# IBM CRYPTOGRAPHIC SECURITY CHIP FOR PC CLIENTS, MANUFACTURED BY ATMEL (AT90SP0801)

# COMMON CRITERIA

# SECURITY TARGET

# VERSION 4.4

---

Jean E. Petty

September 1, 2001


---

Suite 100 West ¨ 7927 Jones Branch Drive ¨ McLean, VA 22102-3305 ¨ 703 848-0883 ¨ Fax 703 848-0960

## TABLE OF CONTENTS

**SECTION** **PAGE**

## TABLE OF FIGURES

**FIGURE** **PAGE**

## TABLE OF TABLES

**TABLE** **PAGE**

**Revision History**

| | |
|---|---|
| Version 1.2, 8/23/2000 | First submission of the ST to the CygnaCom SEL for evaluation. |
| Version 2.1, 11/14/2000 | Incorporated comments from the ST evaluation. |
| Version 2.3, 1/3/2001 | Incorporated changes based on receipt of additional Atmel documentation |
| Version 3, 2/14/2001 | Incorporated changes based on receipt of CygnaCom SEL ST Evaluation Technical Report Version 2.0.1. |
| Version 3.3, 4/4/2001 | Incorporated changes based on an updated product specification document and comments during TOE evaluation. |
| Version 3.4, 6/4/2001 | Incorporated preliminary changes based on EORs from the evaluation of the ST. |
| Version 4.0, 7/26/2001 | Name of product was changed to better represent TOE. Incorporated changes based on EORs from the evaluation of the ST and the product. |
| Version 4.1, 7/31/2001 | Updated Objectives O.SOF to state objective for SOF-basic, and updated sections 6.1.6 and 8.2.4, SOF. |
| Version 4.2, 8/10/2001 | Updated SOF analysis in section 6.1.6. |
| Version 4.3, 8/15/2000 | Updated SOF analysis in section 6.1.6 in response to EOR AVA_SOF 2. Updated Table 6.6 with new documentation used to meet assurance requirements and for assurance requirements rationale. |
| Version 4.4, 9/1/2001 | Updated ST in response to EORs ASE.3-1 and ASE.3-2. Removed “Draft” from footer. |

# 1 SECURITY TARGET INTRODUCTION

## *1.1 SECURITY TARGET IDENTIFICATION*

TOE Identification: IBM Cryptographic Security Chip for PC Clients manufactured by Atmel (AT90SP0801).

ST Identification: IBM Cryptographic Security Chip for PC Clients manufactured by Atmel (AT90SP0801) Common Criteria Security Target, Version 4.4.

Assurance level: EAL3, augmented.

Registration: <To be filled in upon registration>

Keywords: PC, Secure Client, Digital Signature, Smart Card, RSA.

## *1.2 SECURITY TARGET OVERVIEW*

The IBM Cryptographic Security Chip for PC Clients, manufactured by Atmel, provides RSA digital signature services to a standard PC workstation. The IBM Cryptographic Security Chip for PC Clients is a daughter card on the PC motherboard that allows operation in a PC environment, using standard client operating systems. Within the PC, the IBM Cryptographic Security Chip for PC Clients provides the following security services: RSA digital signature, decryption, and limited authentication.

This ST was developed by CygnaCom Solutions under contract with IBM. The ST revision history is provided in the front matter of this document.

## *1.3 COMMON CRITERIA CONFORMANCE*

The TOE is Part 2 conformant, Part 3 conformant, and meets the requirements of EAL 3 augmented with CC component ADV_SPM.1, with the Common Criteria Version 2.1.

# 2 TOE DESCRIPTION

## *2.1 PRODUCT TYPE*

The TOE consists of a secure signature generation chip, Atmel AT90SP0801, which is a daughter card on the system motherboard of an IBM Personal Computer (PC). The TOE performs RSA digital signature, data decryption, and limited authentication. It is packaged in a 20 Lead SOIC package or a 28 pin TSSOP package and includes EEPROM memory that can be loaded with RSA public/private key pairs. Communications to the main processor is via SMbus (12C). There are a finite number of commands accepted by the chip. The TOE is used in a number of standard, off-the-shelf IBM PC Client configurations. The TOE configuration, hardware, and firmware are physically the same and function the same, regardless of PC configuration.

## *2.2 GENERAL TOE FUNCTIONALITY*

The TOE is designed to compute public key message signatures, to decrypt small amounts of data, and to perform limited authentication in the form of password protection of chip functions. The chip operates with the following logical components, which are depicted in Figure 2-1:

- SMBus for communication with the PC: All communication to and from the PC processor is through the SMBus Communications Port. All communication through the SMBus is transferred to or from the I/O buffer on the chip; no data may be passed directly to or from the SMBus and the registers or other memory locations on the chip.
- I/O Buffer: The I/O buffer is used to transfer data to or from the SMBus interface. It is cleared on power-up. Filling the buffer from one source or another automatically invalidates the previous contents. All crypto functions transfer information to or from the buffer to perform calculations. This information is not stored in any nonvolatile memory and is lost on power cycles. All bits are sent to or read from the chip most significant bit first.
- Registers: The registers located in the chip are visible to the chip software. Table 2.1 below provides a summary of the registers. Only some registers are available for read or write, as shown in the table. Registers may not be read or written directly. Instead, the load and store commands are used to transfer information between the register and the I/O buffer, which is accessed by the system using read and write commands. The chip requires that the amount of data read from or written to the buffer be identical to the size of the register to which the transaction corresponds.
- Commands: Table 2.2 provides a summary of the commands. Data transfer commands to and from the chip (read and write) follow the SMBus V1.1 standard, using only some of the command protocols. These are described in Table 2.2 below. Other commands perform various internal operations of the chip, using data already stored in either the I/O buffer or the registers. These other commands provide the functionality of the chip, such as PKCS sign and decryption, and are summarized in Table 2.2 below.
- Crypto Engine: The crypto engine performs RSA signing and decryption, including computation and storage in volatile memory of intermediate results. The crypto engine and intermediate data is not accessible by the user. All results are stored to a register or the I/O buffer (depending upon the command) and are accessible to the user only through the command set.

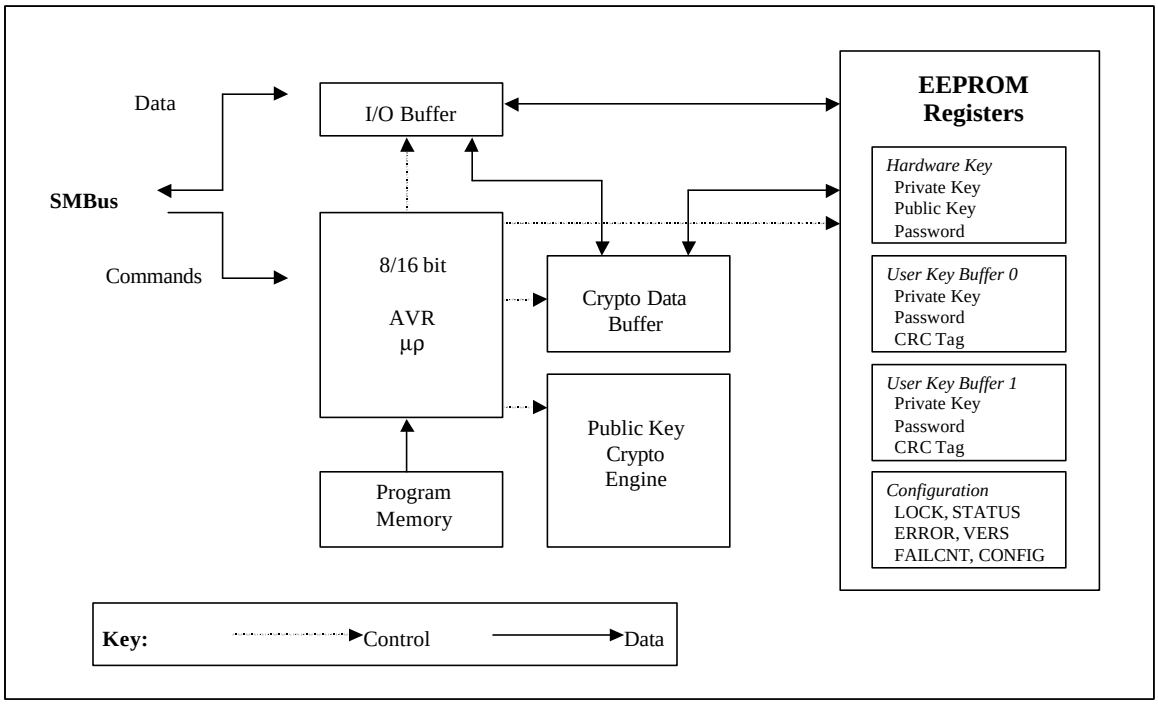


**Figure 2-1. Logical Diagram of Chip Operation**

The chip is included as a daughter card on the system motherboard of certain models of IBM PCs. Chip initialization is accomplished by loading the chip hardware private/public key pair and a hardware password. Initialization also involves system administration steps that ensure proper configuration of the system. The required system configuration is defined in Table 2.3 below and is provided in the System Administration Guide. The initialization step is required before any chip functionality is accessible to the user.

The chip performs RSA signature and decryption. There are two types of key pairs: the hardware key pair and user specified key pairs. The hardware key pair is stored on the chip at chip initialization. The hardware key pair has an associated password, which is also specified at initialization. The hardware password is used to gain access to signing and decryption operations using the hardware key pair. The hardware password is also used to access administrative functions that configure the chip. The hardware private/public key pair and a hardware password are required to be loaded on the chip for proper operation.

User specified keys may be loaded by the user or application at any time during normal system operation. Up to two user keys can be loaded in the chip at once. Additional user keys loaded to the chip overwrite the content of the two user key registers. The keys may or may not be password protected, at the discretion of the user. The user keys may be used for RSA signature and decryption.

There are three types of passwords: the hardware password, which is directly associated with access to the hardware key pair and other functionality on the chip; the Failure Counter Reset password, which is a password set at initialization that allows the reset of the failed password counter; and user passwords, which may or may not be specified to protect access to user keys. Password protection is specified at system initialization for the hardware password. It is important to note that the hardware password protects the hardware key pair and the configuration of the chip. The hardware key pair, the hardware password, and the system configuration data are considered TSF data. In the required system configuration, the hardware password is required and

must be set according to appropriate password guidelines at system initialization. The Failure Counter Reset password must be set at system initialization according to appropriate password guidelines as defined in the System Administration Guide. The user passwords are discretionary, depending upon the application and utilization of the chip by the user. The user passwords control access to cryptographic operations on the chip using user keys. User key pairs are considered to be user data and user passwords are considered to be TSF data.

**Table 2.1 – Summary of the Registers**

| Name | Mode | Description |
|---|---|---|
| VERS_R | R | Chip revision. |
| HWPRIV_R | P | Hardware Private Key. This register stores the Hardware Private Key, which can be written to the register only with correct entry of the hardware password and only if the LOCK_R register is 0. |
| HWPUB_R | R | Hardware Public Key. |
| HWPWD_R | P7C | Hardware password. |
| LOCK_R | P7 | Hardware key lock. This register, when the LSB is set as non-zero, permanently prevents writing of HWPRIV_R and LOCK_R |
| FAILRSTPWD_R | P7C | Failure counter reset password. This register contains a password that is checked in order to clear the FAILCNT_R register. |
| FAILCNT_R | R | Failed password attempt counter. This register keeps track of the total number of failed password check attempts on any of the passwords within the chip. As the total number of cumulative failed password checks grows, at certain intervals the chip locks up and prevents all operations for a specified period of time. This register can be cleared (set back to 0) with either the pw_check[HW] or the pw_check[FailReset] commands. |
| MAXBLK_R | P7 | Max size of read block, default 32 (decimal). This register defines the maximum number of bytes to be returned on the read command. |
| CONFIG_R | P7 | Chip configuration. This register contains bits that are used to enable or disable certain operation of the chip. The contents of this register is stored in the EEPROM memory of the chip, is retained across power cycles, and is unaffected by the chip clear operation. On a write, bits 3-7 are ignored; on a read their value is unspecified. Values:<br>▪ Bit 0, if “1” enables the chip clear operation to be performed, assuming that STATUS_R/bit 7 is “1” and LOCK_R/bit 0 is a “0”.<br>▪ Bit 1, if “1” enables the hardware and FailReset passwords to be written regardless of the state of STATUS_R/bit 7.<br>▪ Bit 2 returns 1 on a read, is ignored on a write.<br>▪ Bits 3 - 7 are ignored. |
| SOURCE_R | RW | Key decode source. The decode_key command uses this register as the decryption key pointer. A value of 0xFF in SOURCE_R signifies the hardware private key. |

**Table 2.1 – Concluded**

| Name | Mode | Description |
|---|---|---|
| STATUS_R | RW | Chip operating status. This register combines status information within the chip into a single byte. Bits 0, 1, 2, and 5 convey information that is stored in the EEPROM memory of the chip and their state is retained across power cycles. The remaining bits convey either volatile state information or other information that is stored in SRAM and are not retained across power cycles.<br>▪ Bit 0, On a read, if "1" indicates that the hardware private key is valid.<br>▪ Bit 1, On a read, if "1" indicates that the HW private key is locked.<br>▪ Bit 2, On a read, if "1" indicates that the chip is in a lockout period as a result of the number of invalid passwords entered.<br>▪ Bit 3, On a write, if "1" then all password check flags will be cleared.<br>▪ Bit 4, Chip clear. If STATUS_R/bit 7 is a "1" and CONFIG_R/bit 0 is a "0", then writing a "1" to this bit will cause all internal keys to be invalidated and the hardware and failure passwords to be reset to "0".<br>▪ Bit 5, If "1" indicates that the chip clear operation has been executed but that either the hardware password or the failure reset password have not been subsequently written by the user. After both passwords have been written this bit is cleared.<br>▪ Bit 6, On a read, if "1" indicates that the chip is enabled and all functions are available. If "0" the chip is disabled and only loads and stores of STATUS_R or loads and reads of FAILCNT_R, ERROR_R and VERS_R are permitted.<br>▪ Bit 7, On power-up or after the reset pin is asserted, this bit is set to a "1"; when this bit is a "1" it can be written by the system. When it is a "0", writes to this bit are ignored and writes to some registers are prohibited. |
| ERROR_R | R | Error code from command. This register is used to convey to the system the cause of a command failure. After every command other than the read or write commands used to transfer data to and from the chip, the error register will be set. |
| CRC0_R | R | User Key Buffer # 0 CRC Label. The chip writes these registers during the user key decoding process to provide a reasonably unique identifier tag for keys stored on the chip. |
| CRC1_R | R | User Key Buffer # 1 CRC Label. Same as CRC0_R, above. |
| Constant_0 | R | Returns a constant 0 |

Key for Mode: Types of accesses that are permitted to this register from the I/O buffer:

- R: Read-only. May be read regardless of configuration or passwords.
- RW: May be read and written without passwords.
- P: May be written after the hardware password has been checked. Never readable.
- P7: May be written after the hardware password has been checked and if bit 7 of the STATUS_R register is 1. Always readable.
- P7C: May be written after the hardware password has been checked and if either bit 7 of the STATUS_R register or bit 1 of the CONFIG register is 1. Never readable.

Note that an 0x prefix denotes hexadecimal.

The required system configuration shows the content of the chip registers after the chip has been initialized and is ready for operation. Table 2.3 also shows the default value of the chip registers when the user receives the system. System initialisation may be accomplished a number of different ways, depending upon the application and user of the PC. System initialisation is not done automatically and must be performed correctly for the chip to be in a secure state.

**Table 2.2 – Summary of the Commands**

| Command | Description |
|---|---|
| Store (buffer register) | This command transfers the contents of the I/O buffer to the specified register. The exact number of bytes required must have been previously written into the I/O buffer or the command will fail. Note that only some registers may be written, as summarized in Table 2.1 above. |
| Load (buffer register) | This command transfers the contents of the specified register to the I/O buffer. Note that only some registers may be read, as summarized in Table 2.1 above. |
| Decode_key[b,k] | This command decodes the appropriate piece(s) of the user private key structure using a private key stored on the chip. Either the hardware key or a user specified key can be used to perform this decryption. The source key is specified in the SOURCE_R register. |
| Pw_check[HW] | The Pw_check[HW] command compares the 64 bit value stored in the I/O buffer with the hardware password stored in the HWPWD_R register. If they match, a CMOS HW password check flag is set and additional operations may be performed. |
| Pw_check[FailReset] | The Pw_check[FailReset] command compares the 64 bit value stored in the I/O buffer with the failed reset counter password in FAILRSTPWD_R. If they match, the failed password counter in FAILCNT_R is reset. |
| PKCS_sign[HW] | This command uses the hardware private key to sign a block of data. The data to be signed must be sent to the I/O buffer by the system before this command is executed. This command may only be executed after the hardware password has been entered. Note that the external system is required to perform the SHA-1 hash and send the digest to the chip; this is not performed by the chip. |
| PKCS_sign[k] | This command uses a previously loaded user key to sign a block of data. The data to be signed must be sent to the I/O buffer by the system before this command is executed. Depending on the value of the key mode for the selected user key, the password may have to be entered before this command can be executed. Note that the external system is required to perform the SHA-1 hash and send the digest to the chip; this is not performed by the chip. |
| Pw_check[k] | This command compares the 64 bit value stored in the I/O buffer with the password written into the appropriate user key buffer. If they match, the corresponding user password check flag will be set. This flag is reset on power up, on system reset and when a new key is loaded into the buffer. This flag can also be reset using STATUS_R/bit 3. |
| Decrypt_data[k] | This command decrypts small quantities of user information that has been previously encrypted with user keys. |
| Write | The write command uses the Block Write protocol of the SMBus V1.1 standard. Note that in this chip the count value can exceed 32. The chip does not support the “Write Byte” and “Write Word” protocols of the SMBus spec. |
| Read | The read command uses the Block Read protocol of the SMBus V1.1 standard. Note that in this chip the read command can be optionally executed without the preceding partial block write command. This chip does not support the Receive Byte, Read Byte, and Read Word protocols of the SMBus spec. |

Square brackets indicate commands that have parameters in the second byte where:

- b represents the parameter block number for key decoding. These parameter blocks contain the information use to decode the user private keys.
- k represents the user key buffer number that is to be used by the command.
- HW indicates the hardware key or the hardware password, depending upon the command.
- FailReset indicates the Failed reset counter password.

**Table 2.3 – Required System Configuration**

| Register Name | Default Value | Value after Initialization | Notes |
|---|---|---|---|
| VERS_R | N/A | N/A | Set at factory. Not user configurable |
| HWPRIV_R | invalid | Private key generated off chip. 2560 bits which must be specified in correct format and generated by the administrator in a manner that complies with security policy | Hardware Private Key can be written to the register only with correct entry of the hardware password and only if the LOCK_R register is 0. Attempts to read this register when the key is invalid will result in an error. |
| HWPUB_R | invalid | N/A, Value not stored; Generated from Private Key components | Hardware Public Key. Attempts to read this register when the key is invalid will result in an error. |
| HWPWD_R | 0 | Administrator specified password that complies with password policy | Hardware password. |
| LOCK_R | LSB=0 | LSB=0 | Hardware key lock. This register, when the LSB is set as non-zero, permanently prevents writing of HWPRIV_R and LOCK_R. All other bits are ignored. |
| FAILRSTPWD_R | 0 | Administrator specified password that complies with password policy | Failure counter reset password. This register contains a password that is checked in order to clear the FAILCNT_R register. |
| FAILCNT_R | 0 | 0 | This register can be cleared (set back to 0) with either the Pw_check[HW] or the Pw_check[FailReset] commands. |
| MAXBLK_R | 0x20 | 0x20 | Max size of read block, default 32. |
| CONFIG_R | Bit 0 = 1<br>Bit 1 = 1<br>Bit 2 = 1 | Bit 0 = 0<br>Bit 1 = 1<br>Bit 2 = 1 | Bits 3-7 are ignored. Values:<br>▪ Bit 0, if “1” enables the chip clear operation to be performed, assuming that STATUS_R/bit 7 is “1” and LOCK_R/bit 0 is a “0”.<br>▪ Bit 1, if “1” enables the hardware and FailReset passwords to be written regardless of the state of STATUS_R/bit 7<br>▪ Bit 2 is not used, but is set to “1”. |
| SOURCE_R | OxFF | OxFF | Key decode source. The Decode_key command uses this register as the decryption key pointer. A value of 0xFF in SOURCE_R signifies the hardware private key. |

Note that an 0x prefix denotes hexadecimal.

**Table 2.3 – Concluded**

| Register Name | Default Value | Value after Initialization | Notes |
|---|---|---|---|
| STATUS_R | 10100000 (0xA0) | 01000001 (0x41) | Chip operating status.<br>▪ Bit 0, On a read, if "1" indicates that the hardware private key is valid.<br>▪ Bit 1, On a read, if "1" indicates that the HW private key is locked.<br>▪ Bit 2, On a read, if "1" indicates that the chip is in a lockout period as a result of the number of invalid passwords entered.<br>▪ Bit 3, On a write, if "1" then all password check flags will be cleared.<br>▪ Bit 4, Chip clear. If STATUS_R/bit 7 is a "1" and CONFIG_R/bit 0 is a "0", then writing a "1" to this bit will cause all internal keys to be invalidated and the hardware and failure passwords to be reset to "0".<br>▪ Bit 5, If "1" indicates that the chip clear operation has been executed but that either the hardware password or the failure reset password have not been subsequently written by the user. After both passwords have been written this bit is cleared.<br>▪ Bit 6, On a read, if "1" indicates that the chip is enabled and all functions are available. If "0" the chip is disabled and only loads and stores of STATUS_R or loads and reads of FAILCNT_R, ERROR_R and VERS_R are permitted.<br>▪ Bit 7, On power-up or after the reset pin is asserted, this bit is set to a "1"; when this bit is a "1" it can be written by the system. When it is a "0", writes to this bit are ignored and writes to some registers are prohibited. |
| ERROR_R | 0 | N/A | Not under control of user |
| CRC0_R | 0 | 0 | User Key Buffer # 0 CRC Label. The chip writes these registers during the user key decoding process to provide a reasonably unique identifier tag for keys stored on the chip. |
| CRC1_R | 0 | 0 | User Key Buffer # 1 CRC Label. Same as CRC0_R. |
| Constant_0 | 0 | 0 | Returns a constant 0 |

Note that an 0x prefix denotes hexadecimal.

## *2.3 TOE BOUNDARY*

The TOE includes only the IBM Cryptographic Security Chip for PC Clients manufactured by Atmel (AT90SP0801). All communication to the TOE is through the SMbus on the TOE. The TOE performs RSA digital signature, data decryption, and limited authentication. Related security services such as user identification and authentication, RSA key generation, secure hash, and software random number generation are available through APIs and other software running on the PC. These services and APIs are not part of the TOE.

## *2.4 TOE ENVIRONMENT*

The TOE is included as a daughter card on the system motherboard of a number of IBM PC models. Operation of the TOE is only possible after initialization of the TOE at the user site. Initialization is not performed at the factory. The user is can determine whether the TOE is initialized by checking the STATUS_R register. If STATUS_R/Bit 5 is set to “1” then the chip is not initialized. The TOE is typically used by application(s) on the PC. Note that data to initialize the TOE, such as the hardware public/private key pair, are created outside the TOE, but are required for TOE operation.

# 3 SECURITY ENVIRONMENT

This section identifies the following:

- Secure usage assumptions,
- Organizational security policies, and
- Threats to Security
- Assumptions for the IT environment

## *3.1 SECURE USAGE ASSUMPTIONS*

Table 3.1 lists the Secure Usage Assumptions.

**Table 3.1 – Secure Usage Assumptions**

| | Assumption Name | Assumption Description |
|---|---|---|
| 1 | A.CONFIGURATION | The IBM Secure Signature Generation Chip will be properly installed and configured according to the System Administrator's Guide. |
| 2 | A.THREAT_LEVEL | The threat level for the TOE authentication function is assumed to be SOF-basic. |

## *3.2 SECURITY POLICIES*

Table 3.2 lists the security policies.

**Table 3.2 – Organizational Security Policies**

| | Policy Name | Policy Description |
|---|---|---|
| 1 | P.TSP | A TOE security policy (TSP) must identify all roles, services, and security-relevant data items, and specify what access (if any) a user, performing a service within the context of a given role, has to each of the security-relevant data items. |

## *3.3 THREATS TO SECURITY*

Table 3.3 lists the threats to security.

**Table 3.3 – Threats to Security**

| | Threat Name | Threat Description |
|---|---|---|
| 1 | T.ACCESS | An authorized user may be able to access or modify sensitive data for which that user is not authorized to access. |
| 2 | T.ACCESS_UNAUTH | An unauthorized person may be able to gain access to the system and to data. |
| 3 | T.BYPASS | An attacker may be able to bypass the TOE security functionality and gain unauthorized access to keys, data, and operations. |
| 4 | T.CRYPTO_OP | Cryptographic algorithms and/or cryptographic key sizes may be insufficient, allowing them to be deciphered by users that are not authorized access to the encrypted data. |
| 5 | T.CRYPTO_MGT | Incorrect cryptographic key destruction may cause an inadvertent disclosure of sensitive information. |
| 6 | T.DIG_SIG | Incorrect use of the digital signature and decryption capabilities of the chip could cause a loss of sensitive data, e.g., if a user application improperly supplied a public key to the chip to use in signing when a private key was required or vice versa. |
| 7 | T.IMPLEMENT | The TOE may not adequately protect cryptographic keys and data stored in the TOE, causing loss of confidentiality and integrity. |
| 8 | T.TAMPER | An attacker may be able to tamper with TSF data or programs. |

## *3.4 ASSUMPTIONS FOR THE IT ENVIRONMENT*

Table 3.4 lists the Secure Usage Assumptions for the IT environment.

**Table 3.4 –Assumptions for the IT Environment**

| | Assumption Name | Assumption Description |
|---|---|---|
| 1 | AE.CONFORMANCE | The use of the TOE does not guarantee the security of the overall system. The responsible authority in each user organization shall assure that the organization's computer or telecommunication systems provide an acceptable level of security for the given application and environment. |
| 2 | AE.OFF_CHIP | The following functions are preformed off-chip and must be performed securely and correctly: hardware key pair generation; user key pair generation; hashing; encryption of user key pairs, passwords, and key_mode byte as applicable for the chip Decode function; encryption of data for the chip Decrypt function. |
| 3 | AE.PHYSICAL_PROTECTION | The TOE provides no protection against physical threats such as simple power analysis, differential power analysis, external signals, extreme temperature, or physical tampering. Physical protection is assumed to be provided by the environment. |

# 4 SECURITY OBJECTIVES

## *4.1 SECURITY OBJECTIVES FOR THE TOE*

Table 4.1 lists the security objectives for the TOE.

**Table 4.1 – Security Objectives for the TOE**

| | Objective Name | Objective Description |
|---|---|---|
| 1 | O.AUTHENTICATION | The TOE must be able to provide the capability of associating a password or PIN known to a user to a key pair or set of system capabilities to support access control. |
| 2 | O.ADMIN | The TOE must provide functionality that enables an authorised administrator to configure the system in accordance with a specified TOE security policy. |
| 3 | O.CRYPTO_KEY | The TOE must perform cryptographic key destruction in accordance with PKCS-1 |
| 4 | O.CRYPTO_OP | The TOE must perform cryptographic operations, including RSA digital signature, RSA decryption, and public key generation in accordance with specified algorithms and cryptographic keys of a specified size, in accordance with PKCS-1 and of sufficient size to protect private/public key pairs from deciphering. |
| 5 | O.CONF_INTEGRITY | The TOE must ensure the confidentiality and integrity of key pairs and sensitive data stored in the TSF. |
| 6 | O.DIG_SIG | Sufficient guidance must be provided to application developers to allow correct and secure usage of the digital signature and decryption capabilities of the chip. |
| 7 | O.ENFORCEPOLICY | The TOE must enforce a clearly defined and documented informal TOE security policy (TSP) model. |
| 8 | O.I&A | The TOE must authenticate a password or PIN entered by a user before granting access to cryptography-related IT assets and operations within the TOE. |
| 9 | O.NONBYPASS | The TOE shall ensure that the TOE security functionality cannot be bypassed. |
| 10 | O.SELF_PROTECT | The TSF will maintain a domain for its own execution that protects it and its resources from external interference, tampering, or unauthorized disclosure. |
| 11 | O.SOF | The strength of function for the authentication mechanism must be SOF basic. |

## *4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT*

Table 4.2 lists security objectives for the environment.

**Table 4.2 – Security Objectives for the Environment**

| | **Objective Name** | **Objective Description** |
|---|---|---|
| 1 | OE.CONFIGURATION | The TOE must be installed and configured properly. |
| 2 | OE.CONFORMANCE | The responsible authority in each user organization must provide an acceptable level of security, including physical security such as simple power analysis, differential power analysis, external signals, extreme temperature, or physical tampering, for the given application and environment. |
| 3 | OE.OFF_CHIP | The TOE must be provided with correct, secure data for hardware key pair generation; user key pair generation; hashing; encryption of user key pairs, passwords, and key_mode byte as applicable for the chip Decode function; encryption of data for the chip Decrypt function |

# 5 IT SECURITY REQUIREMENTS

## *5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS*

This section contains the security functional requirements for the TOE. All of the functional requirements have been taken from Part 2 of the Common Criteria. The functional components are listed in Table 5.1.

**Table 5.1 – Functional Components**

| No. | Component | Component Name |
|---|---|---|
| **Class FCS: Cryptographic support** | | |
| 1 | FCS_CKM.1 | Cryptographic key generation |
| 2 | FCS_CKM.4 | Cryptographic key destruction |
| 3 | FCS_COP.1 | Cryptographic operation |
| **Class FDP: User Data Protection** | | |
| 4 | FDP_ACC.1 | Subset access control |
| 5 | FDP_ACF.1 | Security attribute based access control |
| 6 | FDP_ITC.2 | Import of user data with security attributes |
| **Class FIA: Identification and Authentication** | | |
| 7 | FIA_AFL.1 | Authentication failure handling |
| 8 | FIA_UAU.1 | Timing of authentication |
| 9 | FIA_UID.1 | Timing of identification |
| **Class FMT: Security Management** | | |
| 10 | FMT_MOF.1 | Management of security functions behavior |
| 11 | FMT_MSA.1 | Management of security attributes |
| 12 | FMT_MSA.2 | Secure security attributes |
| 13 | FMT_MSA.3 | Static attribute initialisation |
| 14 | FMT_MTD.1 | Management of TSF Data |
| 15 | FMT_SMR.1 | Security roles |
| **Class FPT: Protection of the TOE Security Functions** | | |
| 16 | FPT_RVM.1 | Non-bypassability of the TSP |
| 17 | FPT_SEP.1 | TSF domain separation |
| 18 | FPT_TDC.1 | Inter-TSF basic TSF data consistency |

The following sections contain the functional components from the Common Criteria (CC) Part 2 with the operations completed. The standard CC text is in regular font; the text inserted by the Security Target (ST) author is in italic font enclosed in brackets.

### 5.1.1 FCS – Cryptographic Support

#### FCS_CKM.1 Cryptographic key generation

Hierarchical to: No other components.

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*RSA public key component from private*

*key components*] and specified cryptographic key sizes: [*1024 bits*] that meet the following: [*PKCS-1*].

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

Application Note: The key pair is generated by the environment and not on the chip.

**FCS_CKM.4 Cryptographic key destruction**

Hierarchical to: No other components.

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*zeorization*] that meets the following: [*FIPS 140-1, Section 4.8.5, Key Destruction*].

Dependencies: FDP_ITC.1 Import of user data without security attributes; FMT_MSA.2 Secure security attributes

**FCS_COP.1 Cryptographic operation**

Hierarchical to: No other components.

FCS_COP.1.1;1 The TSF shall perform [*digital signature generation*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*512-1024 bits*] that meet the following: [*PKCS-1*].

FCS_COP.1.1;2 The TSF shall perform [*key decryption*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*512-1024 bits*] that meet the following: [*PKCS-1*].

FCS_COP.1.1;3 The TSF shall perform [*data decryption*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*512 - 1024 bits]* that meet the following: [*PKCS-1*].

Dependencies: FDP_ITC.1 Import of user data without security attributes; FCS_CKM.4 Cryptographic key destruction; FMT_MSA.2 Secure security attributes

### 5.1.2 FDP – User Data Protection

**FDP_ACC.1 Subset access control**

Hierarchical to: No other components.

FDP_ACC.1.1 The TSF shall enforce the [*cryptographic operation access controls*] on [

*a) Subjects: commands executing on behalf of users.*

*b) Objects: private keys and data sent to the TOE to be signed or decrypted.*

*c) Operations: use of the private key for sign or decrypt];]*

Dependencies: FDP_ACF.1 Security attribute based access control

**FDP_ACF.1 Security attribute based access control**

Hierarchical to: No other components.

| | |
|---|---|
| FDP_ACF.1.1 | The TSF shall enforce the [*cryptographic operation access controls*] to objects based on [*user password and key_mode byte*]. |
| FDP_ACF.1.2 | The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [*:*<br><br>*1) The user must supply a valid password.*<br><br>*2) a valid key_mode byte for the requested operation.* |
| FDP_ACF.1.3 | The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [*None*]. |
| FDP_ACF.1.4 | The TSF shall explicitly deny access of subjects to objects based on the [N*one.*] |
| Dependencies: | FDP_ACC.1 Subset access control; FMT_MSA.3 Static attribute initialisation |

**FDP_ITC.2 Import of user data with security attributes**

Hierarchical to: No other components.

| | |
|---|---|
| FDP_ITC.2.1 | The TSF shall enforce the [*cryptographic operation access controls*] when importing user data, controlled under the SFP, from outside of the TSC. |
| FDP_ITC.2.2 | The TSF shall use security attributes associated with the imported user data. |
| FDP_ITC.2.3 | The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received. |
| FDP_ITC.2.4 | The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data. |
| FDP_ITC.2.5 | The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC:[*no additional importation control rules*]. |
| Dependencies: | FDP_ACC.1 Subset access control; FMT_MSA.3 Static attribute initialisation, FPT_TDC.1 Inter-TSF basic TSF data consistency, FTP_TRP.1 Trusted path. |

### 5.1.3 FIA – Identification and Authentication

**FIA_AFL.1 Authentication failure handling**

Hierarchical to: No other components

| | |
|---|---|
| FIA_AFL.1.1 | The TSF shall detect when [*10*] unsuccessful authentication attempts occur related to [*the hardware password and the FAILRSTPWD password supplied when requesting any operation requiring those passwords. In addition, the chip will track cumulative unsuccessful password attempts for all operations.* ]. |

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall [*lockout the user for a specific period of time, as specified by the FAILCNT_R register (see table below). In the case of 10 unsuccessful authentication attempts on the hardware password or the FAILRSTPWD password, the FAILCNT_R register will be checked. If it is less than 224, it will be incremented to 224. If it is greater than 224, it will be incremented to the next multiple of 32. The lockout period will then begin. For cumulative unsuccessful password attempts for all passwords, including the hardware, FAILRSTPWD, and user passwords, the FAILCNT_R register is incremented by 1 for each unsuccessful attempt. Lockout periods apply when FAILCNT_R reaches each multiple of 32,*

| *Cumulative Failure* | *Lockout Period* |
|---|---|
| *32* | *1.2 Minutes* |
| *64* | *2.4 Minutes* |
| *96* | *4.8 Minutes* |
| *…* | *…* |
| *224* | *1 Hour, 17 Minutes* |
| *256* | *2 Hours, 34 Minutes* |
| *…* | *…* |
| *384* | *1.7 Days* |
| *…* | *…* |
| *512+* | *27.2 Days* |

].

Dependencies: FIA_UAU.1 Timing of authentication

**FIA_UAU.1 Timing of Authentication**

Hierarchical to: No other components

FIA_UAU.1.1 The TSF shall allow [read, write, store, load, sign, decrypt, and decode operations that do not require a password] on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

*Application Note:* *Administrator authentication is accomplished by the Administrator supplying the hardware password in the Pw_check[HW] command that matches the password stored in the HWPWD_R register. User authentication is accomplished by the User supplying the appropriate password that matches the password stored for the specified key register identifier in the each command. Note that the user key register identifier specified may point to a key with no password specified and therefore the key may be used by the world.*

Dependencies: FIA_UID.1 Timing of identification

**FIA_UID.1 Timing of identification**

Hierarchical to: No other components

FIA_UID.1.1 The TSF shall allow [read, write, store, load, sign, decrypt, and decode operations that are available to the world] *o*n behalf of the user to be performed before the user is identified.

FIA_UID.1.2;1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

*Application Note:* *Administrator identification is performed implicitly by the command that is executed, i.e., Pw_check[HW]. User identification is performed by the user supplying the key register identifier in the command that is executed. User creation/initialization occurs when another user or the administrator issues the decode command on behalf of the new user. The decode command causes the chip to decrypt (using the hardware private key or another private key loaded on the chip) and load a user key pair, password, and access control field to a specified user key register. The access control field determines if the owner or the world can perform operations using the key. The owner is the new user, who may then issue commands that specify the appropriate key identifier*.

No dependencies.

### 5.1.4 FMT – Security Management

**FMT_MOF.1 Management of security functions behavior**

Hierarchical to: No other components

FMT_MOF.1.1 The TSF shall restrict the ability to [*modify the behavior of*] the functions [

- *Lock the hardware key (LOCK_R)*
- *Modify the Hardware password (HWPWD_R)*
- *Modify the FAILRSTPWD password (FAILRSTPWD)*
- *Modify the MAXBLK_R register*
- *Modify the CONFIG_R register in order to enable chip clear operation*
- *Execute the Pw_check[HW] command*
- *Execute the Pw_check[FailReset] command*

] to [*the Administrator role*]

Dependencies: FMT_SMR.1 Security roles

**FMT_MSA.1 Management of security attributes**

Hierarchical to: No other components

FMT_MSA.1.1 The TSF shall enforce the [*cryptographic operation access controls*] to restrict the ability to [*initialize*] the security attributes [*user password and key_mode byte*] to [*the user key owner*].

Dependencies: FDP_ACC.1 Subset access control; FMT_SMR.1 Security roles.

**FMT_MSA.2 Secure security attributes**

Hierarchical to: No other components

FMT_MSA.2.1 The TSF shall ensure that only secure values are accepted for security attributes.

Dependencies: ADV_SPM.1 Informal TOE security policy model; FDP_ACC.1 Subset access control; FMT_MSA.1 Management of security attributes; FMT_SMR.1 Security roles.

**FMT_MSA.3 Static attribute initialisation**

Hierarchical to: No other components

FMT_MSA.3.1 The TSF shall enforce the [*cryptographic operation access controls*] to provide [*un-initialized*] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the [*user key owner*] to specify alternative initial values to override the default values when an object or information is created.

Application Note: In the TOE initial system configuration, the hardware private key and hardware password are stored in the chip and no user keys are stored in the chip. In this "default" initialization mode, only the hardware private key is available for signing and decode commands. The decrypt command is not available. A user key owner must decode a key, password, and key_mode byte that was previously encrypted with the hardware public key in order to load a user key and override the default values.

Dependencies: FDP_ACC.1 Subset access control; FMT_SMR.1 Security roles.

**FMT_MTD.1 Management of TSF data**

Hierarchical to: No other components.

FMT_MTD.1.1; 1 The TSF shall restrict the ability to [*see table below*] the [*see table below*] to [*see Table below*].

| Operation | TSF Data | Role |
|---|---|---|
| Initialize, Modify | • Hardware key<br>• Hardware key password<br>• Password failure counter reset password | Administrator |
| Initialize, Modify | • Password failure counter<br>• Password flags<br>• Maximum block size<br>• CONFIG_R register in order to enable chip clear operation<br>• Lock_R register to lock the hardware key | Administrator |
| Initialize | • User password<br>• User key_mode byte | User |

Dependencies: FMT_SMR.1 Security roles

**FMT_SMR.1 Security roles**

Hierarchical to: No other components.

FMT_SMR.1.1 The TSF shall maintain the roles: [*administrator; user*]

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

*Application note:* *Association of any user with a role will be performed through verification of the password supplied by the user when any operation is requested. Administrators will be required to supply the hardware password or FAILRSTPSW password, which will be verified by comparing it to the password maintained in registers within the crypto chip.*

Dependencies: FIA_UID.1 Timing of identification

### 5.1.5 FPT – Protection of the TOE Security Functions

**FPT_RVM.1 Non-bypassability of the TSP**

Hierarchical to: No other components.

FPT_RVM.1.1 The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

No dependencies.

**FPT_SEP.1 TSF domain separation**

Hierarchical to: No other components.

FPT_SEP.1.1 The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.1.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

No dependencies.

**FPT_TDC.1 Inter-TSF basic TSF data consistency**

Hierarchical to: No other components.

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret [*user password and user key_mode byte]* when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use [*CRC error coding*] when interpreting the TSF data from another trusted IT product.

No dependencies.

### 5.1.6 Strength of Function Requirement

The threat level for the TOE authentication function is assumed to be SOF-basic.

## *5.2 SECURITY FUNCTIONAL REQUIREMENTS FOR THE IT ENVIRONMENT*

**FCS_CKM.1 Cryptographic key generation**

Hierarchical to: No other components.

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*RSA*] and specified cryptographic key sizes: [512 -*1024 bits*] that meet the following: [*PKCS-1*].

**FCS_COP.1 Cryptographic operation**

Hierarchical to: No other components.

FCS_COP.1.1;1 The TSF shall perform [*hashing*] in accordance with a specified cryptographic algorithm [*SHA-1*] and cryptographic key sizes [*none*] that meet the following: [*PKCS-1*].

FCS_COP.1.1;2 The TSF shall perform [*encryption of key pairs and data*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*512-1024 bits*] that meet the following: [*PKCS-1*].

Application Note: Note that FCS_COP.1.1;2 refers to encryption of the “blob” of data that is input to the Decode command or the encryption of the data string that is decrypted in the Decrypt command. The data “blob” includes key pair data, a key_mode byte, a password at the discretion of the user, and other control data. The format of the blob that is encrypted is provided in the TOE functional specification. This requirement specifies the successful encryption of the blob or some other string of user data using RSA; correct formatting is essential for the Decode or Decrypt commands to successfully complete, but that is not a security requirement and is therefore not covered.

## *5.3 TOE SECURITY ASSURANCE REQUIREMENTS*

The Security Assurance Requirements for the TOE are the assurance components of Evaluation Assurance Level 3 (EAL3) augmented with ADV_SPM.1, Informal security policy model. The augmentation is necessary because ADV_SPM.1 is a dependency for FMT_MSA.2. None of the assurance components is refined. The assurance components are listed in Table 5.2.

**Table 5.2 - Assurance Components**

| **Assurance Class** | | **Assurance Components** |
|---|---|---|
| **Configuration Management** | | |
| | ACM_CAP.3 | Authorisation controls |
| | ACM_SCP.1 | TOE CM coverage |
| **Delivery and Operation** | | |
| | ADO_DEL.1 | Delivery procedures |
| | ADO_IGS.1 | Installation, generation, and start-up procedures |
| **Development** | | |
| | ADV_FSP.1 | Informal functional specification |
| | ADV_HLD.2 | Security enforcing high-level design |
| | ADV_RCR.1 | Informal correspondence demonstration |
| | ADV_SPM.1 | Informal security policy model **(augmentation)** |
| **Guidance Documents** | | |
| | AGD_ADM.1 | Administrator guidance |
| | AGD_USR.1 | User guidance |
| **Life Cycle Support** | | |
| | ALC_DVS.1 | Identification of security measures |
| **Tests** | | |
| | ATE_COV.2 | Analysis of coverage |
| | ATE_DPT.1 | Testing: high-level design |
| | ATE_FUN.1 | Functional testing |
| | ATE_IND.2 | Independent testing – sample |
| **Vulnerability Assessment** | | |
| | AVA_MSU.1 | Examination of guidance |
| | AVA_SOF.1 | Strength of TOE security function evaluation |
| | AVA_VLA.1 | Developer vulnerability analysis |

### 5.3.1 Class ACM: Configuration management

#### ACM_CAP.3 Authorisation controls

Objectives
A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labeling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Providing controls to ensure that unauthorized modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

Dependencies: ACM_SCP.1 TOE CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.3.1D The developer shall provide a reference for the TOE.

ACM_CAP.3.2D The developer shall use a CM system.

ACM_CAP.3.3D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.3.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.3.2C The TOE shall be labelled with its reference.

ACM_CAP.3.3C The CM documentation shall include a configuration list and a CM plan.

ACM_CAP.3.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.3.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.3.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.3.7C The CM plan shall describe how the CM system is used.

ACM_CAP.3.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.3.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.3.10C The CM system shall provide measures such that only authorised changes are made to the configuration items.

Evaluator action elements:

ACM_CAP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ACM_SCP.1 TOE CM coverage**

Objectives

A CM system can control changes only to those items that have been placed under CM. Placing the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Dependencies: ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_SCP.1.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.1.1C The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, and CM documentation.

ACM_SCP.1.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

### 5.3.2 Class ADO: Delivery and operation

**ADO_DEL.1 Delivery procedures**

Dependencies: No dependencies.

Developer action elements:

ADO_DEL.1.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.1.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements:

ADO_DEL.1.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

Evaluator action elements:

ADO_DEL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ADO_IGS.1 Installation, generation, and start-up procedures**

Dependencies: AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E The evaluator shall determine that the installation, generation**,** and start-up procedures result in a secure configuration.

### 5.3.3 Class ADV: Development

**ADV_FSP.1 Informal Functional Specification**

Dependencies: ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.1.1D The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP.1.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.1.2C The functional specification shall be internally consistent.

ADV_FSP.1.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.

ADV_FSP.1.4C The functional specification shall completely represent the TSF.

Evaluator action elements:

ADV_FSP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

**ADV_HLD.2 Security enforcing high-level design**

Dependencies: ADV_FSP.1 Informal functional specification

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall be internally consistent.

ADV_HLD.2.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.2.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.2.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_HLD.2.9C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator action elements:

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

**ADV_RCR.1 Informal correspondence demonstration**

Dependencies: No dependencies.

Developer action elements:

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

Evaluator action elements:

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ADV_SPM.1 Informal security policy model**

This assurance requirement is included because it is a dependency of FMT_MSA.2.

Dependencies: ADV_FSP.1 Informal functional specification.

Developer action elements:

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C. The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

Evaluator action elements:

ADV_SPM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

### 5.3.4 Class AGD: Guidance Documents

**AGD_ADM.1 Administrator guidance**

Dependencies: ADV_FSP.1 Informal functional specification

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**AGD_USR.1 User guidance**

Dependencies: ADV_FSP.1 Informal functional specification

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C The user guidance shall describe the use of user-accessible security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation supplied for evaluation.

AGD_USR.1.6C The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

### 5.3.5 Class ALC: Life cycle support

#### ALC_DVS.1 Identification of security measures

Dependencies: No dependencies.

Developer action elements:

ALC_DVS.1.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.1.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator action elements:

ALC_DVS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E The evaluator shall confirm that the security measures are being applied.

### 5.3.6 Class ATE: Tests

#### ATE_COV.2 Analysis of coverage

Objectives

In this component, the objective is to establish that the TSF has been tested against its functional specification in a systematic manner. This is to be achieved through an examination of developer analysis of correspondence.

Application notes

The developer is required to demonstrate that the tests which have been identified include testing of all of the security functions as described in the functional specification. The analysis should not only show the correspondence between tests and security functions, but should provide also sufficient information for the

evaluator to determine how the functions have been exercised. This information can be used in planning for additional evaluator tests. Although at this level the developer has to demonstrate that each of the functions within the functional specification has been tested, the amount of testing of each function need not be exhaustive.

Dependencies:

ADV_FSP.1 Informal functional specification

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ATE_DPT.1 Testing: high-level design**

Objectives

The subsystems of a TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

Application notes

The developer is expected to describe the testing of the high-level design of the TSF in terms of subsystems. The term subsystem is used to express the notion of decomposing the TSF into a relatively small number of parts.

Dependencies: ADV_HLD.1 Descriptive high-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.1.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.1.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

Evaluator action elements:

ATE_DPT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ATE_FUN.1 Functional testing**

Objectives

The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies: No dependencies.

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

**ATE_IND.2 Independent testing – sample**

Objectives

The objective is to demonstrate that the security functions perform as specified. Evaluator testing includes selecting and repeating a sample of the developer tests.

Application notes

The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include such things as machine-readable test documentation, test programs, etc.

This component contains a requirement that the evaluator has available test results from the developer to supplement the programme of testing. The evaluator will repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE in a different manner. By using a platform of validated developer test results the evaluator is able to gain confidence that the TOE operates correctly in a wider range of conditions than would be possible purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised particular concerns.

Dependencies: ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

### 5.3.7 Class AVA: Vulnerability Assessment

#### AVA_MSU.1 Examination of guidance

Objectives

The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect.

Dependencies: ADO_IGS.1 Installation, generation, and start-up procedures

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.1.1D The developer shall provide guidance documentation.

Content and presentation of evidence elements:

AVA_MSU.1.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.1.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.1.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.1.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

Evaluator action elements:

AVA_MSU.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.1.2E The evaluator shall repeat all configuration and installation procedures to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.1.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

**AVA_SOF.1 Strength of TOE security function evaluation**

Dependencies: ADV_FSP.1 Informal functional specification

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

Content and presentation of evidence elements:

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that the strength claims are correct.

**AVA_VLA.1 Developer vulnerability analysis**

Objectives

A vulnerability analysis is performed by the developer to ascertain the presence of obvious security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

Dependencies: ADV_FSP.1 Informal functional specification

ADV_HLD.1 Descriptive high-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.1.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.1.2D The developer shall document the disposition of obvious vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.1.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

Evaluator action elements:

AVA_VLA.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.1.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the obvious vulnerabilities have been addressed.

# 6 TOE SUMMARY SPECIFICATION

## *6.1 IT SECURITY FUNCTIONS*

The primary security functions of the TOE are listed below:

- Cryptographic Operation (CO), including:
  - Generation of the hardware public key (CO-1)
  - RSA Digital Signature (CO-2)
  - Data Decryption (CO-3)
  - Key Decryption (CO-4)
  - Key Destruction (CO-5)
- Access Control (AC)
- Authentication (AU), including:
  - Authentication failure (AU-1)
  - Timing of authentication (AU-2)
  - Timing of identification (AU-3)
- Security Management (SM)
  - Management of TOE Functions and Data (SM-1)
  - Roles (SM-2)
  - Security Attributes (SM-3)
- System Architecture (SA)
  - Interfaces (SA-1)
  - Cyclic Redundancy Check (SA-2)

These security functions map to functional and assurance requirements as shown in Table 6.1. The IT security functions are labeled in the text titles for reference in Section 8 of this document. Please note that where specific chip commands are referenced in the text below, the definition of the command and the meaning of command parameters are provided in Table 2.2.

**Table 6.1 – Security Functions mapped to Functional and Assurance Requirements**

| TOE Security Function | Sub-function | Requirement | Requirement Name |
|---|---|---|---|
| Cryptographic Operation (CO) | CO-1 | FCS_CKM.1 | Cryptographic key generation |
| | CO-5 | FCS_CKM.4 | Cryptographic key destruction |
| | CO-2<br>CO-3<br>CO-4 | FCS_COP.1 | Cryptographic operation |
| Access Control (AC) | AC | FDP_ACC.1 | Subset access control |
| | AC | FDP_ACF.1 | Security attribute based access control |
| | AC | FDP_ITC.2 | Import of user data with security attributes |
| Authentication (AU) | AU-1 | FIA_AFL.1 | Authentication failure handling |
| | AU-2 | FIA_UAU.1 | Timing of authentication |
| | AU-3 | FIA_UID.1 | Timing of identification |
| Security Management (SM) | SM-1 | FMT_MOF.1 | Management of security functions behavior |
| | SM-3 | FMT_MSA.1 | Management of security attributes |
| | SM-3 | FMT_MSA.2 | Secure security attributes |
| | SM-3 | FMT_MSA.3 | Static attribute initialisation |
| | SM-2 | FMT_MTD.1 | Management of TSF Data |
| | SM-2 | FMT_SMR.1 | Security roles |
| System Architecture (SA) | SA-1 | FPT_RVM.1 | Non-bypassability of the TSP |
| | SA-1 | FPT_SEP.1 | TSF domain separation |
| | SA-2 | FPT_TDC.1 | Inter-TSF basic TSF data consistency |

### 6.1.1 Cryptographic Operation (CO)

There are five functions within the TOE related to cryptographic operation: Hardware public key generation, RSA Digital Signature, Data decryption, key decryption, and key destruction. Each of these functions, their related CC functional security requirements, and the TOE satisfaction of the requirements are described below. Specific data block formats and details command operations and registers are provided in the document Atmel AT90SP0801, which is proprietary and confidential.

#### 6.1.1.1 Hardware Public Key Generation (CO-1)

FCS_CKM.1 requires that the TSF generate cryptographic keys in accordance with the RSA key generation algorithm. The TOE does not store the hardware public key value; instead, when the hardware public key is read, it generates the hardware public key based on the hardware private key components stored in chip registers. Note that the TOE performs no other key generation operations and that no key pairs are generated by the TOE.

#### 6.1.1.2 RSA Digital Signature (CO-2)

FCS_COP.1 provides requirements for cryptographic operation that include performing RSA digital signature with cryptographic key sizes from 512-1024 bits. The TOE has two digital signature commands, PKCS_sign[k] and PKCS_sign[HW], depending upon whether a user key is specified or the hardware key is specified.

PKCS_sign[k] performs RSA digital signature with a user specified key that is indicated by the user key buffer number [k]. The user key would have been previously stored in the chip with a Decode_key command. The data to be signed must be sent to the I/O buffer by the system before the PKCS_sign command is executed. When the signature command is received, the buffer is padded to 1024 bits using standard PKCS#1 block type 1 padding and then signed using the private key k as specified in the command. The signature data can vary between 1 and 64 bytes

and should include both the message digest (hash) and the algorithm identifier. Note that hashing is not performed on the chip and is not part of the TOE. If the number of bytes within the buffer is not within the range of 1 through 64 when the signature command is executed, the chip will return error code 1 and the command will fail. The key_mode byte for each user key controls the way the password requirements interact with the use of the user key for signing. The key_mode byte is included in the key block. If the key_mode byte is set to require a password, then a password must have been entered before the execution of the PKCS_sign[k] command. A user password is entered by executing the Pw_check[k] command where [k] specifies the user key buffer number.

PKCS_sign[HW] command is identical to the PKCS_sign[k] command, except that it uses the hardware private key to sign instead of a user specified key. Entry of the hardware password is required prior to the execution of the PKCS_sign[HW] command. The hardware password is entered using the Pw_check[HW] command. The hardware private key is always 1024 bits in length.

#### 6.1.1.3 Data Decryption (CO-3)

FCS_COP.1 provides requirements for cryptographic operations for performing data decryption for RSA key sizes of 512 to 1024 bits. Decrypt_data[k] is used to decrypt small quantities of user information that have been encrypted with user public keys. The user key buffer number is specified by [k]. The user key would have been previously stored in the chip with a Decode_key command. As in the signing operations described above, if the key_mode byte is set to require a password, then a password must have previously been entered. The user may optionally specify in the key_mode byte whether decryption is allowed using the key. If the appropriate password has been supplied and if the key_mode byte indicates that the key may be used for decryption, then the chip uses the specified user key to decrypt the incoming 1024 bit value. It will also decrypt data from a 512-bit block if a 512-bit key is selected to do the decryption and the incoming data is first padded to 1024 bits with 0s. The chip looks in the decrypted data block for the 0x00 (end of padding) indicator to determine how many bytes have been encrypted. If the decrypted data is not properly formatted according to PKCS #1 (0x00, 0x02 followed by non-zero padding) or there are more than 32 bytes of data, the chip will return the error code 0x96: “Invalid Data Decryption Format.” After decryption, the data returned to the I/O buffer is:

- One byte of format information, signifying data length
- Five to thirty-two bytes of decrypted data, from the least significant bytes of the decrypted block
- Two bytes of CRC, computed over both format and data. (Note that the CRC function is used to verify the proper information is written to the chip, since data written to the buffer may not be read back.)

#### 6.1.1.4 Key Decryption (CO-4)

FCS_COP.1 provides requirements for cryptographic operations that include key decryption upon import of a block of specially formatted data. Decode_key[b,k], is a specialized decryption operation. The secure signature generation chip provides a capability for protecting and loading user keys. The chip is designed to provide the capability to use a trusted key (usually the Hardware Public Key, but another 1024 bit user key may also be used) to encrypt other user keys for safe storage off of the chip and to then securely decrypt those keys on the chip. The process is:

1) The public key (either Hardware public key or another user public key) is used to encrypt a user key, password, and a key_mode byte in a specified format.
2) The encrypted user key block is stored off chip (safely, since it is encrypted with the public key and can only be decrypted with the trusted private key),

3) When needed, the encrypted user key block is brought back into the chip. The Decode_key command is used to decrypt the user key block and load the user key to a buffer for signing operations.

The Decode_key[b,k] command decodes the appropriate piece(s) of the user private key structure using a private key stored on the chip. The decode result is stored in a user key buffer in EEPROM. Either the hardware key or any of the user keys can be used to perform this decryption. This source key is specified in the SOURCE_R register. The value within SOURCE_R is set to 0xFF on reset, indicating the hardware key as the default source for the private key that will be used to decrypt an incoming key block. If any key other than the hardware key is used as the source key, the password check flag for that key must be set by the appropriate Pw_check command before the key decode operation is attempted. Alternatively, the user key may have a key_mode byte that indicates that the password may be bypassed for decode operations. In this case, the password check flag does not have to be set for the decode command to succeed.

If the source key is not valid when the Decode_key operation is attempted or the source key ID number is the same as the target key the command will fail. Source keys must be 1024 bits in length. If the SOURCE_R register points to a 512 bit key, then subsequent key decode operations will not result in a valid user key being loaded into the buffer and error code 0x91 “Internal cryptographic error” will result. Keys to be decrypted may either be 512 or 1024 bits.

The decode command consists of two steps. In the first step, Decode_key[0,k] where k is the user key used for decryption, the chip decrypts the private modulus and the user key password and key mode byte and then stores them directly within EEPROM. In the second step, Decode_key[1,k], the public modulus is sent, unencrypted. All of the remaining pre-computed values are derived and written into the appropriate locations within the EEPROM. No other commands may be executed between the two decode commands or the decode operation may fail.

#### 6.1.1.5 Key Destruction (CO-5)

FCS_CKM.4 requires that the TOE provide a means to securely zeroize keys, thereby destroying them, in accordance with FIPS 140-1 standards. Key destruction is performed by setting the STATUS_R/bit 4 to “1” when: STATUS_R/bit 7 is “1”, CONFIG_R/bit 0 is “1”, and LOCK_R/bit 0 is “0”. The chip clear operation causes all internal keys, including the hardware key, to be zeroized. The hardware password and FAILRSTPWD password will also be reset to 0 and all password check flags are cleared.

### 6.1.2 Access Control (AC)

Access control is required to protect sensitive information and operations on the chip. FDP_ACC.1 and FDP_ACF.1 require the enforcement of cryptographic operation access controls on all system users for data and operations performed on that data. The chip provides access control by denying access to some data and operations and allowing access to other data and operations only with the entry of a correct password. Table 6.3 shows system access controls on commands and registers that meet the requirements of FDP_ACC.1 and FDP_ACF.1. Table 6.4 summarizes the Access Control Policy

**Table 6.3 – System Access Controls on Commands and Registers**

| **Command** | **Access Control** |
|---|---|
| Store (buffer register) | Access control is based on the controls on the registers that are the arguments of the command. For detailed controls on registers, see register controls below. |
| Load (buffer register) | Access control is based on the controls on the registers that are the arguments of the command. For detailed controls on registers, see register controls below. |
| Decode_key[b,k] | If the key used for decode operation is the hardware key, the hardware password must have previously been supplied. If the key used for decode is a user key, a password must be supplied only if the user key specifies that a password is required for the decode command. |
| PKCS_sign[k] | This command allows signature of data by a user key. Note that setting appropriate flags with a password supplied through the Pw_check[k] command is discretionary as specified by the user in the key_mode byte. |
| Pw_check[k] | This command compares the 64 bit value stored in the I/O buffer with the password written into the appropriate user key buffer. If they match, the corresponding user password check flag will be set. This flag is reset on power up, on system reset and when a new key is loaded into the buffer. This flag can also be reset using STATUS_R/bit 3. |
| Decrypt_data[k] | This command decrypts small quantities of user information that has been previously encrypted with user keys. Note that allowing data decryption with a particular key and setting appropriate flags with a password supplied through the Pw_check[k] command is discretionary as specified by the user in the key_mode byte. |
| Write | No access controls. |
| Read | No access controls. |
| **Register** | **Access Control** |
| VERS_R | Read-only. May be read by any user regardless of configuration or passwords. |
| HWPRIV_R | May be written only by administrator. Never readable. |
| HWPUB_R | Read-only. May be read by any user regardless of configuration or passwords. |
| HWPWD_R | May be written only by the administrator. Never readable. |
| LOCK_R | May be written only by the administrator. Always readable. |
| FAILRSTPWD_R | May be written only by the administrator. Never readable. |
| FAILCNT_R | Read-only. May be read by any user regardless of configuration or passwords. |
| MAXBLK_R | May be written only by the administrator. Always readable. |
| CONFIG_R | May be written only by the administrator. Always readable. |
| SOURCE_R | No access controls |
| STATUS_R | No access controls |
| ERROR_R | Read-only. May be read by any user regardless of configuration or passwords. |
| CRC0_R | Read-only. May be read by any user regardless of configuration or passwords. |
| CRC1_R | Read-only. May be read by any user regardless of configuration or passwords. |
| Constant_0 | Read-only. May be read by any user regardless of configuration or passwords. |

Square brackets indicate commands that have parameters in the second byte where:

- b represents the parameter block number for key decoding. These parameter blocks contain the information use to decode the user private keys.
- k represents the user key buffer number that is to be used by the command.
- HW indicates the hardware key or the hardware password, depending upon the command.
- FailReset indicates the Failed reset counter password.

**Table 6.4 – Summary of Access Control Policy**

| Operation | Objects | Access |
|---|---|---|
| Sign | Hardware Private Key | Owner (Administrator) |
| | User Private Key | Owner, World (at owner's discretion) |
| Decrypt | Hardware Private Key | Not applicable |
| | User Private Key | Owner, World (at owner's discretion) |
| Decode Key | Hardware Private Key | World |
| | User Private Key | Owner, World (at owner's discretion) |

FDP_ITC.2 requires that cryptographic operation access controls be enforced, as described above, and that certain rules be enforced when importing user data. Data is transferred to or from the I/O buffer on the chip using the SMBus interface. All bits are sent to or read from the chip most significant bit first. Only block reads and writes are supported. There is no read after a write on the I/O buffer; a CRC function is used to verify that the proper information is being written to the chip. All data read from the chip which has a size of greater than 32 bits contains a CRC value as the last two bytes, while smaller register reads do not include a CRC. The I/O buffer may be read twice to determine their validity.

Security attributes are associated with user keys when the user keys stored in the chip using the Decode_key command. These security attributes are the user password and key_mode byte. Note that these attributes may be null values at the user's decretion. The Decode_key commands associates the user key, password and key_mode byte and the data is stored appropriate buffers in the appropriate order.

### 6.1.3 Authentication (AU)

#### 6.1.3.1 Authentication Failure (AU-1)

FIA_AFL.1 requires that the TOE detect and respond to authentication failures. The chip uses the Pw_check[HW], the Pw_check[FailReset], and the Pw_check[k] commands to allow the user to enter a password. The commands check the hardware, FAILRSTPWD, and user passwords, respectively. For the Pw_check[HW] and Pw_check[FailReset] commands, the chip detects when 10 unsuccessful authentication attempts occurs.

The detection of 10 unsuccessful authentication attempts is one of two mechanisms to prevent an exhaustive attack on either the hardware or FAILRSTPWD passwords. The chip accomplishes the detection through a single attempt counter. Normally, the counter is set to 0. Each time there is a failed password check to the hardware or FAILRSTPWD passwords, the counter is incremented. If the correct password is entered, the counter is cleared. If the counter has a value of 9 or more before either the HW or FailReset password check command starts and the incorrect password is sent to the chip, the second protection mechanism begins: the FAILCNT_R is checked. If FAILCNT_R has a value of less than 224, it will be incremented to 224. If it is greater than 224, it will be incremented to the next multiple of 32. A value of any multiple of 32 in the FAILCNT_R register causes an immediate lockout. The multiple of 32 determines the length of the lockout as shown in the table 6.5 below.

The FAILCNT_R register also works independently of the single attempt counter. For cumulative unsuccessful password attempts for all passwords, including the hardware, FAILRSTPWD, and user passwords, the FAILCNT_R register is incremented by 1 for each unsuccessful attempt. Lockout periods apply when FAILCNT_R reaches each multiple of 32.

**Table 6.5 – Cumulative Password Failure Count and Lockouts**

| Cumulative Failure | Lockout Period |
|---|---|
| 32 | 1.2 Minutes |
| 64 | 2.4 Minutes |
| 96 | 4.8 Minutes |
| … | … |
| 224 | 1 Hour, 17 Minutes |
| 256 | 2 Hours, 34 Minutes |
| … | … |
| 384 | 1.7 Days |
| … | … |
| 512+ | 27.2 Days |

The power must be maintained during the lockout period as the lockout counter is kept in volatile memory. If power is removed during the lockout period, the timer starts again when the chip is powered up. During the lockout period, loads and stores and corresponding reads and writes of STATUS_R, as well as loads and reads of FAILCNT_R, ERROR_R and VERS_R are permitted. No other registers may be accessed and no other commands may be executed.

The FAILCNT_R register is cleared when either the Pw_check[HW] or the Pw_check[FailReset] commands are successfully executed. To provide adequate security, the passwords stored in these two registers should be random through all 64 bits and not be locatable in any password dictionary, as defined in the TOE Security Policy and in the Administrator's Guide. The chip does not provide any internal checks to ensure that passwords follow these policies. Because the failure delay mechanism is tied to the FAILCNT_R register, when it is reset, the failure penalty goes back to 1.2 minutes after the 32$^{nd}$ subsequent failure.

#### 6.1.3.2 Timing of Authentication (AU-2)

FIA_UAU.1 specifies that read, write, store, load, sign, decrypt, and decode operations can be used prior to user authentication, so long as that operation does not implicitly require that a password flag be set. The availability of these operations prior to authentication depends on 1) which register is being loaded or stored, or 2) which key is specified for the operation. Detailed rules for which registers can be loaded or stored prior to authentication are provided in Table 2.1. For sign, decrypt, and decode operations, keys may be decoded into the chip that require no password to be specified and allow all possible operations (i.e., decrypt data and sign operations). These operations are available prior to authentication of the user.

The administrator is authenticated by issuing the Pw_check[HW] command and supplying a password that matches the password in the HWPWD_R register. Users are authenticated by supplying the appropriate password that matches the password stored for the specified key register identifier.

#### 6.1.3.3 Timing of Identification (AU-3)

FIA_UID.1 specifies that that read, write, store, load, sign, decrypt, and decode operations that are available to the world can be used prior to user identification. For all other operations, the user must be identified. The user is identified by supplying the key register identifier in the command that is executed. User creation/initialization occurs when another user or the administrator issues the decode command on behalf of the new user. The decode command causes the chip to decrypt (using the hardware private key or another private key loaded on the chip) and load a user key pair, password, and access control field to a specified user key register. The access control field determines if the owner or the world can perform operations using the key. The owner is the new user, who may then issue commands that specify the appropriate key identifier.

### 6.1.4 Security Management (SM)

#### 6.1.4.1 Management of TOE Functions and Data (SM-1)

Security management defines the protection and management mechanisms of the TOE. FMT_MOF.1 defines specific functions that are accessible to the administrator to assist in security management. The administrator is defined as the owner of the hardware and FAILRSTPW passwords, therefore, the administrator has access provided by these two passwords. These functions include:

- Load or change the hardware private key, by first specifying the hardware password to the system using the Pw_check[HW] command to set the CMOS HW password check flag and then using the Store command to store the hardware private key to the HWPRIV_R register. Note that the LOCK_R register must be zero and that the hardware private key must be 1024 bits.
- Lock the hardware key so that it may not be changed .
- Change the hardware password. Note that either STATUS_R/Bit 7 or CONFIG_R/Bit 1 must be 1 for the hardware password to be writable.
- Reset the Failed Password Counter by successful executing either the Pw_check[HW] or the Pw_check[FailReset] commands.
- Change the FAILRSTPWD password. Note that either STATUS_R/Bit 7 or CONFIG_R/Bit 1 must be one for the FAILRSTPWD password to be writable.
- Change the maximum size of the read block (default 32) to another value. Note that STATUS_R/Bit 7 must be one for this register to be writable.
- Modify the CONFIG_R register in order to enable chip clear operations and write hardware and FAILRSTPWD passwords. Note that STATUS_R/Bit 7 must be one for this register to be writable.

#### 6.1.4.2 Roles (SM-2)

FMT_MTD.1 and FMT_SMR.1 define security roles and specific functions accessible to those roles. Two roles are defined: administrator and user. The administrator is any user who has the hardware passwords. A user may be any user of the chip. A user may or may not have user passwords associated with keys used for signing and decryption. Note that the system administrator is required to initialize the system. The administrator has access to the functions described above. The user has access only to those registers that allow read or read/write access, as defined in Table 6.3, above, to sign and decrypt operations using user keys loaded with or without passwords, and to decode key operations with or without passwords. Note that the administrator initializes the FAILRSTPWD and is considered the owner of that password, however, at the administrator's discretion, the password may be shared with other users.

#### 6.1.4.3 Security Attributes (SM-3)

FMT_MSA.1, FMT_MSA.2, and FMT_MSA.3 define requirements for security attributes. Attributes are associated with user keys. Security attributes are the password and key_mode byte. Attributes may be null values at the user's discretion. The user initializes the security attributes when the associated key is loaded in the system. All user keys are initialized through the Decode_key command. Security attributes are loaded with the key as part of the decode command. Security attributes and the key must have previously been encrypted with another public key in a specified format. Once the keys and attributes are stored on the chip, no other user has access to the data, since keys are overwritten when data is loaded and the data may not be read.

### 6.1.5 System Architecture (SA)

#### 6.1.5.1 Interfaces and Communication (SA-1)

FPT_RVM.1 and FPT_SEP.1 require non-bypassability of the TSP and TSF domain separation. The TOE allows access to users only through defined commands and registers. Access to commands and registers have restrictions, as defined in Table 6.3 above. The TOE provides no access to the TOE program memory, the crypto engine or buffers containing intermediate results. All external communication is through the SMBus, through the I/O buffer. Separation between the security domains of subjects is accomplished by requiring the hardware or FAILRSTPWD password to access certain functions. Through this mechanism, the user has accessed to a limited set of commands and registers, while the Administrator has additional access. Note that the administrator has no access to commands using user keys if a password is specified and that password is unknown to the Administrator. Users may specify passwords associated with key pairs and no other user would be able to execute sign and decryption commands with those keys unless they had the password.

#### 6.1.5.2 Cyclic Redundancy Check (SA-2)

FPT_TDC requires that the system provide data consistency. This requirement is met through the CRC function, which is used to verify that the proper information is being written to the chip, since data written to the I/O buffer may not be read back. All data read from the chip that has a size of greater than 32 bits (public keys, signatures, encoded keys, decrypted data, etc.) contains a CRC value as the last two bytes, while smaller register reads do not include a CRC. The I/O buffer may be read twice to determine validity. In the case of internally generated cryptographic data that is written to the buffer by the chip and read by the external PC system, the process works in reverse. The chip will compute a CRC and append it to the end of the data in the buffer. If the CRC computed over the data by the PC system does not match that read from the chip's buffer the system should retry the read of the buffer. The details of the CRC equation are included in the Atmel AT90SP0801.

### 6.1.6 Strength of Function Requirement

The threat level for the TOE authentication function is assumed to be SOF-basic. This defines a level of authentication strength of function where analysis shows that the function provides basic protection against straightforward or intentional breach of TOE security by attackers possessing a minimum attack potential.

The hardware and FAILRSTPWD passwords are 64 bits each. In compliance with the Administrator's Guide, the passwords:

- Must be exactly 8 characters long and must not be zeros.
- Must contain alphanumeric characters only.
- Must not be a common word, a word in any existing password dictionaries, or a word easily guessed (such as "password").

User passwords are discretionary, however, in compliance with the User's Guide, users must use passwords and follow the same guidelines for selecting passwords as those of the administrator, listed above.

Analysis was performed using the following assumptions:

- It is assumed that the SOF analysis is limited to users who choose to specify passwords, i.e., user passwords are discretionary and if a user decides not to use a password as an authentication mechanism, than no SOF claims are made. Note that the hardware

password and FAILRSTPWD must be supplied as part of chip initialization; these passwords are not discretionary

- It is assumed that the administrators and the users will follow password guidelines listed above.
- It is assumed that attackers would have access to commonly available password crackers, particularly those that use dictionary and exhaustive search attacks.
- It is assumed that the environment provides protections such that passwords could not be captured en route to the chip, therefore the analysis covers only those attacks that guess passwords or retrieve them from the TOE through some vulnerability; the scope of the SOF analysis is the TOE.
- Although words easily guessed or those in a known password dictionary are not supposed to be used, in order to present the worst-case scenario, it is assumed that users will select natural language passwords, thereby greatly reducing the possible passwords that could be used.
- It is assumed that users would clear their keys from the system after user to avoid an exhaustive attack on user key passwords. This is recommended in the user guide. User passwords are therefore not considered in the analysis.
- It is assumed that natural language passwords number no greater than 100,000.
- It is assumed that the attacker would first use a dictionary attack that would include common strategies for guessing passwords such as selecting a user login name, pAsSwOrD, simple transformations for common words, etc.
- Motivation of the attacker is not considered as part of this analysis because the system is multi-purpose and there is no way of knowing the value of the assets protected by the TOE. It is assumed that the value of the assets is low and therefore motivation on the part of the attacker is moderate to low.
- It is assumed that there are no time limitations on the attacker.

The chip does not allow reads to the registers that hold the Hardware, FAILRSTPWD, and user passwords. Based on the vulnerability analysis, there is no obvious vulnerability such as buffer overflow, etc, that would allow an attacker access to these registers. Therefore, the SOF analysis focused on password “cracker” attacks. With 100,000 password possibilities, on average, the cracker would guess the password in 50,000 tries.

If there were no other protections, it would be relatively simple to break the password mechanism in a short time with 50,000 tries. However, the TOE has authentication failure protection on the hardware and FAILRSTPWD, which locks the attacker out of the system on the $10^{th}$ incorrect password supplied for a period of one hour and 17 minutes. Thereafter, every single subsequent failed attempt causes the lockout period to double. The lockouts do not stop until a correct Hardware or FAILRSTPWD is entered.

Given the authentication failure protection mechanism, it would take months to crack the password, given that the attacker on average must try 50,000 passwords and the lockout periods double with each lockout, i.e., the first lockout after 10 tries is one hour and 17 minutes, the next password try lockout period would be two hours and 34 minutes, etc.

The attack potential for the TOE authentication mechanism was scored using Table B.3 in Annex B.8 of the CEM. The attack potential was scored as 17, i.e., a layman with no equipment (score of 0) would take more than a month to guess the password, with the score of 8 for elapsed time and 9 for Access to the TOE (total of 17). The calculated attack potential is therefore a 17, which corresponds to SOF-basic.

## *6.2 SECURITY FUNCTIONS FOR THE IT ENVIRONMENT*

The IT environment requires that certain cryptographic functions be performed that are not performed on the chip. These functions include:

- Generation of the hardware key pair
- Generation of user key pairs
- Hashing
- Encryption of the "blob" including user keys, key_mode byte, and, at the discretion of the user, password; the blob is input to the Decode command.
- Encryption of data that is input to the Decrypt command.

This functionality is covered by the FCS_CKM.1 and FCS_COP.1 iterations provided under Section 5.2, Security Functional Requirements for the IT Environment. FCS_CKM.1 1 requires that the environment generate cryptographic keys in accordance with the RSA key generation algorithm. Hardware keys must be 1024 bits. User keys may be either 512 or 1024 bits. FCS_COP.1 specifies that the environment will perform hashing in accordance with SHA-1 and that RSA encryption will be used to encrypt blobs (as defined above) and data.

## *6.3 ASSURANCE MEASURES*

The assurance level selected for the TOE was EAL3 because it provides appropriate assurance measures for the expected application of the product. EAL3 ensures sound development practices and requires a moderate level of independently assured security. In the PC industry, this level of security is required for digital signature applications that are expected to use the TOE.

Appropriate assurance measures will be employed to satisfy the security assurance requirements. The evaluation will confirm whether the assurance measures are sufficient to satisfy the assurance requirements. The assurance measures will consist of the set of evaluation evidence listed in Table 6.6, below. The documents listed in the table will be used as to satisfy assurance evaluation requirements.

**Table 6.6 – Assurance Evaluation Evidence**

| Component | Deliverables |
|---|---|
| ACM_CAP.3 & ACM_SCP.1 | 7026 - General Quality Specification For Major Change Determination, Notification, And Response, Rev. O<br>7028 - General Quality Specification For Records And Archives, Rev. S<br>7037 - General Quality Specification For Product Configuration Management, Rev. L<br>7040 - General Quality Specification For Document Control, AF<br>100-022 – Wafer Fabrication Process Routes, Flow, and Pattern Mask Lists, Rev. R<br>100-045 – Instructions for Submission of Changes to Controlled Documents (ECN Record), Rev. Q<br>100-055 – Bill-of-Materials, Rev X<br>1400-030 – Work Instructions for Microsoft WORD Systems, Rev. B<br>1400-031 – Work Instructions for COMETS Systems, Rev. A<br>800-008 – IBM Secure Signature Chip Datasheet Distribution Procedure Common Criteria Evaluation, Rev. A<br>800-012 – Tracking Configuration Management Requirements, Rev. B<br>100-056 – Part Number Generation, Rev. AK |
| ADO_DEL.1 | 793-300 – Standard Operating Procedures for Shipping, Rev F<br>IBM Corporation, IBM Personal Computer Company Embedded Cryptographic Test Plan, Andy Trotter/Scott Elliott, Version .8, 18 January 1999 |
| ADO_IGS.1 | 800-004 – IBM Secure Signature Chip Secure Installation Generation Startup Procedures CC Eval, Rev B |
| ADV_FSP.1 | ATMEL Secure Signature Generation Chip AT90SP0801, 1495AX-07/05/01 |
| ADV_HLD.2 | ATMEL Secure Signature Generation Chip AT90SP0801, 1495AX-07/05/01 |
| ADV_RCR.1 | 800-005 – IBM Secure Signature Chip Representation Correspondence CC Eval, Rev B |
| ADV_SPM.1 | 800-006 – IBM Secure Signature Chip Security Policy Model CC Eval, Rev B |
| AGD_ADM.1 | 800-002 – IBM Secure Signature Chip Admin Guide CC Eval, Rev B |
| AGD_USR.1 | 800-001 – IBM Secure Signature Chip User Guide CC Eval, Rev B |
| ALC_DVS.1 | 800-011 – IBM Secure Signature Chip Lifecycle Management CC Eval, Rev A |
| ATE_COV.2 | 800-010 – IBM Secure Signature Chip Verification CC Eval, Rev A<br>800-009 – IBM Secure Signature Chip Test Setup Procedure CC Eval, Rev A<br>Test Scripts and Test Results |
| ATE_DPT.1 | 800-010 – IBM Secure Signature Chip Verification CC Eval, Rev A<br>800-009 – IBM Secure Signature Chip Test Setup Procedure CC Eval, Rev A<br>Test Scripts and Test Results |
| ATE_FUN.1 | 800-009 – IBM Secure Signature Chip Test Setup Procedure CC Eval, Rev A<br>Test Scripts and Test Results |
| ATE_IND.2 | 800-009 – IBM Secure Signature Chip Test Setup Procedure CC Eval, Rev A<br>TOE for testing |
| AVA_MSU.1 | 800-002 – IBM Secure Signature Chip Admin Guide CC Eval, Rev B<br>800-001 – IBM Secure Signature Chip User Guide CC Eval, Rev B<br>800-004 – IBM Secure Signature Chip Secure Installation Generation Startup Procedures CC Eval, Rev B<br>793-300 – Standard Operating Procedures for Shipping, Rev F |
| AVA_SOF.1 | 800-007 – IBM Secure Signature Chip Security Target CC Eval, Rev B |
| AVA_VLA.1 | 800-003 – IBM Secure Signature Chip Vulnerability Analysis CC Eval, Rev B |

# 7 PP CLAIMS

This Security Target was not written to address any existing Protection Profile.

# 8 RATIONALE

## *8.1 SECURITY OBJECTIVES RATIONALE*

The first section shows that all of the secure usage assumptions, organizational security policies, and threats to security have been addressed. The second section shows that each IT security objective and each non-IT security objective counters at least one assumption, policy, or threat.

### 8.1.1 All Assumptions, Policies and Threats Addressed

Table 8.1 shows that all the identified Threats to Security have been addressed. Table 8.2 shows that all of the Organizational Security Policies have been addressed. Table 8.3 shows that all of the Secure Usage Assumptions have been addressed. The rationale for each of these mappings in discussed below.

**Table 8.1 – All Threats to Security Countered by Objectives**

| | Threat Name | Threat Description | Objective |
|---|---|---|---|
| 1 | T.ACCESS | An authorized user may be able to access or modify sensitive data for which that user is not authorized to access. | O.AUTHENTICATION |
| 2 | T.ACCESS_UNAUTH | An unauthorized person may be able to gain access to the system and to data. | O.I&A |
| 3 | T.BYPASS | An attacker may be able to bypass the TOE security functionality and gain unauthorized access to keys, data, and operations. | O.NONBYPASS |
| 4 | T.CRYPTO_OP | Cryptographic algorithms and/or cryptographic key sizes may be insufficient, allowing them to be deciphered by users that are not authorized access to the encrypted data. | O.CRYPTO_OP |
| 5 | T.CRYPTO_MGT | Incorrect cryptographic key destruction may cause an inadvertent disclosure of sensitive information. | O.CRYPTO_KEY |
| 6 | T.DIG_SIG | Incorrect use of the digital signature and decryption capabilities of the chip could cause a loss of sensitive data, e.g., if a user application improperly supplied a public key to the chip to use in signing when a private key was required or vice versa. | O.DIG_SIG |
| 7 | T.IMPLEMENT | The TOE may not adequately protect cryptographic keys and data stored in the TOE, causing loss of confidentiality and integrity. | O.CONF_INTEGRITY |
| 8 | T.TAMPER | An attacker may be able to tamper with TSF data or programs. | O.SELF_PROTECT |

Threats T.ACCESS is addressed by the capability to associate a password known to a user to a key pair or set of operations. By associating a password to a key pair or set of operations, the system counters the threat of an authorized user accessing data that he/she is not authorized to access. Access can only be gained with an appropriate password.

T.ACCESS_UNAUTH is countered by the ability of the TOE to authenticate users prior to granting access to system assets or operations. An unauthorized user can impersonate a user only if the unauthorized user has a valid password.

T.BYPASS is countered by O.NONBYPASS, which ensures that an unauthorized user or other attacker is prevented from accessing the TOE by ensuring that TOE security functionality such as identification and authentication cannot be bypassed.

T.CRYPTO_OP is addressed by O.CRYPTO_OP, which specifies that all cryptographic operations must be performed in accordance with specified algorithms with cryptographic keys in a specified size in accordance with PKCS-1 standards.

T.CRYPTO_MGT is countered by O.CRYPTO_KEY which specifies that cryptographic key destruction must be performed in accordance with PKCS-1.

T.DIG_SIG is addressed by O.DIG_SIG, which specifies that sufficient guidance must be given in the use of the TOE, in the form of documentation and TOE operation, to prevent incorrect use of the digital signature and decryption capabilities of the TOE.

T.IMPLEMENT is countered by O.CONF_INTEGRITY. O.CONF_INTEGRITY states that the confidentiality and integrity of key pairs and sensitive data stored in the TSF must be protected.

T.TAMPER is addressed by O.SELF_PROTECT. This objective ensures that the TOE maintains its own protected domain. This protected domain will provide protection for critical operations and elements so they can be better protected in the case of a software or hardware malfunction.

**Table 8.2 – All Security Policies Addressed by Objectives**

| | Policy Name | Policy Description | Objective |
|---|---|---|---|
| 1 | P.TSP | A TOE security policy (TSP) must identify all roles, services, and security-relevant data items, and specify what access (if any) a user, performing a service within the context of a given role, has to each of the security-relevant data items. | O.ENFORCEPOLICY |

P.TSP defines a requirement for a TOE security policy model, which is addressed by O.ENFORCEPOLICY. O.ENFORCEPOLICY ensures that the TOE defines and implements a TOE security policy model.

Assumptions mapped to objectives in Tables 8.3 and 8.4 are self explanatory.

**Table 8.3 – All Secure Usage Assumptions Met by Objectives**

| | Assumption Name | Assumption Description | Objective |
|---|---|---|---|
| 1 | A.CONFIGURATION | The IBM Secure Signature Generation Chip will be properly installed and configured according to the System Administrator's Guide. | O.ADMIN<br>OE.CONFIGURATION |
| 2 | A.THREAT_LEVEL | The threat level for the TOE authentication function is assumed to be SOF-basic. | O.SOF |

**Table 8.4 – All Assumptions for the IT Environment Met by IT Environment Objectives**

| | Assumption Name | Assumption Description | Objective |
|---|---|---|---|
| 1 | AE.CONFORMANCE | The use of the TOE does not guarantee the security of the overall system. The responsible authority in each user organization shall assure that the organization's computer or telecommunication systems provide an acceptable level of security for the given application and environment. | OE.CONFORMANCE |
| 2 | AE.OFF_CHIP | The following functions are preformed off-chip and must be performed securely and correctly: hardware key pair generation; user key pair generation; hashing; encryption of user key pairs, passwords, and key_mode byte as applicable for the chip Decode function; encryption of data for the chip Decrypt function. | OE.OFF_CHIP |
| 3 | AE.PHYSICAL_PROTECTION | The TOE provides no protection against physical threats such as simple power analysis, differential power analysis, external signals, extreme temperature, or physical tampering. Physical protection is assumed to be provided by the environment. | OE.CONFORMANCE |

### 8.1.2 All Objectives Necessary

Table 8.5 shows that there are no unnecessary IT security objectives for the TOE, since each objective addresses at least one threat, organizational security policy, or secure usage assumption. Mappings are discussed in the previous section.

**Table 8.5 – All IT Security Objectives for the TOE Necessary**

| | Objective Name | Objective Description | Threat/Policy/ Assumption |
|---|---|---|---|
| 1 | O.AUTHENTICATION | The TOE must be able to provide the capability of associating a password or PIN known to a user to a key pair or set of system capabilities to support access control. | T.ACCESS |
| 2 | O.ADMIN | The TOE must provide functionality that enables an authorised administrator to configure the system in accordance with a specified TOE security policy. | A.CONFIGURATION |
| 3 | O.CRYPTO_KEY | The TOE must perform cryptographic key destruction in accordance with PKCS-1 | T.CRYPTO_MGT |
| 4 | O.CRYPTO_OP | The TOE must perform cryptographic operations, including RSA digital signature, RSA decryption, and public key generation in accordance with specified algorithms and cryptographic keys of a specified size, in accordance with PKCS-1 and of sufficient size to protect private/public key pairs from deciphering. | T.CRYPTO_OP |
| 5 | O.CONF_INTEGRITY | The TOE must ensure the confidentiality and integrity of key pairs and sensitive data stored in the TSF. | T.IMPLEMENT |
| 6 | O.DIG_SIG | Sufficient guidance must be provided to application developers to allow correct and secure usage of the digital signature and decryption capabilities of the chip. | T.DIG_SIG |
| 7 | O.ENFORCEPOLICY | The TOE must enforce a clearly defined and documented informal TOE security policy (TSP) model. | P.TSP |
| 8 | O.I&A | The TOE must authenticate a password or PIN entered by a user before granting access to cryptography-related IT assets and operations within the TOE. | T.ACCESS_UNAUTH |
| 9 | O.NONBYPASS | The TOE shall ensure that the TOE security functionality cannot be bypassed | T.BYPASS |
| 10 | O.SELF_PROTECT | The TSF will maintain a domain for its own execution that protects it and its resources from external interference, tampering, or unauthorized disclosure. | T.TAMPER |
| 11 | O.SOF | The strength of function for the authentication mechanism must be SOF basic. | A.THREAT_LEVEL |

Table 8.6 shows that there are no unnecessary IT security objectives for the environment, since each objective addresses at least one secure usage assumption.

**Table 8.6 – All IT Security Objectives for the Environment Necessary**

| | Objective Name | Objective Description | Assumption |
|---|---|---|---|
| 1 | OE.CONFIGURATION | The TOE must be installed and configured properly. | A.CONFIGURATION |
| 2 | OE.CONFORMANCE | The responsible authority in each user organization must provide an acceptable level of security, including physical security such as simple power analysis, differential power analysis, external signals, extreme temperature, or physical tampering, for the given application and environment. | AE.CONFORMANCE AE.PHYSICAL_ PROTECTION |
| 3 | OE.OFF_CHIP | The TOE must be provided with correct, secure data for hardware key pair generation; user key pair generation; hashing; encryption of user key pairs, passwords, and key_mode byte as applicable for the chip Decode function; encryption of data for the chip Decrypt function | AE.OFF_CHIP |

## *8.2 SECURITY REQUIREMENTS RATIONALE*

### 8.2.1 All Objectives Met by Security Requirements

Tables 8.7 and 8.8 show how the IT security objectives are met.

**Table 8.7– Mapping of IT Security Objectives to Requirements**

| No | Objective Name | Security Requirement |
|---|---|---|
| 1 | O.AUTHENTICATION | FIA_UAU.1<br>FIA_UID.1 |
| 2 | O.ADMIN | FMT.MOF.1<br>FMT_MTD.1<br>FMT_SMR.1 |
| 3 | O.CRYPTO_KEY | FCS_CKM.4 |
| 4 | O.CRYPTO_OP | FCS_CKM.1<br>FCS_COP.1 |
| 5 | O.CONF_INTEGRITY | FDP_ACC.1<br>FDP_ACF.1<br>FDP_ITC.1<br>FMT_MSA.1<br>FMT_MSA.2<br>FMT_MSA.3<br>FMT_MTD.1<br>FPT_TDC.1 |
| 6 | O.DIG_SIG | FMT_MOF.1<br>FPT_RVM.1<br>FDP_ACC.1<br>FDP_ACF.1<br>ADV_SPM.1<br>AGD_USR.1. |
| 7 | O.ENFORCEPOLICY | ADV_SPM.1 |
| 8 | O.I&A | FIA_AFL.1<br>FIA_UAU.1<br>FIA_UID.1 |
| 9 | O.NONBYPASS | FMT_MSA.1<br>FMT_MSA.2<br>FMT_MSA.3<br>FPT_RVM.1 |
| 10 | O.SELF_PROTECT | FPT_SEP.1 |
| 11 | O.SOF | FIA_AFL.1 |

**Table 8.8– Mapping of IT Security Objectives for the Environment to Requirements**

| No | Objective Name | Security Requirement |
|---|---|---|
| 1 | OE.CONFIGURATION | FMT_MOF.1<br>FMT_MTD.1 |
| 2 | OE.CONFORMANCE | ADV_SPM.1<br>AVA_VLA.1 |
| 3 | OE.OFF_CHIP | FCS_CKM.1<br>FCS_COP.1 |

O.AUTHENTICATION is met by FIA_UAU.1, FIA_UAU.6, and FIA_UID.1. FIA_UAU.1 and FIA_UID.1 specify that functionality such as signing and decrypting with a key pair requires the entry of a password. These requirements allow the TOE to meet the objective of providing the capability of associating a password known to a user with a key pair or set of system capabilities. The association includes role and specific data such as key pairs.

O.ADMIN is met by the requirements FMT_MOF.1, FMT_MTD.1 and FMT_SMR.1. FMT_MOF.1 defines specific functionality used to configure and initialize the system that is available only to the administrator. FMT_MTD.1 defines the operations that may be performed by users and administrators. FMT_SMR.1 specifies that users be associated with roles, allowing the definition of an administrator and a user role, with specific rules. .

O.CRYPTO_KEY is met by FCS_CKM.4, which specifies methods for key destruction.

O.CRYPTO_OP is met by FCS_COP.1 and FCS_CKM.1. FCS_COP.1 requires that the cryptographic operations to be performed according to PKCS-1 for RSA and FCS_CKM.1 requires that hardware public key generation be performed according to PKCS-1 for RSA using a key size of 1024 bits.

O.CONF_INTEGRITY is met by FDP_ACC.1, FDP_ACF.1, FDP_ITC.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, FMT_MTD.1, and FPT_TDC.1. The confidentiality of key pairs and sensitive data is assured by access controls. FDP_ACC.1 and FDP_ADF.1 define the specific access controls. FDP_ITC.2 ensures that user data is imported in a controlled and secure manner with security attributes, thereby enhancing the integrity of the data to be stored in the TSF. FMT_MSA.1, FMT_MSA.2, and FMT_MSA.3 define controls on security attributes, including the user password and key_mode byte that are imported with each key. The attributes are securely importing and associated with the user key in chip memory. FMT_MTD.1 restricts access to certain data to the administrator role. FPT_TDC.1 defines CRC error coding for all data over 32 bits that is imported into the chip, produced by the chip, and loaded to the I/O buffer.

O.DIG_SIG is concerned with providing guidance and controls that allow correct and secure usage of the TOE capabilities. FMT_MOF.1, FPT_RVM.1, FDP_ACC.1, FDP_ACF.1, and ADV_SPM.1, an assurance requirement, meet this objective. FMT_MOF.1 defines the functionality available to the administrator, thereby providing controls on how operations and data can be used in the TOE. FPT_RVM.1 ensures that TSP enforcement functions are invoked and succeed before each function is allowed to proceed. FDP_ACC.1 and FDP_ADF.1 define the specific access controls that further limit and control the usage of the TOE capabilities, including subjects, objects, and operations. ADV_SPM.1 requires an informal security policy model that guides and controls the usage of the TOE. AGD_USR.1 specifies that a user guide be provided for correct use of digital signature capabilities and operations.

O.ENFORCEPOLICY is met by ADV_SPM.1, which requires the development and enforcement of an informal security policy model.

O.I&A is concerned with user authentication prior to granting access to assets and operations within the TOE. This objective is met by FIA_AFL.1, FIA_UAU.1, and FIA_UID.1. These three requirements define authentication failure handling and timing of authentication and identification. Together, they ensure that specified data and operations are password protected and that the TOE is not vulnerable to exhaustive password attacks.

O.NONBYPASS is met by FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FPT_RVM.1. FMT_MSA.1, 2, and 3 define security attribute associated with user key pairs that the define TSF operations that may be performed using that key pair. FPT_RVM.1 requires that TSP enforcement functions be invoked and succeed.

O.SELF_PROTECT is met by FPT_SEP.1. This requirement states that TSF must maintain a domain for its own execution that protects it and its resources from interference and tampering.

O.SOF states that the SOF for the authentication mechanism must be SOF basic. In addition to the TOE characteristic of 64 bit passwords, FIA_AFL.1 provides authentication failure handling mechanisms to decrease vulnerability of the authentication mechanism to attack.

OE.CONFIGURATION specifies that the TOE must be properly installed and configured. FMT_MOF.1 and FMT_MTD.1 provide for the capabilities to initialize and configure the system and limit that capability to the administrator role.

OE.CONFORMANCE specifies that organizations must provide an acceptable level of security. This acceptable level is defined by ADV_SPM.1 and possible threats that must be countered are described in response to the AVA_VLA.1 requirement for a vulnerability assessment.

OE.OFF_CHIP specifies that the hardware key and user keys must be securely generated off-chip (i.e., outside the TOE), and that hashing and encryption must be performed off chip to provide secure and correct inputs to the chip. These requirements are met by iterations of FCS_CKM.1 and FCS_COP.1 provided in section 5.2, Security Functional Requirements for the IT Environment.

### 8.2.2 All Functional Components Necessary

Tables 8.9 shows that each functional requirement is necessary, since it is used to address at least one of the IT security objectives. Discussion of the mapping is provided in the previous section.

**Table 8.9– Mapping of Functional Requirements to IT Security Objectives**

| | Component | Component Name | Objective |
|---|---|---|---|
| 1 | FCS_CKM.1 | Cryptographic key generation | O.CRYPTO_OP |
| 2 | FCS_CKM.4 | Cryptographic key destruction | O.CRYPTO_KEY |
| 3 | FCS_COP.1 | Cryptographic operation | O.CRYPTO_OP |
| 4 | FDP_ACC.1 | Subset access control | O.CONF_INTEGRITY<br>O.DIG_SIG |
| 5 | FDP_ACF.1 | Security attribute based access control | O.CONF_INTEGRITY<br>O.DIG_SIG |
| 6 | FDP_ITC.2 | Import of user data with security attributes | O.CONF_INTEGRITY |
| 7 | FIA_AFL.1 | Authentication failure handling | O.I&A<br>O.SOF |
| 8 | FIA_UAU.1 | Timing of authentication | O.AUTHENTICATION<br>O.I&A |
| 9 | FIA_UID.1 | Timing of identification | O.AUTHENTICATION<br>O.I&A |
| 10 | FMT_MOF.1 | Management of security functions behavior | O.ADMIN<br>O.DIG_SIG |
| 11 | FMT_MSA.1 | Management of security attributes | O.CONF_INTEGRITY<br>O.NONBYPASS |
| 12 | FMT_MSA.2 | Secure security attributes | O.CONF_INTEGRITY<br>O.NONBYPASS |
| 13 | FMT_MSA.3 | Static attribute initialisation | O.CONF_INTEGRITY<br>O.NONBYPASS |
| 14 | FMT_MTD.1 | Management of TSF Data | O.ADMIN<br>O.CONF_INTEGRITY |
| 15 | FMT_SMR.1 | Security roles | O.ADMIN |
| 16 | FPT_RVM.1 | Non-bypassability of the TSP | O.DIG_SIG<br>O.NONBYPASS |
| 17 | FPT_SEP.1 | TSF domain separation | O.SELF_PROTECT |
| 18 | FPT_TDC.1 | Inter_TSF basic TSF data consistency | O.CONF_INTEGRITY |

### 8.2.3 Satisfaction of Dependencies

Table 8.10 shows the dependencies between the functional requirements. Note that dependencies to assurance requirements show a reference of "Assurance" and that these requirements are included in the assurance requirements for the TOE.

**Table 8.10 – Functional Requirements Dependencies**

| No. | Component | Component Name | Dependencies | Reference |
|---|---|---|---|---|
| 1 | FCS_CKM.1 | Cryptographic key generation | FCS_COP.1<br>FCS_CKM.4<br>FMT_MSA.2 | 3<br>2<br>12 |
| 2 | FCS_CKM.4 | Cryptographic key destruction | FDP_ITC.1<br><br>FMT_MSA.2 | 6 & see section 8.2.3.1)<br><br>12 |
| 3 | FCS_COP.1 | Cryptographic operation | FDP_ITC.1<br><br>FCS_CKM.4<br>FMT_MSA.2 | 6 & see section 8.2.3.1<br><br>2<br>12 |
| 4 | FDP_ACC.1 | Subset access control | FDP_ACF.1 | 5 |
| 5 | FDP_ACF.1 | Security attribute based access control | FDP_ACC.1<br>FMT_MSA.3 | 4<br>13 |
| 6 | FDP_ITC.2 | Import of user data with security attributes | FDP_ACC.1<br>FTP_TRP.1<br>FPT_TDC.1 | 4<br>see section 8.2.3.2<br>18 |
| 7 | FIA_AFL.1 | Authentication failure handling | FIA_UAU.1 | 8 |
| 8 | FIA_UAU.1 | Timing of authentication | FIA_UID.1 | 9 |
| 9 | FIA_UID.1 | Timing of identification | None | N/A |
| 10 | FMT_MOF.1 | Management of security functions behavior | FMT_SMR.1 | 15 |
| 11 | FMT_MSA.1 | Management of security attributes | FDP_ACC.1<br>FMT_SMR.1 | 4<br>15 |
| 12 | FMT_MSA.2 | Secure security attributes | FDP_ACC.1<br>FMT_MSA.1<br>FMT_SMR.1 | 4<br>11<br>15 |
| 13 | FMT_MSA.3 | Static attribute initialisation | FMT_MSA.1<br>FMT_SMR.1 | 11<br>15 |
| 14 | FMT_MTD.1 | Management of TSF Data | FMT_SMR.1 | 15 |
| 15 | FMT_SMR.1 | Security roles | FIA_UID.1 | 9 |
| 16 | FPT_RVM.1 | Non-bypassability of the TSP | None | N/A |
| 17 | FPT_SEP.1 | TSF domain separation | None | N/A |
| 18 | FPT_TDC.1 | Inter_TSF basic TSF data consistency | None | N/A |

Certain dependencies are not met as shown in Table 8.9. The justification for not including these dependencies is provided in the following subsections.

#### 8.2.3.1 Dependencies on FDP_ITC.1

User keys are imported into the TOE with security attributes using the Decode_key command. Security attributes are the key_mode byte. Since FDP_ITC.1 specifies import user data without security attributes, it was not applicable to the TOE. FDP_ITC.2, Import of user data with security attributes and is a substitute for the dependencies for FDP_ITC.1 in FCS_CKM.4 and FCS_COP.1.

#### 8.2.3.2 Dependencies on FTP_TRP.1

FTP_TRP.1 is a dependency of FDP_ITC.2. FTP_TRP.1 was deemed unnecessary because the TOE only executes the TSF and does not execute any unevaluated software. Thus, trusted path is not required since there is no masquerading threat. All inputs go to the TSF and stay there (there is no untrusted software to which input may be passed). Similarly, all output is generated by the TSF and goes directly to the output interface.

### 8.2.4 Strength of Function Rationale

The threat level for the TOE authentication function is assumed to be SOF-basic. The required hardware and FAILRSTPSW passwords are 64 bits each. A user password may also optionally be used. The Administrator Guide, User Guide, and Security Policy documents define a password policy that advises the user on password selection and change frequency. SOF analysis is described in Section 6. The authentication failure mechanisms provide significant protection against password cracking.

### 8.2.5 Assurance Rationale

The assurance level selected for the TOE was EAL3 because it provides appropriate assurance measures for the expected application of the product. EAL3 ensures sound development practices and requires a moderate level of independently assured security. In the PC industry, this level of security is required for digital signature applications that are expected to use the TOE.

ADV_SPM.1 is an augmentation of EAL3 requirements. It was included because it is a dependency of FMT_MSA.2, which in turn is a dependency of FCS_CKM.4 and FCS_COP.1. The TOE Security Policy is defined in the IBM Secure Signature Generation Chip Security Policy.

## *8.3 TOE SUMMARY SPECIFICATION RATIONALE*

### 8.3.1 All TOE Security Functional Requirements Satisfied

Table 8.11 shows that the IT Security Functions in the TOE Summary Specification (TSS) address all of the TOE Security Functional Requirements. The mappings are discussed in detail in Section 6. As described in Section 6, all functional components map to a TOE Security Function. TOE Security functions are:

- Cryptographic Operation (CO), including:
  - Generation of the hardware public key (CO-1)
  - RSA Digital Signature (CO-2)
  - Data Decryption (CO-3)
  - Key Decryption (CO-4)
  - Key Destruction (CO-5)
- Access Control (AC)
- Authentication (AU), including:

  - Authentication failure (AU-1)
  - Timing of authentication (AU-2)
  - Timing of identification (AU-3)
- Security Management (SM)
  - Management of TOE Functions and Data (SM-1)
  - Roles (SM-2)
  - Security Attributes (SM-3)
- System Architecture (SA)
  - Interfaces (SA-1)
  - Cyclic Redundancy Check (SA-2)

**Table 8.11 – Mapping of Functional Requirements to TOE Summary Specification**

| No | Functional Component | Functional Requirement | TOE Security Function |
|---|---|---|---|
| 1 | FCS_CKM.1 | Cryptographic key generation | CO-1 |
| 2 | FCS_CKM.4 | Cryptographic key destruction | CO-5 |
| 3 | FCS_COP.1 | Cryptographic operation | CO-2<br>CO-3<br>CO-4 |
| 4 | FDP_ACC.1 | Subset access control | AC |
| 5 | FDP_ACF.1 | Security attribute based access control | AC |
| 6 | FDP_ITC.2 | Import of user data with security attributes | AC |
| 7 | FIA_AFL.1 | Authentication failure handling | AU-1 |
| 8 | FIA_UAU.1 | Timing of authentication | AU-2 |
| 9 | FIA_UID.1 | Timing of identification | AU-3 |
| 10 | FMT_MOF.1 | Management of security functions behavior | SM-1 |
| 11 | FMT_MSA.1 | Management of security attributes | SM-3 |
| 12 | FMT_MSA.2 | Secure security attributes | SM-3 |
| 13 | FMT_MSA.3 | Static attribute initialisation | SM-3 |
| 14 | FMT_MTD.1 | Management of TSF Data | SM-2 |
| 15 | FMT_SMR.1 | Security roles | SM-2 |
| 16 | FPT_RVM.1 | Non-bypassability of the TSP | SA-1 |
| 17 | FPT_SEP.1 | TSF domain separation | SA-1 |
| 18 | FPT_TDC.1 | Inter-TSF basic TSF data consistency | SA-2 |

### 8.3.2 All TOE Summary Specification (TSS) Functions Necessary

Table 8.12 shows that all of the IT Security Functions in the TOE Summary Specification (TSS) help meet TOE Security Functional Requirements. Discussion of this mapping is provided in the subsection above.

**Table 8.12 – Mapping of TOE Summary Specification to Functional Requirements**

| TOE Security Function | Requirement | Requirement Name |
|---|---|---|
| Cryptographic Operation<br>CO-1, CO-2, CO-3, CO-4, CO-5 | FCS_CKM.1 | Cryptographic key generation |
| | FCS_CKM.4 | Cryptographic key destruction |
| | FCS_COP.1 | Cryptographic operation |
| Access Control | FDP_ACC.1 | Subset access control |
| | FDP_ACF.1 | Security attribute based access control |
| | FDP_ITC.1 | Import of user data without security attributes |
| Authentication<br>AU-1, AU-2, AU-3 | FIA_AFL.1 | Authentication failure handling |
| | FIA_UAU.1 | Timing of authentication |
| | FIA_UID.1 | Timing of identification |
| Security Management<br>SM-1, SM-2, SM-3 | FMT_MOF.1 | Management of security functions behavior |
| | FMT_MSA.1 | Management of security attributes |
| | FMT_MSA.2 | Secure security attributes |
| | FMT_MSA.3 | Static attribute initialisation |
| | FMT_MTD.1 | Management of TSF Data |
| | FMT_SMR.1 | Security roles |
| System Architecture<br>SA-1, SA-2 | FPT_RVM.1 | Non-bypassability of the TSP |
| | FPT_SEP.1 | TSF domain separation |
| | FPT_TDC.1 | Inter-TSF basic TSF data consistency |

### 8.3.3 Assurance Measures Rationale

The assurance measures rationale shows how all assurance requirements were satisfied. This information is provided in Table 6.6 and is not repeated here.

## *8.4 PP CLAIMS RATIONALE*

Not applicable.

# 9 ACRONYMS

| | |
|---|---|
| **CC** | Common Criteria for IT Security Evaluation |
| **CM** | Configuration Management |
| **EAL** | Evaluation Assurance Level |
| **RNG** | Random Number Generator |
| **SF** | Security Function |
| **SFP** | Security Function Policy |
| **ST** | Security Target |
| **TOE** | Target of Evaluation |
| **TSC** | TSF Scope of Control |
| **TSF** | TOE Security Functions |
| **TSP** | TOE Security Policy |

# 10 REFERENCES

### IBM/Atmel Documentation

Atmel Secure Signature Generation Chip AT90SP0801

### Standards

CCITSE *Common Criteria for Information Technology Security Evaluation*,