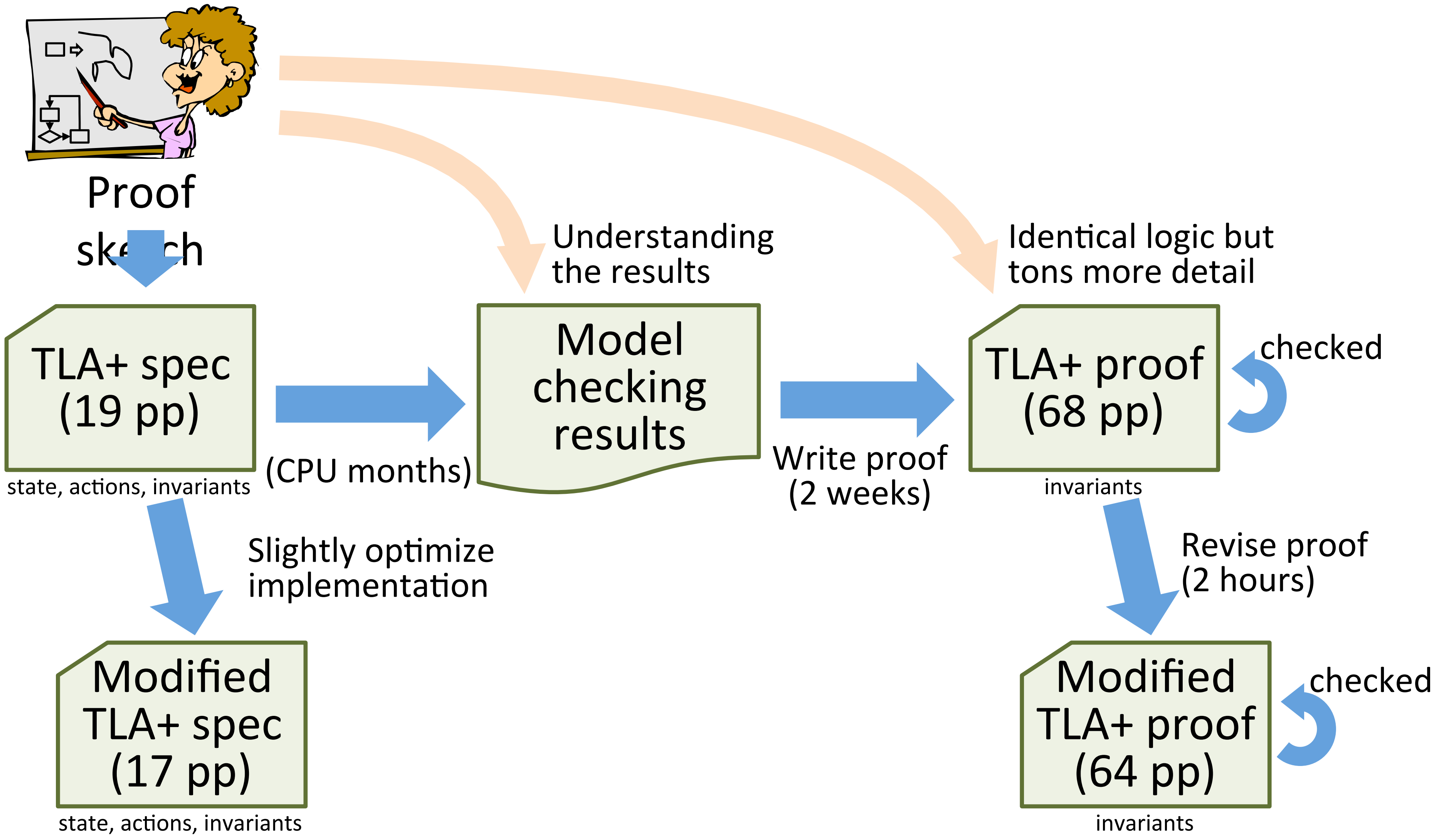
After analysis: these bugs happen not to violate safety

Now write the formal proof



Proof does not permit the append-only log violation bugs.

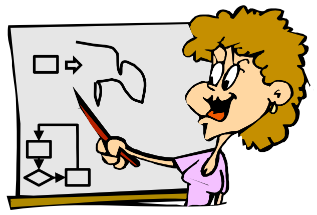
Include a slight optimization



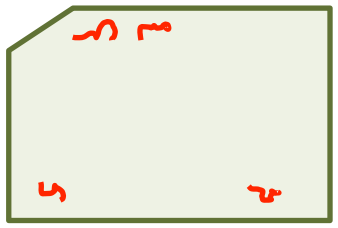
Proof sketch
Understanding the results
Identical logic but tons more detail
TLA+ spec (19 pp)
state, actions, invariants
(CPU months)
Model checking results
Write proof (2 weeks)
TLA+ proof (68 pp)
checked
invariants
Slightly optimize implementation
Revise proof (2 hours)
Modified TLA+ spec (17 pp)
state, actions, invariants
Modified TLA+ proof (64 pp)
checked
invariants

Conclusions

- Proof sketch was valuable
  - Helped understand model results
  - Guided formal proof



- Assurance via intentional bugs before proof
- Better to specify the actual invariant, not the (stronger) proof invariant



- Amazingly easy to create proof for slightly modified specification

invariants