Praxis
High Integrity
Systems

Document Set	Tokeneer ID Station		Reference S.P1229.40.4

Title	**:**	Security Properties		
Synopsis	**:**	This document presents a formal specification of the Security Properties of the TIS core and a demonstation of satisfaction of these security properties by the specified TIS core.		
Contents	**:**	See table of contents		
Status	**:**	Definitive		
Issue Number	**:**	1.1		
Date	**:**	14th August 2008		
Copied To	**:**	**NSA** Randolph Johnson **SPRE Inc.**	**Praxis High Integrity Systems** Project Team	
Front Sheet	**:**	Quality		
Originators	**:**	David Cooper	**Signed**	**:**
Approver	**:**	Janet Barnes	**Approved**	**:**

0 DOCUMENT CONTROL

Changes History

All issues of this document have been type-checked with ƒUZZ *and have given no errors.*

Issue 0.1 (10th July 2003) First draft issued for comments within Praxis.

Issue 1.0 (23rd July 2003) Updated following internal review. First issue to client for comment.

Issue 1.1 (14th August 2008) Updated for public release.

Changes Forecast

Updates following client review, if any.

References

1 The Z Notation: A Reference Manual, J.M Spivey, Prentice Hall, Second Edition, 1992
2 TIS Software Requirements Specification, Version 2.0, S.P1229.41.1.
3 TIS Kernel Protection Profile, SPRE Inc, Version 1.0, 5 February 2003.
4 TIS Security Target, S.P1229.40.1.
5 TIS Formal Specification, S.P1229.50.1.

Abbreviations

TIS	Token ID Station

1 TABLE OF CONTENTS

2 INTRODUCTION

In order to demonstrate that developing highly secure systems to the level of rigour required by the higher assurance levels of the Common Criteria is possible, the NSA has asked Praxis High Integrity Systems to undertake a research project to develop part of an existing secure system (the Tokeneer System) in accordance with their high-integrity development process. This development work will then be used to show the security community that is is possible to develop secure systems rigorously in a cost effective manner.

This document is the formal statement of the key security properties the Token ID Station (TIS) shall possess, written using the Z notation.

2.1 Structure of this Document

This document should be read in conjunction with the formal specification, [5].

Section 3.1 states the key security properties as theorems on the formal specification.

Section 3.2 gives mathematical arguments that the formal specification given in [5] does in fact possess each of the properties stated.

3 SECURITY PROPERTIES

We define a general operation, representing all the calculation and decisions, but excluding the Polling of the peripherals and the Updating of the peripherals. Many of the security properties refer to properties that hold during one execution of this general operation.

$$
\begin{aligned}
&\mathit{TISOp} \mathrel{\widehat=} (\mathit{TISEnrolOp}\\
&\quad \lor \mathit{TISUserEntryOp}\\
&\quad \lor \mathit{TISAdminLogon}\\
&\quad \lor \mathit{TISStartAdminOp}\\
&\quad \lor \mathit{TISAdminOp}\\
&\quad \lor \mathit{TISAdminLogout}\\
&\quad \lor \mathit{TISIdle})\\
&\qquad \land \mathit{LogChange}
\end{aligned}
$$

3.1 Security Properties

3.1.1 Property 1: Unlock with Token

If the latch is unlocked by the TIS, then the TIS must be in possession of either a User Token or an Admin Token. The User Token must either have a valid Authorisation Certificate, or must have valid ID, Privilege, and I&A Certificates, together with a template that allowed the TIS to successfully validate the user's fingerprint. Or, if the User Token does not meet this, the Admin Token must have a valid Authorisation Certificate, with role of "guard".

As the property refers to the real world (*latch*), we must look at the combined effect of carrying out a TIS calculation and then updating the world. Hence we define:

$$\mathit{TISOpThenUpdate} \mathrel{\widehat=} \mathit{TISOp} \mathbin{⨾} \mathit{TISUpdate}$$

▷ See: *TISOp* (p. 5)

We also make statements about validity in the absence of currency checks. We therefore define the following two schemas, which are direct copies of the schemas *UserTokenWithOKAuthCert* and *UserTokenOK*, but with the currency checks removed.

```
UserTokenWithOKAuthCertNoCurrencyCheck
  KeyStore
  UserToken
  currentTime : TIME
  ----------
  currentUserToken ∈ ran goodT
  ∃ TokenWithValidAuth •
      (goodT(θTokenWithValidAuth) = currentUserToken
      ∧ (∃ AuthCert • θAuthCert = the authCert ∧ AuthCertOK))
```

$$
\begin{array}{l}
\textit{UserTokenOKNoCurrencyCheck} \\
\hline
\textit{KeyStore} \\
\textit{UserToken} \\
\textit{currentTime} : \textit{TIME} \\
\hline
\textit{currentUserToken} \in \operatorname{ran} \textit{goodT} \\
\exists\, \textit{CurrentToken} \bullet \\
\quad (\textit{goodT}(\theta \textit{ValidToken}) = \textit{currentUserToken} \\
\quad \land (\exists\, \textit{IDCert} \bullet \theta \textit{IDCert} = \textit{idCert} \land \textit{CertOK}) \\
\quad \land (\exists\, \textit{PrivCert} \bullet \theta \textit{PrivCert} = \textit{privCert} \land \textit{CertOK}) \\
\quad \land (\exists\, \textit{IandACert} \bullet \theta \textit{IandACert} = \textit{iandACert} \land \textit{CertOK}))
\end{array}
$$

$$
\begin{array}{l}
\Delta \textit{IDStation}; \Delta \textit{RealWorld} \mid \\
\quad \textit{TISOpThenUpdate} \\
\quad \land \textit{latch} = \textit{locked} \land \textit{latch}' = \textit{unlocked} \\
\vdash \\
\quad (\exists\, \textit{ValidToken} \bullet \textit{goodT}(\theta \textit{ValidToken}) = \textit{currentUserToken} \\
\qquad \land \textit{UserTokenOKNoCurrencyCheck} \\
\qquad \land \textit{FingerOK}) \\
\quad \lor \\
\quad (\exists\, \textit{TokenWithValidAuth} \bullet \textit{goodT}(\theta \textit{TokenWithValidAuth}) = \textit{currentUserToken} \\
\qquad \land \textit{UserTokenWithOKAuthCertNoCurrencyCheck}) \\
\quad \lor \\
\quad (\exists\, \textit{ValidToken} \bullet \textit{goodT}(\theta \textit{ValidToken}) = \textit{currentAdminToken} \\
\qquad \land \textit{authCert} \neq \varnothing \land (\textit{the}\ \textit{authCert}).\textit{role} = \textit{guard})
\end{array}
$$

▷ See: *TISOpThenUpdate* (p. 5), *UserTokenOKNoCurrencyCheck* (p. 5), *UserTokenWithOKAuthCertNoCurrencyCheck* (p. 5)

3.1.2 Property 2: Unlock at allowed time

If the latch is unlocked automatically by the TIS, then the current time must be close to being within the allowed entry period defined for the User requesting access.

"close" needs to be defined, but is intended to allow a period of grace between checking that access is allowed and actually unlocking the latch. "Automatically" refers to the latch being unlocked by the system in response to a user token insertion, rather than being manually unlocked by the guard.

$$
\begin{array}{l}
\Delta \textit{IDStation}; \Delta \textit{RealWorld} \mid \\
\quad \textit{TISOpThenUpdate} \\
\quad \land \textit{latch} = \textit{locked} \land \textit{latch}' = \textit{unlocked} \\
\quad \land \textit{adminTokenPresence} = \textit{absent} \\
\vdash \\
\quad (\exists\, \textit{ValidToken} \bullet \textit{goodT}(\theta \textit{ValidToken}) = \textit{currentUserToken} \\
\qquad \land (\exists\, \textit{recentTime} : \textit{timesRecentTo}\ \textit{currentTime} \bullet \\
\qquad\quad \textit{recentTime} \in \textit{entryPeriod}\ \textit{privCert}.\textit{role}\ \textit{privCert}.\textit{clearance}.\textit{class}))
\end{array}
$$

▷ See: *TISOpThenUpdate* (p. 5)

where we define a function *timesRecentTo* that gives a set of times close to a given time. We define it here loosely.

$$
\begin{array}{l}
\textit{timesRecentTo} : \textit{TIME} \longrightarrow \mathbb{P}\, \textit{TIME} \\
\hline
\forall\, t : \textit{TIME} \bullet t \in \textit{timesRecentTo}\ t
\end{array}
$$

3.1.3 Property 3: Alarm when insecure

An alarm will be raised whenever the door/latch is insecure.

"insecure" is defined to mean the latch is locked, the door is open, and too much time has passed since the last explicit request to lock the latch.

There are two places in which real world updates occur:

$$\begin{aligned}&\mathit{Update} \mathrel{\widehat{=}} \\ &\quad \mathit{TISEarlyUpdate} \lor \mathit{TISUpdate}\end{aligned}$$

$$\begin{aligned}&\mathit{Update} \mid \\ &\quad \mathit{latch}' = \mathit{locked} \\ &\quad \land \mathit{currentDoor}' = \mathit{open} \\ &\quad \land \mathit{currentTime}' \geq \mathit{alarmTimeout} \\ &\vdash \\ &\quad \mathit{alarm}' = \mathit{alarming}\end{aligned}$$

▷ See: *Update* (p. 7)

3.1.4 Property 4: No loss of audit

No audit data is lost without an audit alarm being raised.

$$\begin{aligned}&\mathit{TISOp} \mid \mathit{auditAlarm} = \mathit{auditAlarm}' = \mathit{silent} \\ &\vdash \\ &\quad \mathit{auditLog} \cup (\mathbf{if}\ \mathit{currentFloppy} \in \operatorname{ran} \mathit{auditFile}\ \mathbf{then}\ \mathit{auditFile}^{\sim} \mathit{currentFloppy}\ \mathbf{else}\ \varnothing) \subseteq \\ &\qquad \mathit{auditLog}' \cup (\mathbf{if}\ \mathit{currentFloppy}' \in \operatorname{ran} \mathit{auditFile}\ \mathbf{then}\ \mathit{auditFile}^{\sim} \mathit{currentFloppy}'\ \mathbf{else}\ \varnothing)\end{aligned}$$

▷ See: *TISOp* (p. 5)

3.1.5 Property 5: Audit records are consistent

The presence of an audit record of one type (e.g. recording the unlocking of the latch) will always be preceded by certain other audit records (e.g. recording the successful checking of certificates, fingerprints, etc.)

Such a property would need to be defined in detail, explaining the data relationship rules exactly for each case. This has not been done.

3.1.6 Property 6: Configuration/floppy changed by admin

The configuration data will be changed, or information written to the floppy, only if there is an Admin person logged on to the TIS.

$$\begin{aligned}&\mathit{TISOp} \mid \mathit{adminTokenPresence} = \mathit{absent} \\ &\vdash \\ &\quad \Xi \mathit{Config} \land \Xi \mathit{Floppy}\end{aligned}$$

▷ See: *TISOp* (p. 5)

3.2 Arguments that Security Properties hold

We present informal arguments that each of the security properties is indeed a property of the system as specified.

Note that *TISOp* has signature

$$\begin{array}{l} \Delta \mathit{IDStation} \\ \Xi \mathit{TISControlledRealWorld} \\ \Delta \mathit{TISMonitoredRealWorld} \end{array}$$

and *TISUpdate* has signature

$$\begin{array}{l} \Xi \mathit{IDStation} \\ \Delta \mathit{TISControlledRealWorld} \\ \Delta \mathit{TISMonitoredRealWorld} \end{array}$$

3.2.1 Property 1: Unlock with Token

Restating the property:

$$\begin{array}{l}
\Delta \mathit{IDStation}; \Delta \mathit{RealWorld} \mid \\
\quad \mathit{TISOpThenUpdate} \\
\quad \land \mathit{latch} = \mathit{locked} \land \mathit{latch}' = \mathit{unlocked} \\
\quad \land (\mathit{latch} = \mathit{currentLatch}) \\
\vdash \\
\quad (\exists \mathit{ValidToken} \bullet \mathit{goodT}(\theta \mathit{ValidToken}) = \mathit{currentUserToken} \\
\qquad \land \mathit{UserTokenOKNoCurrencyCheck} \\
\qquad \land \mathit{FingerOK}) \\
\quad \lor \\
\quad (\exists \mathit{TokenWithValidAuth} \bullet \mathit{goodT}(\theta \mathit{TokenWithValidAuth}) = \mathit{currentUserToken} \\
\qquad \land \mathit{UserTokenWithOKAuthCertNoCurrencyCheck}) \\
\quad \lor \\
\quad (\exists \mathit{ValidToken} \bullet \mathit{goodT}(\theta \mathit{ValidToken}) = \mathit{currentAdminToken} \\
\qquad \land \mathit{authCert} \neq \emptyset \land (\mathit{the}\ \mathit{authCert}).\mathit{role} = \mathit{guard})
\end{array}$$

▷ See: *TISOpThenUpdate* (p. 5), *UserTokenOKNoCurrencyCheck* (p. 5), *UserTokenWithOKAuthCertNoCurrencyCheck* (p. 5)

The hypothesis considers only the combination of the general operation and the standard update step. This is justified because *latch* can only change in an *Update*, which is always either

$$\mathit{TISOp} \mathbin{⨟} \mathit{TISUpdate}$$

or

$$\mathit{Poll} \mathbin{⨟} \mathit{TISEarlyUpdate}$$

The second of these can only result in an increase in *currentTime*, which means that the only change to *latch* possible is to become *locked*. This is not the change we are investigating here.

It is therefore reasonable to limit our hypothesis to the case $\mathit{TISOp} \mathbin{⨟} \mathit{TISUpdate}$.

We have started by adding the predicate $\mathit{latch} = \mathit{currentLatch}$ to the hypothosis. This is justified because we are concerned only with the case when the latch is entirely under the control of the TIS, and has been to date. We can therefore assume that at the beginning of this operation the TIS's view of the position of the latch (*currentLatch*) agrees with the actual state of the latch (*latch*).

We will refer to the intermediate state by double-dash, the before-state by no dashes, and the after-state by a single dash.

Taking the hypothesis $latch' = unlocked$, we deduce that $currentLatch'' = unlocked$, from the behaviour of *TISUpdate*. $currentLatch = locked$ from the predicates in the hypothesis.

So we have a change of *currentLatch*. This state of *currentLatch* is entirely defined in the invariant to *DoorLatchAlarm* by *currentTime* and *latchTimeout*. A change can only happen in *TISOp*, due to $\Xi IDStation$ in *TISUpdate*.

There are three components of *TISOp* that do not have $\Xi DoorLatchAlarm$: *UnlockDoorOK*, *ShutdownOK*, and *OverrideDoorLockOK*.

We can ignore *ShutdownOK* because it involves a change to *locked*, not *unlocked*.

UnlockDoorOK

$status = waitingRemoveTokenSuccess$

The sequence of states that must have been passed through is therefore either:

quiescent to *gotUserToken* to *waitingFinger* to *gotFinger* to *waitingUpdateToken* to *waitingEntry* to *waitingRemoveTokenSuccess*

or

quiescent to *gotUserToken* to *waitingEntry* to *waitingRemoveTokenSuccess*

In the first, *ValidToken* and *UserTokenOK* are assured in passing to the *waitingFinger* state, and *FingerOK* is assured in passing to the *waitingUpdateToken* state. This proves the first branch.

In the second, *TokenWithValidAuth* and *UserTokenWithOKAuthCert* are both assured in passing to the *waitingEntry* state. This proves the first branch of the theorem.

OverrideDoorLockOK

$enclaveStatus = waitingStartAdminOp$

and

$the\ currentAdminOp = overrideLock$.

This means that the system must have passed through the following states:

$enclaveQuiescent, role = nil$ to *gotAdminToken* to $enclaveQuiescent, roll \neq nil$ to *waitingStartAdminOp*

In passing to the $enclaveQuiescent, role \neq nil$ state the *rolePresent* is set from the *role* in the *authCert*, and the Admin Token is checked for validity (and in particular, checked for being present).

In passing to the *waitingStartAdminOp* state the check on the validity of the requested operation ensures that $currentAdminOp \in keyedOp^{\sim} (\!| availableOps |\!)$, and in turn the value of *availableOps* is tied to the value of the *rolePresent* by a state invariant. The state invariant ensures that the operation to override the door lock (the operation we must be considering here) is only available to the *guard*. Therefore, as we know that the hypothesis implies that the operation being carried out is *OverrideDoorLockOK*, we know that $the\,currentAdminOp = overrideLock$, so we know that the *rolePresent* is a *guard*, and hence the role in the *authCert* is a *guard*, as required.

3.2.2 Property 2: Unlock at allowed time

Restating the property:

$$
\begin{array}{l}
\Delta IDStation; \Delta RealWorld \mid \\
\quad TISOpThenUpdate \\
\quad \land latch = locked \land latch' = unlocked \\
\quad \land adminTokenPresence = absent \\
\quad \land (latch = currentLatch) \\
\quad \land (\forall t : TIME \bullet \\
\qquad ((t - tokenRemovalDuration) \mathrel{..} (t + tokenRemovalDuration)) \cap TIME \subseteq timesRecentTo\, t) \\
\vdash \\
\quad (\exists ValidToken \bullet goodT(\theta ValidToken) = currentUserToken \\
\qquad \land (\exists recentTime : timesRecentTo\, currentTime \bullet \\
\qquad\quad recentTime \in entryPeriod\, privCert.role\, privCert.clearance.class))
\end{array}
$$

▷ See: *TISOpThenUpdate* (p. 5), *timesRecentTo* (p. 6)

As in the proof of Property 1, we consider the combination of one general operation and one update.

We add the assumption that all times within *tokenRemovalDuration* of a given time are considered to be "recent" to that time.

We can follow the same proof as for Property 1, and with the additional hypothesis of $adminTokenPresence = absent$ we can exclude the *OverrideDoorLockOK* operation, as intended.

Following the sequence of state changes, we find that in passing to the *waitingRemoveTokenSuccess* state we carry out the *EntryOK* operation, which checks that the *currentTime* is within the entry period defined for the class and role given in the Privilege Cert:

$currentTime \in entryPeriod\, privCert.role\, privCert.clearance.class.$

However, this relates to the *currentTime* given in passing *to* this state, whereas the security property we are proving relates to the *currentTime leaving* this state, to *quiescent*.

To deal with this, consider the state changes allowed subsequently. *WaitingTokenRemoval* allows the system to stay in this state, but for no longer than *tokenRemovalDuration*. After this time, the system is forced into *waitingRemoveTokenFail*, via *TokenRemovalTimeout*, and ceases to be of concern. The only other exit from this state is via *UnlockDoor*, which actually carries out the change to *currentLatch* we are investigating.

Therefore, we can show that the value of *currentTime* when the latch is unlocked cannot differ from the value of *currentTime* when the validity period is checked by more than *tokenRemovalDuration*.

The additional assumption on the proof is that our definition of "recently" is sufficiently broad to accommodate the time lag that our system actually works with.

3.2.3 Property 3: Alarm when insecure

Restating the property:

$$
\begin{array}{l}
\mathit{Update} \mid \\
\quad \mathit{latch}' = \mathit{locked} \\
\quad \land \mathit{currentDoor}' = \mathit{open} \\
\quad \land \mathit{currentTime}' \geq \mathit{alarmTimeout} \\
\vdash \\
\quad \mathit{alarm}' = \mathit{alarming}
\end{array}
$$

▷ See: *Update* (p. 7)

This follows quite directly from the state invariant in *DoorLatchAlarm*.

$alarm'$ is set to $currentAlarm''$ (the only way it is ever set) in both of the Update operations in the hypothesis, and the invariant in the included Ξ*DoorLatchAlarm* specifically ties the conclusion to the remaining predicates in the hypothesis.

3.2.4 Property 4: No loss of audit

Proof not done.

3.2.5 Property 5: Audit records are consistent

Proof not done.

3.2.6 Property 6: Configuration/floppy changed by admin

Proof not done.

3.3 Note on arguments

Many of these arguments depend upon a sequence of state transitions. The arguments presented argue that because the system is in a state with a certain *status*, then a specific series of states (different *status*es) must have been passed through. But because Z specifies the system in terms of atomic state transitions, this argument is not fully justified.

It can be made fully justified by augmenting the state invariants with the properties relied upon in each state. It is possible to prove that these properties are established as the system passes through its state transitions, and because they are carried in the state invariant, they are available to support the proofs.

We have not done this, as we believe it will add little to the assurance of correctess, and is very time consuming. At higher levels of the CC assurance we would be required to carry out more formal proofs, in which case these modifications would be done.