Trusted Virtual Machine

TVM Registers

Register	***Type***	***Operand Reference***
KAX	*General Purpose*	*0x0A*
KBX	*General Purpose*	*0x0B*
KCX	*General Purpose*	*0x0C*
KDX	*General Purpose*	*0x0D*
KPC	*Program Counter*	*0x0E*
KRX	*32B Byte Array*	*0x0F*
KSP	*Stack Pointer*	*0x10*

General Instruction Set Reference

Move (Opcode: 0x88)

MOV $DST $SRC

Moves the contents of the *$SRC* register into the *$DST* register.

Move Immediate (Opcode: 0x89)

MOVI $DST #VAL

Moves the 64-bit immediate *#VAL* into the *$DST* register.

Move KRX (Opcode: 0x91)

MOVK [IMM 32 bytes]

Moves the immediate 32 bytes into the *$KRX* register.

Read KRX (Opcode: 0x92)

RDK, #VAL (1 byte)

#VAL <= 32

Reads up to *#VAL* bytes from stdin into the *$KRX* register.

Dump State (Opcode: 0xDD)

DST

Outputs the current context of the TVM.

Halt (Opcode: 0xFE)

HLT

Outputs the final context of the TVM and halts execution.

Add (Opcode: 0xD3)

ADD $DST $SRC

Adds the contents of the *$SRC* register to the contents of the *$DST* register storing the result in *$DST*.

Add Immediate (Opcode: 0xC6)

ADDI $DST #VAL

Adds the 64-bit immediate *#VAL* to the contents of the *$DST* register storing the result in *$DST*.

Subtract (Opcode: 0xD8)

SUB $DST $SRC

Subtracts the contents of the *$SRC* register from the contents of the *$DST* register storing the result in *$DST*.

Subtract Immediate (Opcode: 0xEF)

SUBI $DST #VAL

Subtracts the 64-bit immediate *#VAL* from the contents of the *$DST* register storing the result in *$DST*.

Multiply (Opcode: 0x34)

MUL $DST $SRC

Multiplies the contents of the *$SRC* register with the contents of the *$DST* register storing the result in *$DST*.

Divide (Opcode: 0xB9)

DIV

Divides the contents of the *$KBX* register with the contents of the *$KCX* register storing the result in *$KAX* and remainder in *$KDX*.

XOR (Opcode: 0xB7)

```
XOR  $DST $SRC
```

XORs the contents of the *$SRC* register with the contents of the *$DST* register storing the result in *$DST*.

Push (Opcode: 0xED)

```
PUSH $SRC
```

Push the contents of the *$SRC* register onto the stack pointed to by *$KSP*.
If the *$SRC* register is *$KRX*, the entire array is pushed onto the stack, 8 bytes at a time, starting with the first 8 bytes.

Pop (Opcode: 0xB1)

```
POP  $DST
```

Pop a 64-bit value off of the stack and store in *$DST*.
If the *$SRC* register is *$KRX*, 4 values are popped off the stack to fill the array, with the first value filling in the last 8 bytes.

Conditional Instruction Set Reference

Compare (Opcode: 0xCC)

```
CMP $REG1 $REG2
```

Compares the contents of $REG1 and $REG2 and updates the internal *$KFLAGS* register *ZeroFlag* and *SignedFlag* bits.

Jump (Opcode: 0x96)

```
JMP #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL*.

Jump if Not Equal (Opcode: 0x9E)

```
JNE #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL* if the *ZeroFlag* is set.

Jump if Greater Than (Opcode: 0x2F)

```
JG #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL* if the *ZeroFlag* and *SignedFlag* are both *zero*.

Jump if Greater Than or Equal (Opcode: 0xF4)

```
JGE #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL* if the *SignedFlag* is *zero*.

Jump if Less Than (Opcode: 0x69)

```
JL #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL* if the *SignedFlag* is set.

Jump if Less Than or Equal (Opcode: 0x5F)

```
JLE #VAL
```

Performs a relative jump using the signed 16-bit immediate *#VAL* if the *ZeroFlag* and *SignedFlag* are set.

Authentication Instruction Set Reference

Authentication Challenge Begin (Opcode: 0xC4)

ACB

Initiates an authentication challenge. The challenge value is placed in *$KAX*

Authentication Challenge Response (Opcode: 0xC5)

ACR

Attempts to authenticate using the challenge response in $KRX. The response should be formatted as follows:

```
+---------------+-------------------------------+----------------+
| IV (12 bytes) | Response Ciphertext (8 bytes) | Tag (12 bytes) |
+---------------+-------------------------------+----------------+
```

A return code will be placed into *$KAX* after execution. The possible codes are:

`0x0` - Success
`0x1` - Decryption did not match challenge
`0x2` - Tag did not match
`0x3` - No challenge given

Cryptographic Instruction Set Reference

AES-GCM Encrypt (Opcode: 0x9B)

AGE $SRC

Encrypts 32 bytes of data pointed to by *$SRC* and places it into *$KRX*.
The first 12 bytes of *$KRX* are used as the IV.
The resulting authentication tag is left-padded to 128 bits with 0's, then pushed onto the stack similar to:
PUSH $KRX

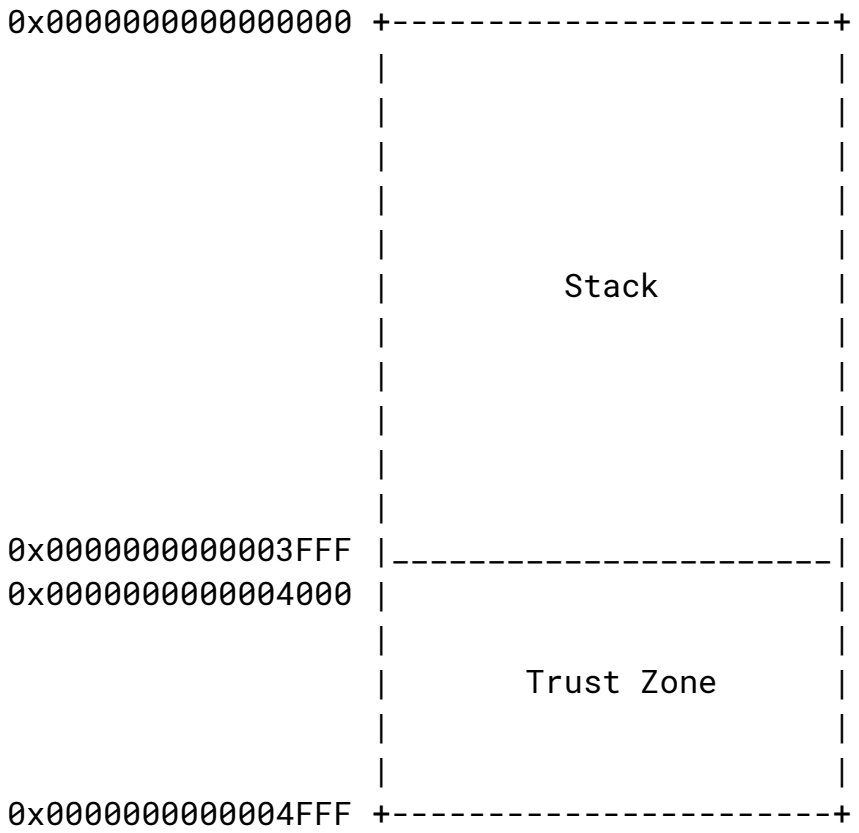
AES-GCM Decrypt (Opcode: 0x7F)

AGD

Decrypts 32 bytes of data loaded *$KRX* leaving the data in *$KRX*.
Uses the IV from the previous execution of AGE.

TVM Memory Layout

The TVM has the following memory layout, containing all valid addresses:



The stack is accessible by any user, but the trust zone can only be accessed after successfully authenticating with the TVM.