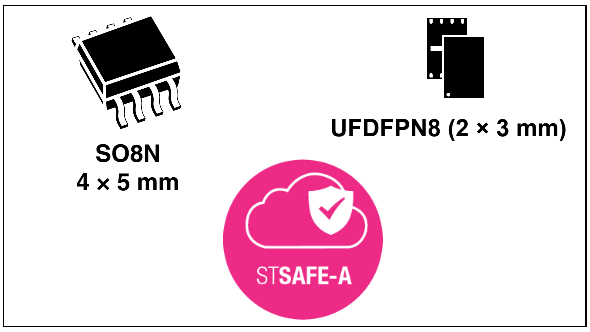
STSAFE-A100

Authentication, state-of-the-art security for peripherals and IoT devices

Datasheet - production data



Features

- Authentication (of peripherals, IoT and USB Type-C devices)
- Secure channel establishment with remote host including transport layer security (TLS) handshake
- Signature verification service (secure boot and firmware upgrade)
- Usage monitoring with secure counters
- Pairing and secure channel with host application processor
- Wrapping and unwrapping of local or remote host envelopes
- On-chip key pair generation

Security features

- Latest generation of highly secure MCUs
  - CC EAL5+ AVA_VAN5 Common Criteria certified
  - Active shield
  - Monitoring of environmental parameters
  - Protection mechanism against faults
  - Unique serial number on each die
  - Protection against side-channel attacks
- Advanced asymmetric cryptography
  - Elliptic curve cryptography (ECC) with NIST or Brainpool 256-bit and 384-bit curves
  - Elliptic curve digital signature algorithm (ECDSA) with SHA-256 and SHA-384 for digital signature generation and verification
  - Elliptic curve Diffie-Hellman (ECDH) for key establishment
- Advanced symmetric cryptography
  - Key wrapping and unwrapping using AES-128/AES-256
  - Secure channel protocols using AES-128
- Secure operating system
  - Secure STSAFE-A100 kernel for authentication and data management
  - Protection against logical and physical attacks

Hardware features

- Highly secure MCU platform
- 6 Kbytes of configurable non-volatile memory
  - Highly reliable CMOS EEPROM technology
  - 30 years' data retention at 25 °C
  - 500 000 erase / program cycles endurance at 25 °C
  - 1.62 V to 5.5 V continuous supply voltage
- Operating temperature: −40 to 105 °C

Protocol

- I²C-bus slave interface
  - Up to 400 kbps transmission speed (Fast mode) and true open-drain pads
  - 7-bit addressing

Packages

- ECOPACK®-compliant SO8N 8-lead plastic small outline and UFDFPN 8-lead ultra thin profile fine pitch dual flat packages

This is information on a product in full production.

Contents

1 Description

The STSAFE-A100 is a highly secure solution that acts as a secure element providing authentication and data management services to a local or remote host. It consists of a full turnkey solution with a secure operating system running on the latest generation of secure microcontrollers.

The STSAFE-A100 can be integrated in IoT (Internet of things) devices, smart-home, smart-city and industrial applications, consumer electronics devices, consumables and accessories.

1.1 Key function overview

Figure 1. Authentication to a remote server (IoT device case)

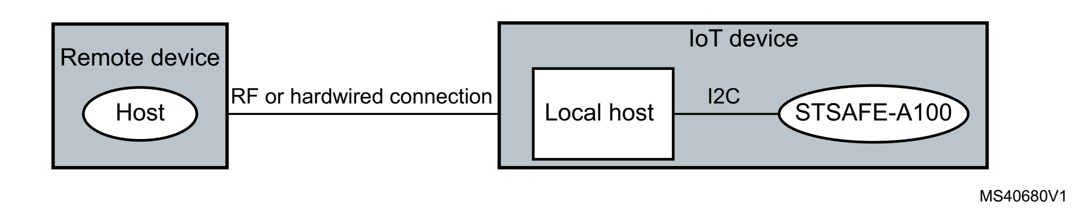
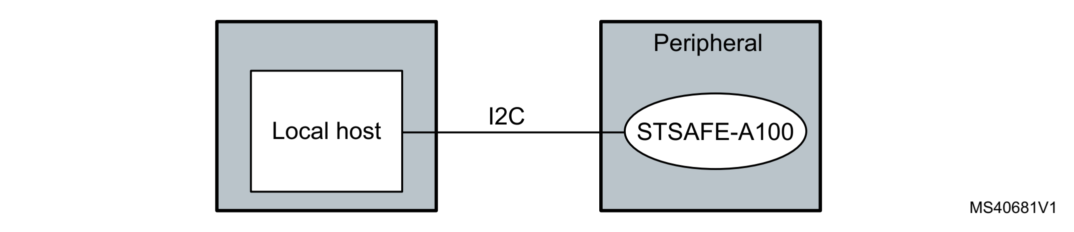


Figure 2. Authentication to a local host (accessory or consumable case)



The STSAFE-A100 can be mounted on:

- a device that authenticates to a remote host (IoT device case), the local host being used as a pass-through to the remote server.
- a peripheral that authenticates to a local host, for example games, mobile accessories or consumables.

The STSAFE-A100 secure element supports the following features:

- Authentication

  The STSAFE-A100's authentication service provides proof to a remote or local host that a certain peripheral or IoT is legitimate. An equipment manufacturer can thus ensure that only authentic peripherals like accessories or consumables can be used in conjunction with the original equipment. In the same way, a service provider can make sure that its service is only provided to the appropriate IoT device.

  The authentication service utilizes the ECC cryptographic scheme with NIST or Brainpool 256-bit and 384-bit curves. It also uses the widely deployed ECDSA signature scheme with SHA-256 and SHA-384 for generating digital signatures. In addition, it is compatible with the USB Type-C authentication scheme.

- Secure-channel key establishment (TLS)

  The STSAFE-A100 helps encrypt communications between a device and a remote host (such as a cloud server or gateway). The key establishment service uses the ECC cryptographic scheme with NIST, or Brainpool 256-bit and 384-bit curves. Moreover, it computes the shared secret with the widely recognized Diffie-Hellman schemes ECDH and ECDHE.
- Signature verification

  The STSAFE-A100 can verify an ECDSA signature by using a public key provided by the local host. This mechanism can offload a local application processor with limited computing power and no elliptic curve cryptography accelerator. It is typically used for the secure boot or secure firmware update of the local host.
- Host authentication

  With its public key slot, the STSAFE-A100 can authenticate a local or remote host. Successful authentication by the STSAFE-A100 grants the local or remote host access to some authorized commands or memory partitions.
- Secure one-way counters (peripheral usage monitoring)

  The manufacturer can limit the usage of disposable accessories or consumables to a given value by presetting the secure one-way counters. These counters can only be decremented.
- Memory partitioning

  The STSAFE-A100 comes with 6 Kbytes of non-volatile memory split into areas, whose read and write access rights can be configured to free access, local host access or remote host access.
- Pairing and secure channel with the host

  The STSAFE-A100 allows a secure channel to be set up with the local host based on AES-128-bit keys for command authorization, command data encryption, response data encryption and response authentication. Typically, this secure channel prevents eavesdropping of sensitive information on the I²C line.
- Wrapping & unwrapping local or remote host envelopes

  The STSAFE-A100 can be used to encrypt or decrypt data between the remote host and the local host. The local host may also use the STSAFE-A100’s encryption/decryption services to store sensitive data to a local, external storage like Flash memory.

1.2 STSAFE-A100’s environment

The STSAFE-A100 comes with a host library that can be ported to a wide range of general-purpose microcontrollers or microprocessors. This library includes a command wrapper as well as generic use cases.

STMicroelectronics also offers key provisioning services for storage of customer credentials in a secure, certified environment.

LICENSED DPA COUNTERMEASURES™

1.3 Pin descriptions

Figure 3. SO8N pinout - Top view

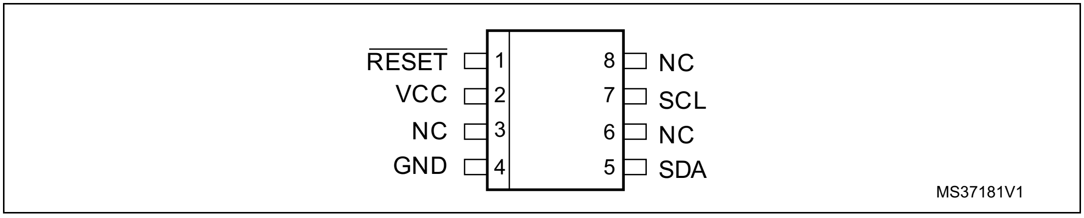


Figure 4. UFDFPN8 pinout - Top view

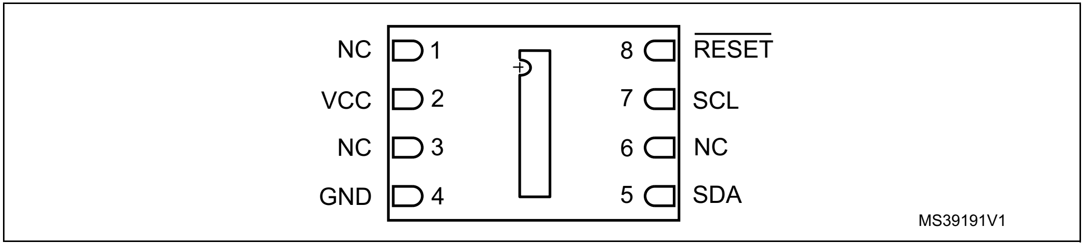


Table 1 provides the name and description of the four contacts on the STSAFE-A100 device. Details on each are provided later in this text.

Table 1. Signal descriptions

Signal	Name	Description
V_{CC}	Supply voltage	The 1.62 to 5.5 V supply voltage is supported for powering all internal STSAFE-A100 functions.
GND	Supply and signals ground	Ground reference pin for power and all I/O signals.
$\overline{\text{RESET}}$	Reset	This input signal is used to reset STSAFE-A100. The $\overline{\text{RESET}}$ pin is pull-down by default meaning that the device is reset if connected to ground or if the pin is floating. The device is active if the $\overline{\text{RESET}}$ pin is tied high.

Table 1. Signal descriptions (continued)

Signal	Name	Description
SCL	Serial clock	This input signal is used to strobe all data in and out of STSAFE-A100. The signal is an input signal only and does not support the clock stretching mode common to generic I^2C. The Clock signal is driven by the I^2C master.
SDA	Serial data	This I/O signal is used to transfer data into and out of STSAFE-A100. The signal uses an open drain output configuration. An external pull-up resistor is used to “pull up” the output.
NC	-	Not connected internally

2 Asymmetric cryptography use cases

This chapter illustrates the many uses of an STSAFE-A100 device using asymmetric cryptography.

2.1 Authentication

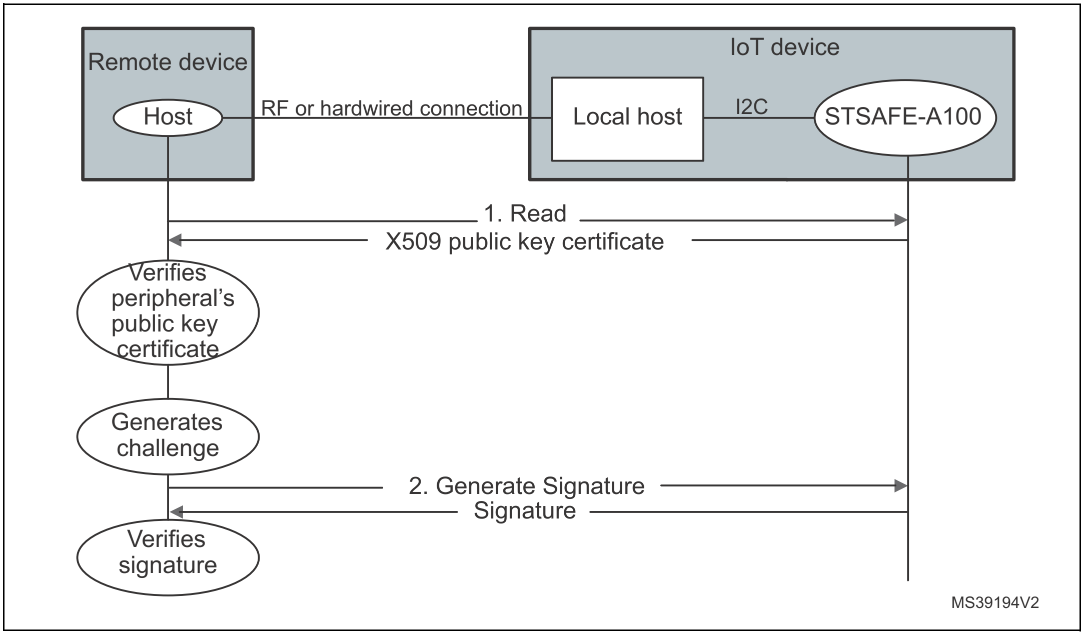
This scenario illustrates the command flow where the STSAFE-A100 is mounted on a device that authenticates to a remote host (IoT device case), the local host being used as a pass-through to the remote server.

The scenario where the STSAFE-A100 is mounted on a peripheral that authenticates to a local host, for example games, mobile accessories or consumables, is exactly the same.

Command flow (see *Figure 5*)

1. Obtain the public key of the STSAFE-A100 chip in the host device:
   - Command 1 is used to read the X509 public key certificate from the data partition of the STSAFE-A100 chip.
   - The host device verifies the X509 public key certificate with the CA public key (the host is responsible for getting a copy of this key). When the verification process succeeds, the host device has an authentic copy of the STSAFE-A100 public key that it will use later on for verification of the signature.
2. The host device generates a challenge and stores it for later use in the verification of the signature. The host device then computes a hash of this challenge and sends it to the STSAFE-A100 in Command 2 in order to fetch the signature that the STSAFE-A100 chip computed with its private key. The host device verifies the signature with the STSAFE-A100 public key (obtained in the first step of this scenario). When valid, the host knows that the peripheral or IoT is authentic.

Figure 5. Example of peripheral or IoT device authentication



2.2 Authentication or data management within signature session

The STSAFE-A100 chip is mounted on an IoT device and has been personalized in the manufacturing environment with a counter and/or a data partition. In order to confirm the data update or counter decrement, the STSAFE-A100 is requested to send a signature over the execution of these operations. The host can then verify the signature using the STSAFE-A100’s public key.

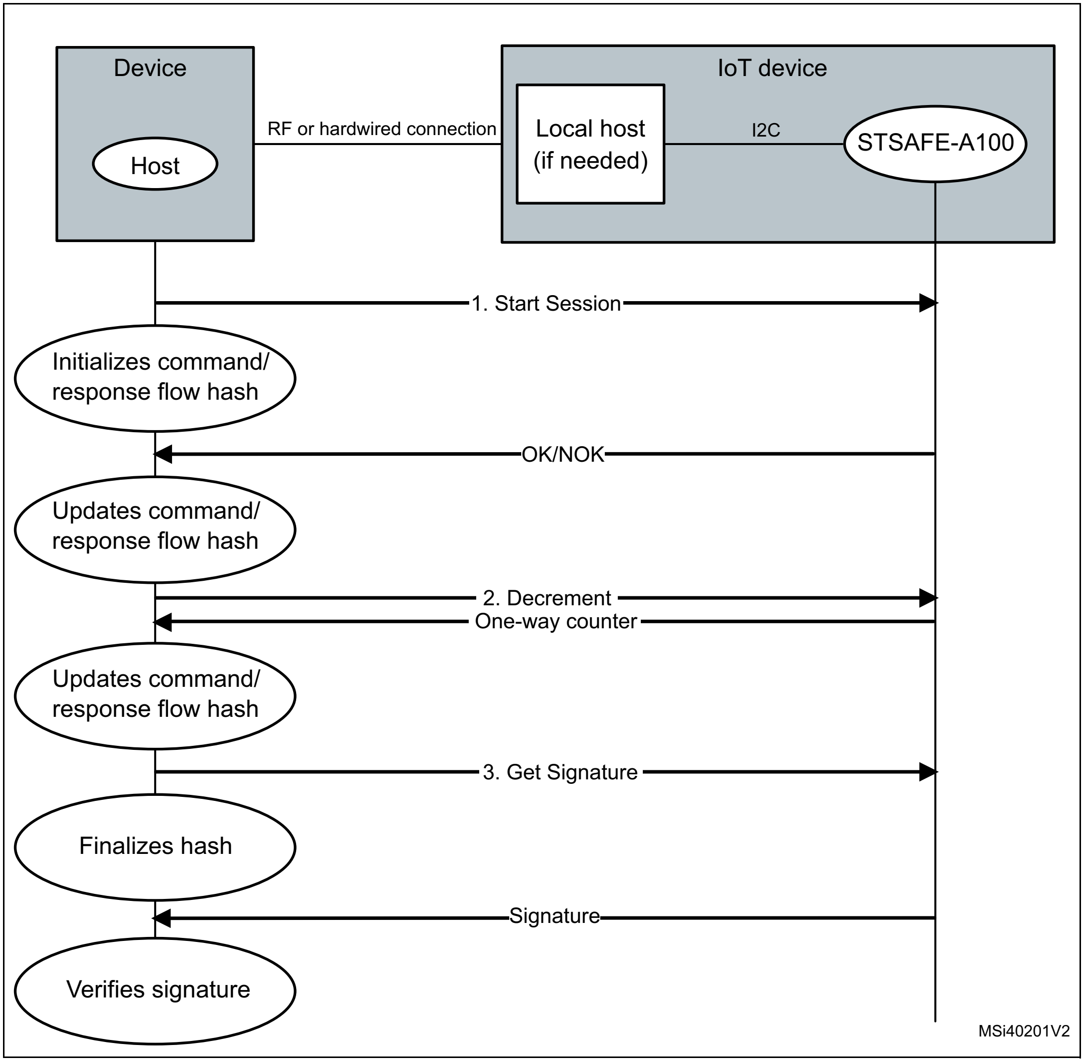
Command flow (see *Figure 6*)

This use case assumes that the host device already has an authentic copy of the STSAFE-A100 public key. If this is not the case, the host device must first fetch the X509 public key certificate from the STSAFE-A100 chip and verify it with its CA public key (see *Section 2.1*).

1. The host starts a signature session in the STSAFE-A100 chip. This instructs the STSAFE-A100 chip to compute a SHA-256 or SHA-384 on the subsequent commands and responses. The host has to do the same on its side.
2. The host requests the STSAFE-A100 to decrease the one-way counter by a certain amount or, reads from or updates a data partition area.
3. The host requests the STSAFE-A100 device to finalize hash computation over the commands/responses issued since reception of the Start Session command, and to sign it. This command can optionally come with a challenge.
4. On its side, the host has already finalized the same hash computation and can verify the received signature using the STSAFE-A100’s previously acquired certificate. When the signature is valid, the host has a proof that the counter was decremented.

Removing command 2 from the command flow constitutes an alternative solution for the IoT device authentication use case.

Figure 6. Authentication or data management within signature session use case



1. This scenario assumes that the X509 certificate has already been exchange between the host and the STSAFE-A100.

2.3 Key Establishment

The STSAFE-A100 is mounted on a device (for example an Internet-of-things device), which communicates with a remote server and needs to establish a secure channel to exchange data with it.

The goal of this use case is to share a secret between the local host and the remote server using the elliptic curve Diffie-Hellman (ECDH) scheme with a static key in the STSAFE-A100. The STSAFE-A100 also supports ECDHE that uses an ephemeral key but this has not been illustrated.

The shared secret should further be derived to one or more working keys, but this is not illustrated here. The working keys can then be used in communication protocols like TLS for example for protecting the confidentiality, integrity and authenticity of the data that are exchanged between the local host and the remote server. Below are some examples of data:

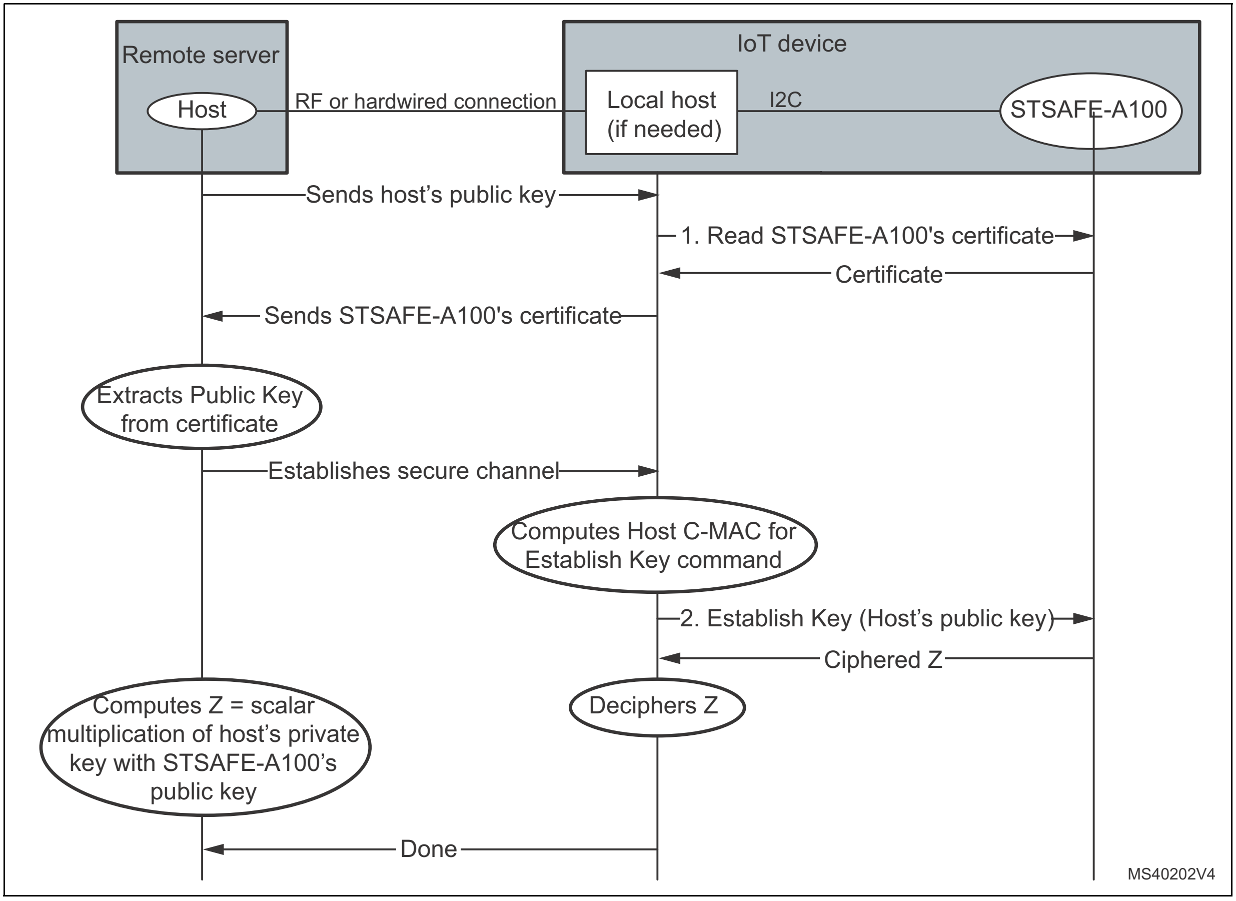
- From the local host to the remote server: power consumption of a smart meter, alarm of a fire sensor or blood pressure data of a health sensor.
- From the remote server to the local host: activating the recharge of the battery of an electric car, activating home appliances like air conditioning or water heaters, or pushing firmware upgrades to IoT devices.

Because the Establish Key command needs to be MACed and its answer is encrypted to avoid eavesdropping on the shared secret, the scenario assumes that the local host has set up a host C-MAC and cipher keys as described in *Section 3.1: Host secure channel setup use case*. It is also assumed that the local host knows the host C-MAC sequence counter; if not, it can send a Query command to the STSAFE-A100.

Command flow (see *Figure 7*)

1. The remote host server sends its certificate to the local host. The local host extracts the public key and can optionally verify the validity of the certificate. In its response, the local host sends the STSAFE-A100's certificate.
2. The remote server verifies the STSAFE-A100's X.509 public key certificate with the CA public key (the host is responsible for getting this key). When the verification succeeds, the remote server has an authentic copy of the STSAFE-A100's public key.
3. The remote server then computes a shared secret (Z) by doing a scalar multiplication of the Host's Private key with the STSAFE-A100's public key.
4. The remote server requests the local host to establish a secure connection.
5. The local host computes the Host's C-MAC for the Establish Key command
6. The local host sends the STSAFE-A100 an Establish Key command providing the remote host's public key appended with the previously computed host's C-MAC. The STSAFE-A100 does the same operation as the remote host server, and performs the scalar multiplication of its private key with the remote server's public key to compute the shared secret (Z). It then encrypts the response using the Host's cipher key.
7. The local host reads the STSAFE-A100's answer and deciphers the shared secret (Z) with the locally stored host's cipher key.
8. The remote host server and the local host have a shared secret Z.

Figure 7. Key Establishment command flow


Remote server
Host
RF or hardwired connection
IoT device
Local host (if needed)
I2C
STSAFE-A100
Sends host's public key
1. Read STSAFE-A100's certificate
Certificate
Sends STSAFE-A100's certificate
Extracts Public Key from certificate
Establishes secure channel
Computes Host C-MAC for Establish Key command
2. Establish Key (Host's public key)
Ciphered Z
Deciphers Z
Computes Z = scalar multiplication of host's private key with STSAFE-A100's public key
Done
MS40202V4

2.4 TLS Handshake protocol use case

This use case shows how the STSAFE-A100 can be used by a local host for implementing the transport layer security (TLS) protocol version 1.2. that is specified in RFC 5246.

All details of the TLS handshake protocol are explained in RFC 5246. The intention of this section is not to re-explain all details or all TLS-specific terminology; the main focus is on the interaction of the local host with the STSAFE-A100 while briefly illustrating how it fits in the overall TLS protocol flow.

A local host can use the STSAFE-A100 on the TLS client and TLS server side for implementing the following cryptographic mechanisms of the TLS handshake protocol:

- Secure random generation. It is useful in the Client Hello and Server Hello messages.
- Signature generation with ECDSA using a private key that is securely stored in the STSAFE-A100. It is useful for generating the signatures in the Server Key Exchange message and Certificate Verify message. The local host is responsible for generating the SHA-256 or SHA-384 message digest and for sending the digest to the STSAFE-A100 in the command data.
- Signature verification with ECDSA and a public key that is sent by the local host to the STSAFE-A100. It is useful for verifying the peer's certificate chain in the Certificate message. It is also useful for verifying the signatures in the Server Key Exchange message and Certificate Verify message. The local host is responsible for generating

the SHA-256 or SHA-384 message digest and for sending the digest to the STSAFE-A100 together with the public key and a reference to the curve that must be used.

- Ephemeral key pair generation in the STSAFE-A100. It is useful when ECDHE has been chosen as the key exchange algorithm. the STSAFE-A100 stores the private key and returns the public key to the local host for inclusion in the Server Key Exchange and Client Key Exchange messages.
- ECDH or ECDHE with a static (ECDH) or an ephemeral (ECDHE) private key in the STSAFE-A100. The local host must send the peer's public key from the Server Key Exchange or Client Key Exchange message to the STSAFE-A100, which returns the shared secret that is encrypted with the Host’s Cipher Key. After decryption with the Host’s Cipher Key, the local host can use the shared secret as the pre-master secret of the TLS handshake protocol.

The STSAFE-A100 does not implement the following cryptographic mechanisms of the TLS handshake protocol:

- conversion of the pre-master secret into the master secret
- generation of the verify data in the Finished message
- expansion of the master secret into a key block that may include a client write MAC key, a server write MAC key, a client write encryption key, a server write encryption key and two initial values
- MACing, encryption and decryption of application data with keys from the key block

The command flow illustrates the integration of the STSAFE-A100 on the TLS client’s side using ECDSA as the signature algorithm and ECDHE as the key exchange algorithm. The STSAFE-A100 can also be integrated on the TLS server’s side but this is not illustrated here.

Command flow (see *Figure 8*)

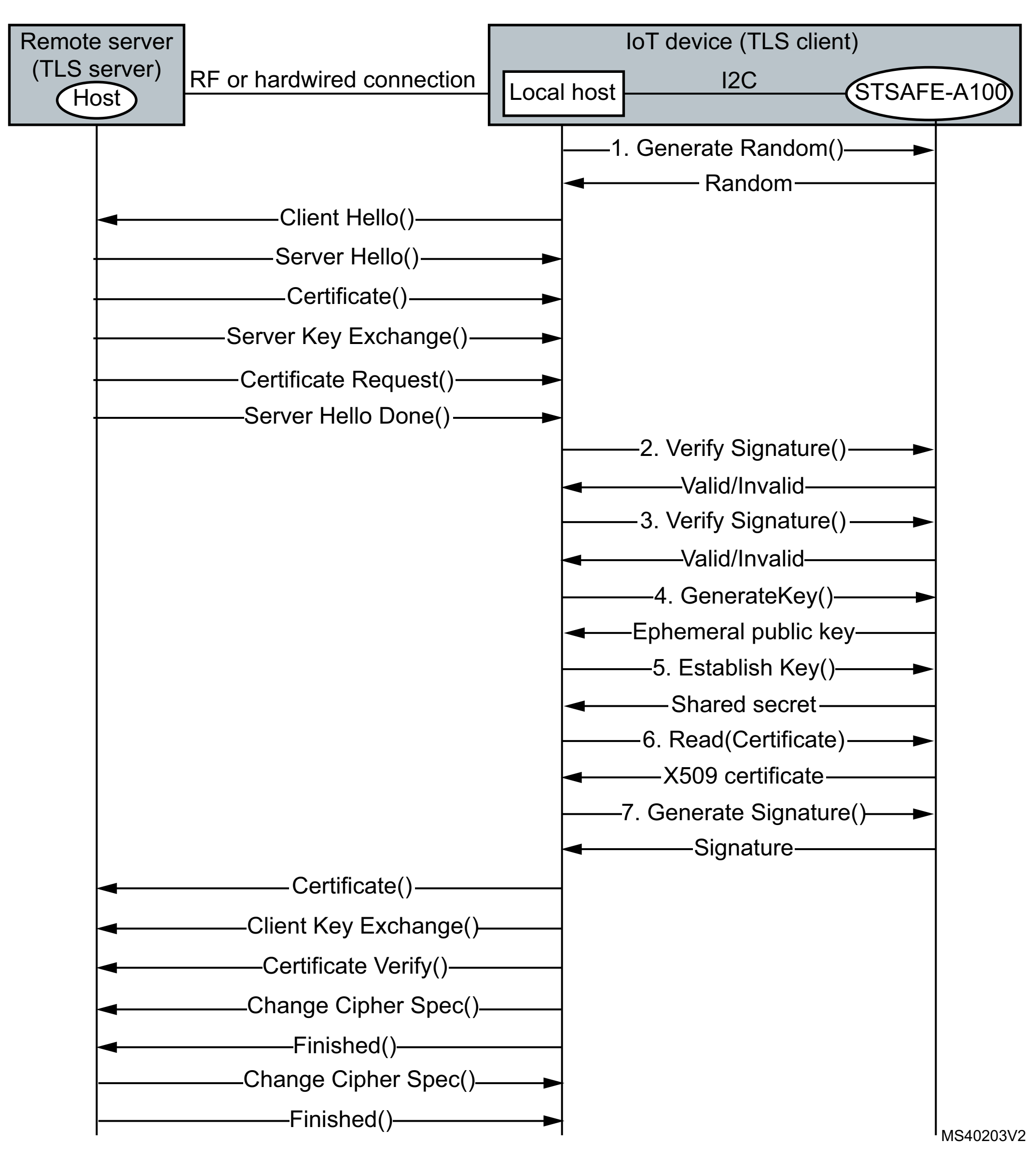
1. The TLS client sends the Client Hello message including the client version, a random that can be obtained from the STSAFE-A100 with the Generate Random command (1), a session ID, the list of supported cipher suites and compression methods and an extension that lists the supported signatures and hash algorithms.
2. The TLS server sends the Server Hello message including the protocol version, a random, a session ID and the chosen cipher suite and compression method. The TLS server also sends the Certificate message including the X509 certificate chain of the TLS server. Upon reception of this message, the local host of the TLS client may use the STSAFE-A100 for verifying the certificate chain. Therefore, the local host must parse every certificate from the chain, hash the *To Be Signed* (TBS) data and send the Verify Signature command (2) to the STSAFE-A100. In the Verify Signature command data, the local host must include a reference to the curve that must be used, the public key, the signature and the hash. The STSAFE-A100 responds with an indication of whether the verification was successful or not. When the certificate chain is composed of more than one certificate, the Verify Signature command must be sent as many times as there are certificates in the chain (this is not illustrated in *Figure 8*).
3. The TLS server sends the Server Key Exchange message including the Diffie-Hellman public key of the TLS server and a signature over the server key exchange parameters. Upon reception of this message, the local host of the TLS client may use the STSAFE-A100 for verifying the signature with the Verify Signature command (3) and the same mechanisms that were applied in step 2.

4. When the signature is valid, the local host may use the Generate Key command 4 of the STSAFE-A100 for generating an ephemeral key pair in the STSAFE-A100. The command data take a reference to the curve that must be used, and the response data includes the public key of the freshly generated key pair. The Generate Key command (4) requires a Host C-MAC in the command but this is not illustrated here (see *Section 3.1*).
5. The local host can now use the Establish Key command (5) of the STSAFE-A100 and give the public key of the TLS server in the command data. The response data contains the shared secret that is computed with ECDHE using the ephemeral private key in the STSAFE-A100 and the public key of the TLS server. The shared secret in the response data is encrypted with the Host's Cipher key and the Establish key command also requires a Host C-MAC. The local host must compute the Host C-MAC and decrypt the response data to obtain the plain-text shared secret that can be used as the pre-master secret of the TLS handshake protocol. The mechanisms linked to the Host C-MAC and Host's Cipher key are not illustrated here (see *Section 3.1*). The local host can now derive the pre-master secret to the master secret and apply the expansion algorithm to obtain the key block; these functions, however, cannot be executed by the STSAFE-A100.
6. The TLS server sends the Certificate Request message including the signature and hash algorithms that are supported by the TLS server. The TLS server also sends the Server *Hello Done* message. The TLS client sends the Certificate message including the X509 certificate chain of the TLS client. This chain may include the X509 certificate of the static private key of the STSAFE-A100, which can be read from it with a Read command (6). This command can typically be executed upon setup of the IoT device and can then be cached by the IoT device so that there is no need any longer to read it from the STSAFE-A100. In case of long certificates, the Read command may be sent multiple times but this is not illustrated in *Figure 8*.
7. The TLS client sends the Client Key Exchange message including the ephemeral Diffie-Hellman public key that was obtained in step 5 in the Establish Key response data from the STSAFE-A100. The TLS client sends the Certificate Verify message including a signature over all handshake messages that have been exchanged so far. The local host may use the STSAFE-A100 for generating this signature. The local host must therefore hash the To Be Signed message and send it to the STSAFE-A100 in the command data of the Generate Signature command (7). The STSAFE-A100 uses its static private key for generating the signature that is returned in the response data.

The handshake protocol continues without any further interaction with the STSAFE-A100.

- The TLS client sends the Change Cipher Spec command.
- The TLS client sends the Finished command including the verify data computed with the Pseudo Random Function, the master secret and all handshake messages exchanged so far.
- The TLS server sends the Change Cipher Spec command.
- The TLS server sends its own Finished command that must be verified by the TLS client.
- The TLS client and TLS server can now exchange applicative data that are protected with keys from the key block. The local host however cannot use the STSAFE-A100 for MACing, encryption and decryption with keys from the key block.

Figure 8. TLS handshake protocol



2.5 Entity authentication

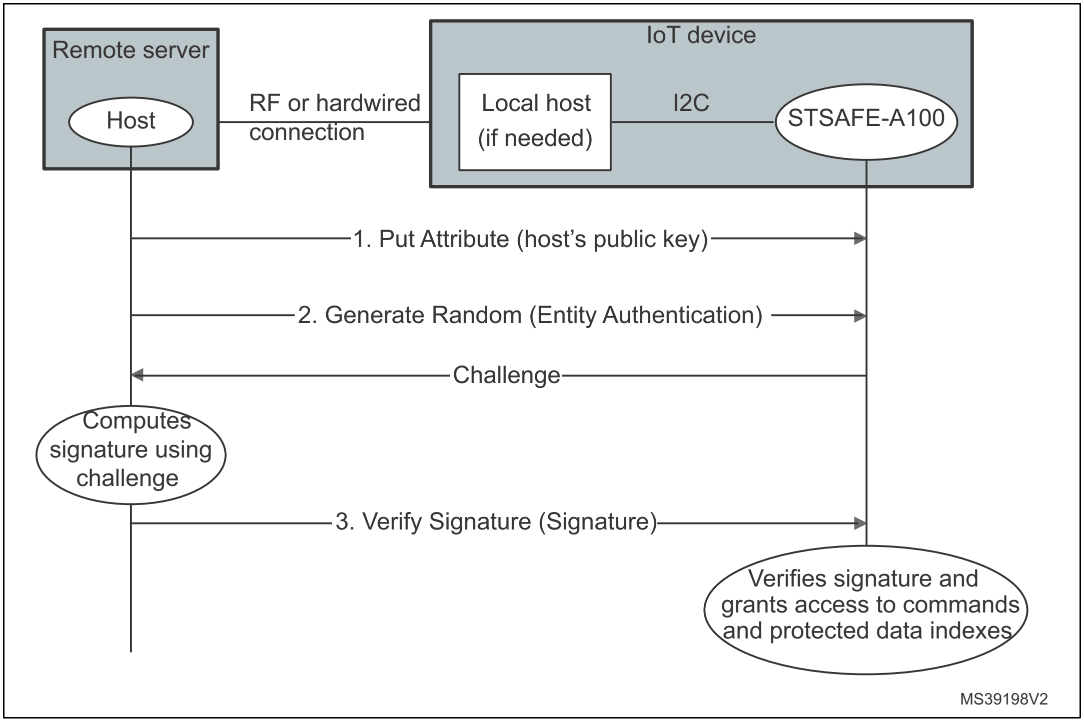
An STSAFE-A100 chip can authenticate an off-chip entity by verifying a digital signature that was generated by the off-chip entity with a private key over a challenge generated by STSAFE-A100.

This functionality requires STSAFE-A100 to contain an authentic copy of the public key corresponding to the private key used by the off-chip entity in its signature generation process.

Command flow (see *Figure 9*)

1. In a trusted environment: Command 1 puts the public key of the remote server inside the STSAFE-A100 chip using the Put Attribute [public key] command.
2. The remote server requests a challenge using the Generate Random [entity authentication] command.
3. The remote server computes a signature using the received challenge and the host's private key.
4. The remote server sends the signature to the STSAFE-A100 chip using the Verify Signature [entity authentication] command.
5. The STSAFE-A100 chip verifies the signature using the remote host's public key and the stored challenge. If the signature is valid, the STSAFE-A100 grants access to commands that require entity authentication.

Figure 9. Entity authentication command flow


Remote server
Host
RF or hardwired connection
IoT device
Local host (if needed)
I2C
STSAFE-A100
1. Put Attribute (host's public key)
2. Generate Random (Entity Authentication)
Challenge
Computes signature using challenge
3. Verify Signature (Signature)
Verifies signature and grants access to commands and protected data indexes
MS39198V2

1. This command flow could also be between the STSAFE-A100 and a local host.

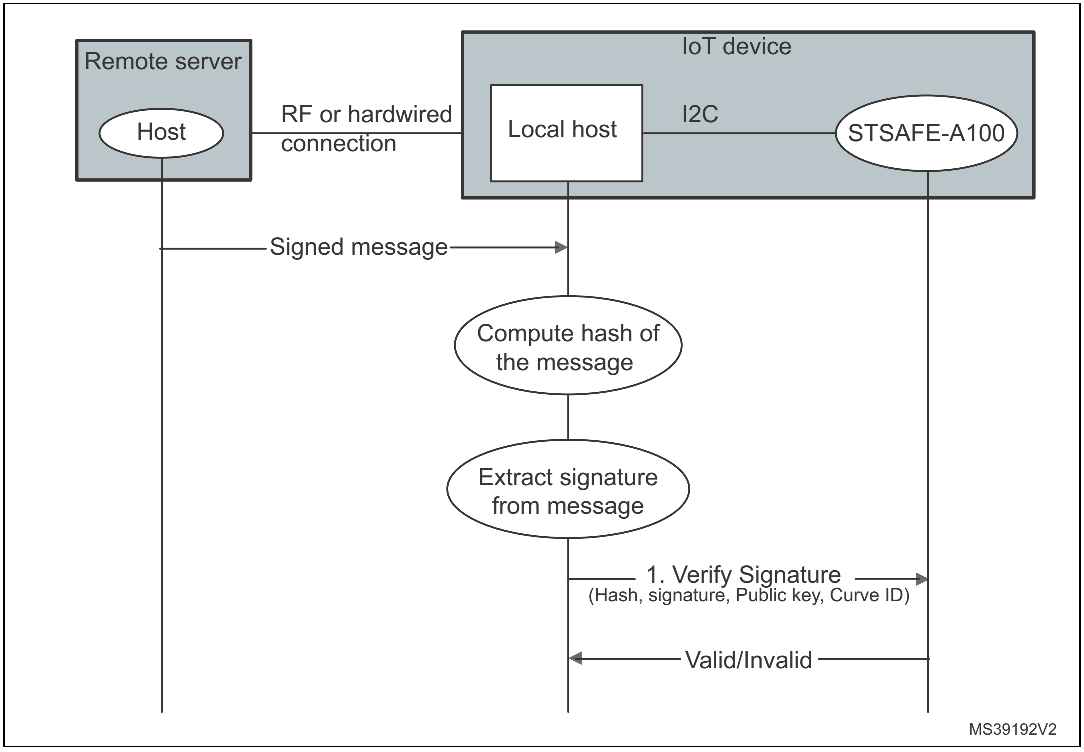
2.6 Public Key Signature Verification

The STSAFE-A100 supports signature verification for local hosts that do not implement EC arithmetics.

Command flow (see *Figure 10*)

1. The local host computes a hash over the message.
2. The local host needs the Public Key that corresponds to the private key that was used to generate the signature. This key could be stored in the STSAFE-A100 data partition, so the local host could use the Read command to get it.
3. The local host needs the ID of the curve used to sign the message.
4. The local host sends the STSAFE-A100 the Verify Signature[Hash, Signature, Public Key, Curve ID] command
5. The STSAFE-A100 verifies the signature and replies with Valid / Invalid.

Figure 10. Public Key Signature Verification command flow


Remote server
Host
RF or hardwired connection
IoT device
Local host
I2C
STSAFE-A100
Signed message
Compute hash of the message
Extract signature from message
1. Verify Signature
(Hash, signature, Public key, Curve ID)
Valid/Invalid
MS39192V2

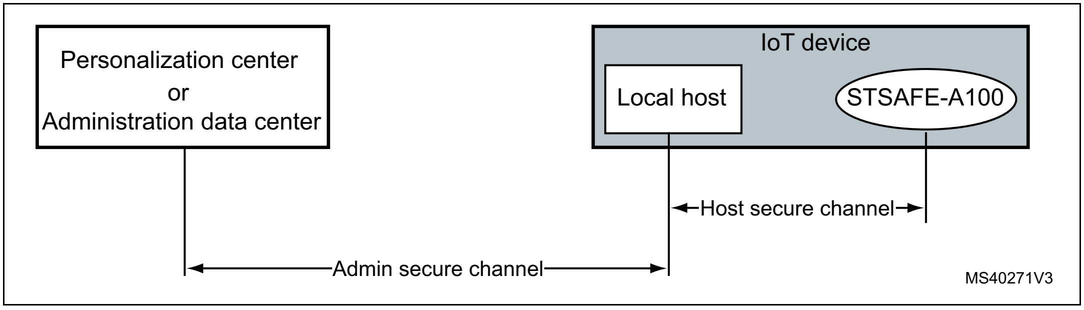
2.7 Applicative data storage

The device comes with 6 Kbytes of EEPROM configurable by the customer for its application data storage. These 6 Kbytes can be partitioned with the appropriate access rights.

3 Symmetric cryptography use cases

With the objective of protecting and authenticating the link between STSAFE-A100 and the local or remote host, some secure channel protocols using symmetric cryptography have been put in place, namely the host secure channel and the admin secure channel.

Figure 11. Host and Admin secure channels



The host secure channel protocol protects the link between a local host and the STSAFE-A100 chip; it constitutes a kind of pairing. It is based on a set of four mechanisms using two symmetric keys, the so-called host keys. These keys are used to MAC the commands (C-MAC) and respective responses (R-MAC). They are also used to encrypt the commands and their respective response to avoid eavesdropping.

The Admin secure channel consists of the C-MAC verification performed by STSAFE-A100 at command reception. It gives STSAFE-A100 the guaranty that the received command has been built by a third party having knowledge of the shared admin secret. Scenarios linked to this secure channel are explained in *Section 3.2*.

3.1 Host secure channel setup use case

In order to execute some specific commands requiring a C-MAC, the local host needs to generate a Host MAC key and a Host cipher key. It puts these keys into the STSAFE-A100's Host key slots.

The PUTATTRIBUTE command is used to input the keys into the STSAFE-A100. It is a free command until the slots are populated.

The DELETEKEY command deletes the keys. This command requires the Admin secure channel.

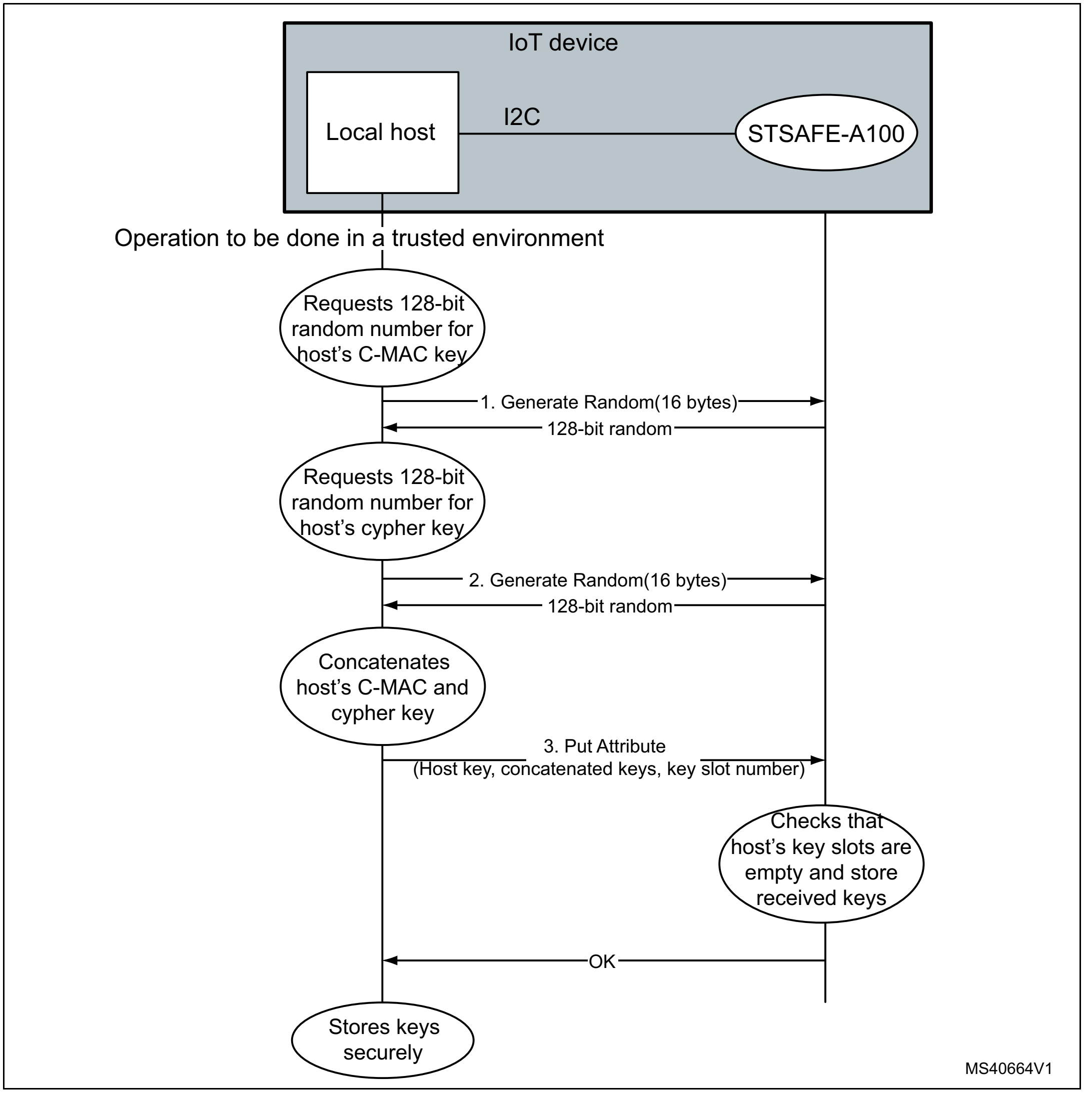
Command flow

This use case assumes that the slots are empty, and cannot be implemented a second time without first deleting the keys present in the slots.

This operation shall be performed in a secure environment, such as the customer manufacturing plant.

1. The local host requests the STSAFE-A100 to generate a 128-bit random to be used as the host C-MAC key.
2. The local host requests the STSAFE-A100 to generate a 128-bit random to be used as the host cipher key.
3. The local host sends the PUTATTRIBUTE command for the “Host key slot” attribute, together with the two generated keys (forming a 256-bit payload).
4. The STSAFE-A100 chip stores the keys into their respective slots and returns a successful response.
5. The local host stores the host C-MAC & cipher keys to a secure area.

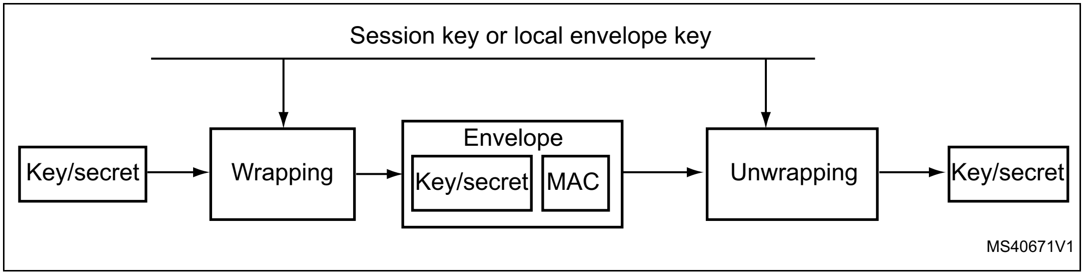
Figure 12. Host secure channel setup case


IoT device
Local host
I2C
STSAFE-A100
Operation to be done in a trusted environment
Requests 128-bit random number for host’s C-MAC key
1. Generate Random(16 bytes)
128-bit random
Requests 128-bit random number for host’s cypher key
2. Generate Random(16 bytes)
128-bit random
Concatenates host’s C-MAC and cypher key
3. Put Attribute (Host key, concatenated keys, key slot number)
Checks that host’s key slots are empty and store received keys
OK
Stores keys securely
MS40664V1

3.2 Wrap/unwrap envelopes

Wrap/unwrap envelope is a mechanism either used to transmit a secret between a server and an IoT over untrusted networks or to securely store NVM keys or plain text to an unprotected IoT. In the former case, we speak of an Issuer envelope whereas in the latter case we speak of a local envelope.

Figure 13. Global principle of the wrapping/unwrapping key



Wrapping is the mechanism used to protect a secret or plain text. The output of wrapping is an envelope.

The envelope consists of the key or plain text to be protected, encrypted with an AES key wrap algorithm. The algorithm uses either a Session Key in the case of an Issuer envelope, or a Local Envelope Key for a local envelope. The envelope also contains the MAC of the encrypted key or plain text to authenticate the envelope.

Unwrapping is the mechanism used to decrypt the envelope and recover the key or plain text.

There could be two use cases for sharing secrets as described below:

- Use case 1: share the secret between the personalization center or the administration data center and the STSAFE-A100 secure element. The method offered by the STSAFE-A100 is to unwrap issuer envelopes.
- Use case 2: share the secret between the local host and the STSAFE-A100. The methods offered are the wrapping/unwrapping of the local envelopes.

3.2.1 Unwrap Issuer Envelope

The STSAFE-A100 supports the UNWRAP ISSUER ENVELOPE command, which has an envelope as command data. This envelop consists of a cryptogram and a MAC computed over it. Upon receiving a command, the STSAFE-A100 verifies the MAC. If valid, the STSAFE-A100 deciphers the cryptogram and returns the payload as the answer to the received command. For MAC verification and decryption, it uses service session keys. Among others, this service allows a working key to be sent from the personalization center or the administration data center with an HSM to a local host with an STSAFE-A100 chip over an untrusted network. The key is protected during its transfer by the envelope. Service session keys are derived from the service base key. Response data are encrypted with the local host's cipher key. As a consequence, the command must also contain a valid Local host's C-MAC.

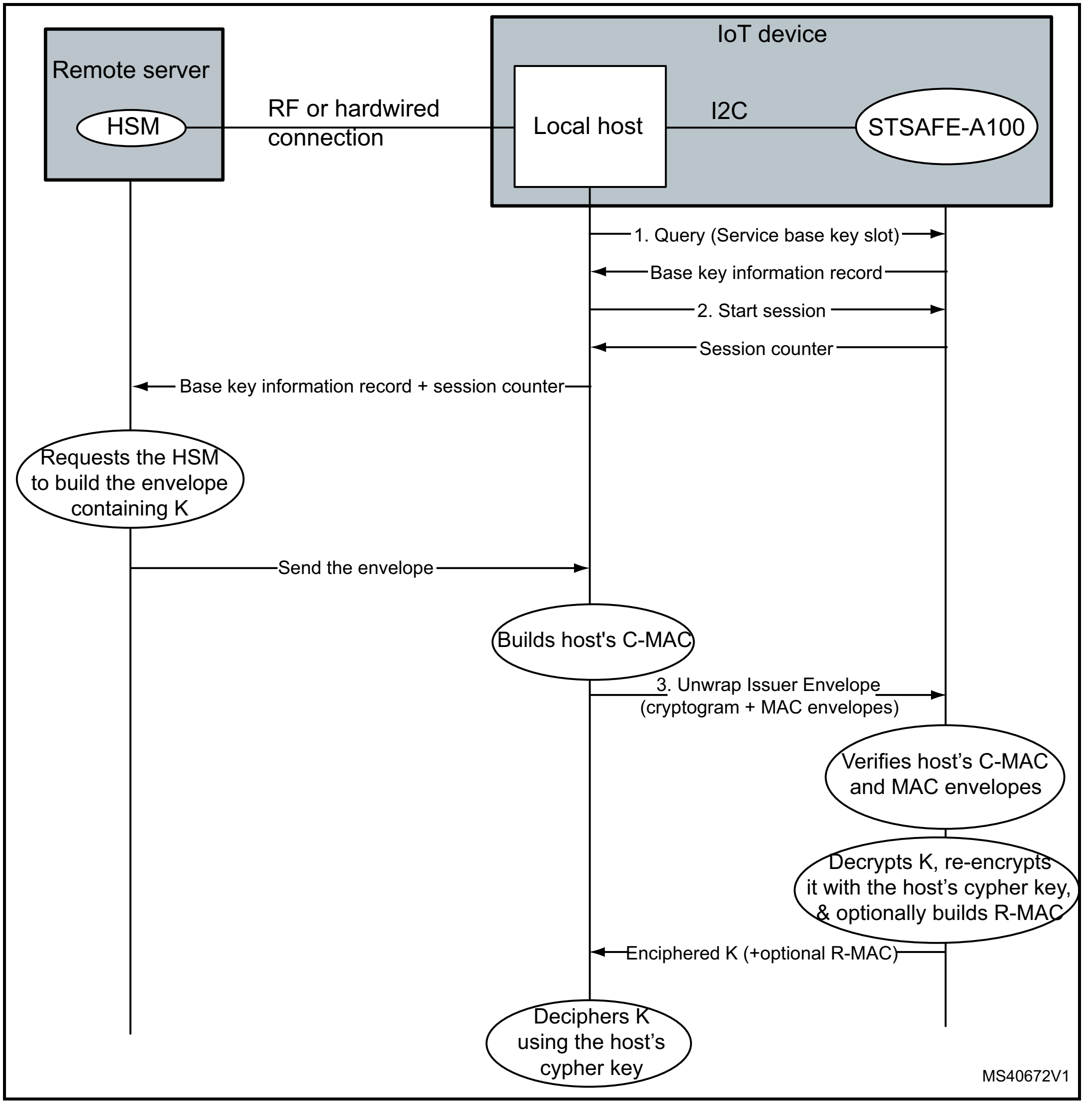
Let us define the sender as the device that generates the envelope using the HSM (usually located in a personalization data center or an administration center). The receiver is the device that receives the envelope and decrypts it to get the key using the STSAFE-A100. The receiver is also known as the local host. K is the Key to be encrypted inside the envelope and to be decrypted using the STSAFE-A100.

Command flow (see *Figure 14*)

1. Establishment of a session key between sender and receiver
   - The local host queries the STSAFE-A100 to get the Base key information record for the Service key (information useful to the sender to know how to derive the same secret as the receiver).
   - The local host Initiates a start session (internally, the STSAFE-A100 establishes the session key derived from the service base key, and builds a session counter. Then it returns the session counter to the local host for sharing with the sender).
   - The local host provides the Base Key information record + Session counter to the sender.
   - Thanks to the base key information record + session counter, the sender establishes a session key using the HSM.

   At the end of this step, both the sender and receiver share the same common secret internally (in the HSM for the sender and in the STSAFE-A100 for the receiver), and this shared secret was never transmitted over the network.
2. Wrapping of issuer envelope by sender
3. Transmission of the issuer envelope over the untrusted network
4. Unwrapping of issuer envelope by receiver
   - The local host provides the issuer envelope (cryptogram + MAC) to the STSAFE-A100 using the Unwrap Issuer Envelope command. Also the local hostmust generate the command C-MAC (in order for the STSAFE-A100 to authenticate the local host).
   - The STSAFE-A100 first verifies the C-MAC using the host's MAC key. If valid, the STSAFE-A100 then checks if the envelope has a correct MAC by using the previously derived session key. If the MAC is valid, the STSAFE-A100 decrypts the cryptogram.
   - The STSAFE-A100 provides the decrypted envelope cryptogram as the response: K re-encrypted with the local host's cipher key.
   - The local host obtains the K key by decryption of the response using the local host's cipher key.

Figure 14. Unwrap Issuer Envelope command flow


Remote server
HSM
RF or hardwired connection
IoT device
Local host
I2C
STSAFE-A100
1. Query (Service base key slot)
Base key information record
2. Start session
Session counter
Base key information record + session counter
Requests the HSM to build the envelope containing K
Send the envelope
Builds host's C-MAC
3. Unwrap Issuer Envelope (cryptogram + MAC envelopes)
Verifies host’s C-MAC and MAC envelopes
Decrypts K, re-encrypts it with the host’s cypher key, & optionally builds R-MAC
Enciphered K (+optional R-MAC)
Deciphers K using the host’s cypher key
MS40672V1

3.2.2 Wrapping and unwrapping local envelopes

The STSAFE-A100 supports the wrapping of working keys managed by the local host. Typical working keys are the local host’s NVM encryption keys.

These working keys can be sent to the STSAFE-A100 chip in the command data of the WRAP LOCAL ENVELOPE command. In the response, an STSAFE-A100 chip returns an envelope, which contains the encrypted working key and a MAC. Such envelope is known as a local envelope.

The local host can use the UNWRAP LOCAL ENVELOPE command to retrieve the working key. The wrapping and unwrapping processes utilize a key from one of the Local Envelope Key slots and the AES key wrap algorithm.

The WRAP LOCAL ENVELOPE command data and the UNWRAP LOCAL ENVELOPE command response data must be encrypted. Both commands require a valid local host’s C-MAC. The local host may optionally request a response MAC (R-MAC).

Local envelope key slots

The STSAFE-A100 supports two local envelope key slots.

Each slot can store an AES-128- or AES-256-bit key that can be used for the wrapping and unwrapping of local envelopes.

A Local Envelope Key is generated randomly by the STSAFE-A100 chip with the GENERATE KEY command, and can take an optional seed. Although this command is a free operation, it is only authorized when the key is not yet present in the slot. As soon as the key is written, the operation is refused.

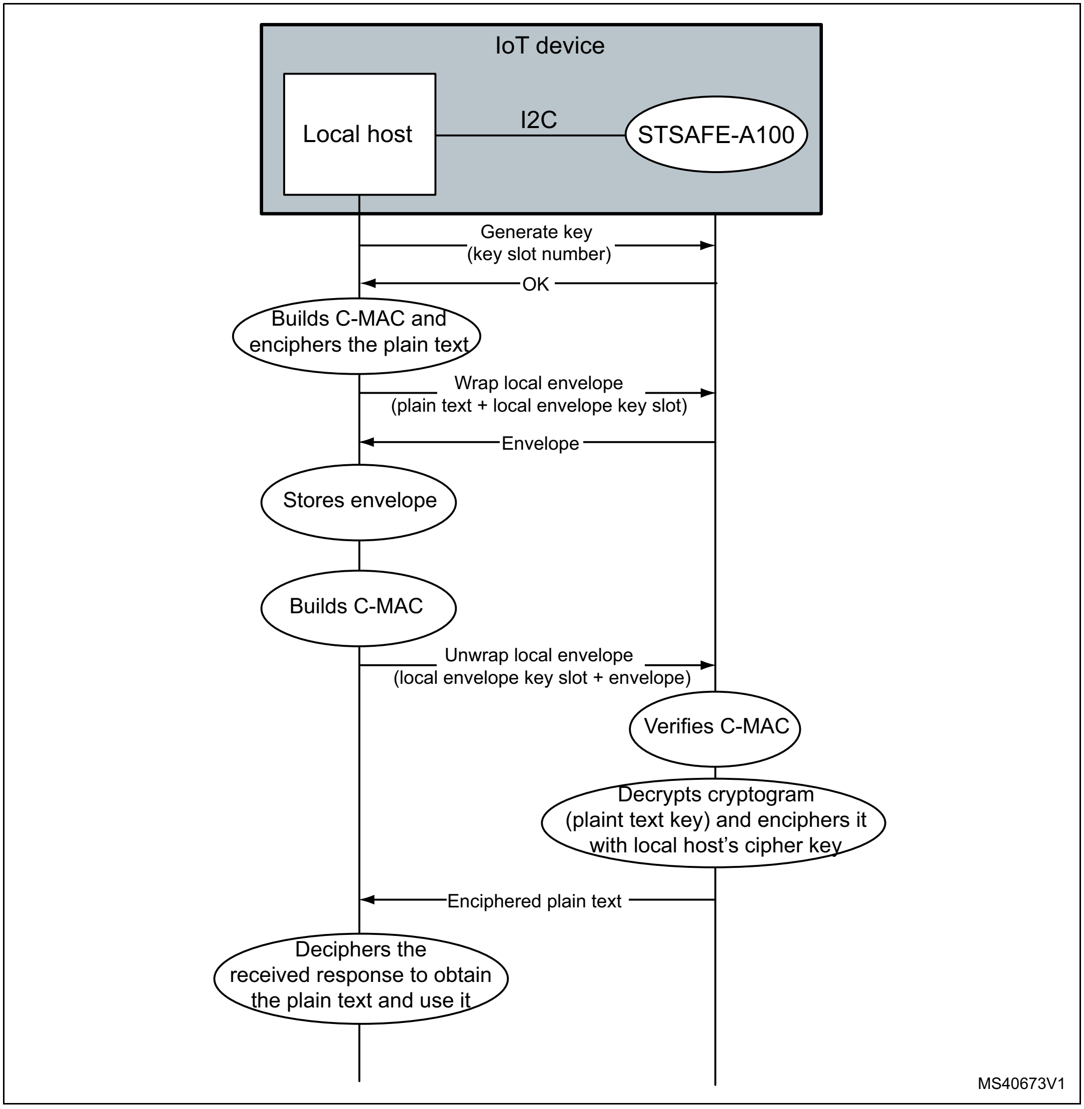
A Local Envelope Key never leaves the STSAFE-A100 chip.

A Local Envelope Key can be deleted from its slot with the DELETE command, which requires authorization with a valid Admin C-MAC. As soon as the key is deleted from its slot, a new key can be generated.

Command flow (see *Figure 15*)

1. Generation of the local envelope Key
   - The local host queries the STSAFE-A100 to randomly generate a local envelope key in one of the two slots, using the GENERATE KEY command.
2. Wrapping of local envelope
   - The local host builds the local envelope by using the STSAFE-A100’s Wrap local envelop command
   - This command requires a local host’s C-MAC, plain text data (usually a cryptographic key that needs to be encrypted with the host’s cipher key) and the local envelope key slot number (the key to use for encryption of plain text data).
   - The response to the Wrap Local envelope command contains the envelope (plain text is encrypted using the Local envelope key).
3. Unwrapping of local envelope by receiver
   - The local host provides the local envelope to the STSAFE-A100 by using the Unwrap local envelop command and the local envelope key slot.
   - The host must generate a local host’s C-MAC with the command.
   - The STSAFE-A100 provides the envelope cryptogram (usually a cryptographic key) decrypted with the local envelope key) in its response.
   - The response is encrypted using the host’s cipher key.
   - The host decrypts the response with the host’s cipher key and obtains the decrypted envelope cryptogram.

Figure 15. Wrap/Unwrap Local Envelop command flow


IoT device
Local host
I2C
STSAFE-A100
Generate key
(key slot number)
OK
Builds C-MAC and enciphers the plain text
Wrap local envelope
(plain text + local envelope key slot)
Envelope
Stores envelope
Builds C-MAC
Unwrap local envelope
(local envelope key slot + envelope)
Verifies C-MAC
Decrypts cryptogram (plaint text key) and enciphers it with local host’s cipher key
Enciphered plain text
Deciphers the received response to obtain the plain text and use it
MS40673V1

4 Command set

Echo

Returns as a response the data that it received as command data.

Reset

Interrupts any on-going session.

Generate Random

Returns the requested number of random bytes.

Start Session

Starts a signature session. It must be used in combination with the Get Signature command.

Hibernate

Sets the product in very-low-power consumption mode. In Hibernate mode, the device remains powered but loses its context.Exiting from Hibernate mode can be performed by triggering the Reset pin or through an I^2C start condition. The device restart is equivalent to a restart after a reset or Power On Reset.

Decrement

Decrements the one-way counter in a counter zone. When the counter reaches zero, the command is refused.

Read

Used to read data from a data partition zone. It will read the data starting from the specified offset within the zone and with the requested length. It will check the access conditions (for example, MAC) and only return the data starting from the specified offset up to the zone boundary.

This command can also be used to change the read access conditions of the zone to a more stricter value.

Update

Used to update data in a zone. It checks if the written data will exceed the zone boundary and if so, does not perform the operation. It also checks whether the access condition is satisfied (for example, MAC) and if not, does not perform the operation.

This command can also be used to change the update access conditions of the zone to a more stricter value; for example, when writing data only once. This command can also be used to perform a bitwise OR with the data being presented, especially when implementing irreversible write operations; that is, bits can only be set to ‘1b’.

Get Signature

This command generates a digital signature over all commands and responses since the start of the signature session.

Generate Signature

This command generates a digital signature over a message digest generated by the host. It is typically used in the IoT device authentication use case.

Verify Signature

This command serves two purposes:

- Message authentication: verifies the signature over the message digest that is computed by the host and given to the STSAFE-A100 chip in the command data together with the public key, the ID of the curve and the signature.
- Entity authentication: verifies the signature generated by an off-chip with a private key based on the challenge generated by the chip. The STSAFE-A100 chip must contain an authentic copy of the off-chip's public key.

Establish Key

This command can be used to establish a shared secret between two hosts by using asymmetric cryptography. The STSAFE-A100 chip computes and provides the shared secret using a private key already in the STSAFE-A100, the provided host's public key and a selected ECC.

Query

Used to check how the chip is configured.

Generate Key

This command is used to generate key pairs (asymmetric cryptography) or local envelope keys (symmetric cryptography).

Unwrap Issuer Envelope

This command is used to unwrap a cryptographic envelope received from a central server.

Wrap Local Envelope

This command is used to wrap data (typically working keys that are entirely managed by the local host) with a local key envelope using an AES key wrap algorithm.

Unwrap Local Envelope

This command is used to unwrap a local envelope with a local envelope key.

Put Attribute

Used to put attributes in the STSAFE-A100 chips like Keys, a password, the host's public key or I2C parameters.

Verify Password

This command performs password verification and remembers the outcome for future authorization of Put Attribute commands.

5 Electrical characteristics

This section summarizes the operating and measurement conditions, and the DC and AC characteristics of the device. The parameters in the DC and AC characteristic tables that follow are derived from tests performed under the measurement conditions summarized in the relevant tables. Users should check that the operating conditions in their circuit match the measurement conditions when relying on the quoted parameters.

5.1 Absolute maximum ratings

Operation of STSAFE-A100 at ranges above the absolute maximum specifications may cause permanent device damage. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 2. Absolute maximum ratings

Name	Description	Conditions	Min.	Max.	Units
$V_{CC\ ABS}$	Absolute maximum power supply	Pins: V_{CC}	–0.3	7	V
V_{IO}	Input or output voltage relative to ground	-	–0.3	$V_{CC\ ABS}$ + 0.3	V
V_{ESD}	Electrostatic Discharge Voltage according to EIA/JEDEC JESD22-A114E specification	Human Body Model. All pins according to specification	-	±5000	V
V_{LU}	Max over voltage for Latch-up Immunity according to EIA/JEDEC - JESD78 specification	Class 1 / Level A Maximum operating temperature		1.5 × $V_{CC\ ABS}$	V
T_A	Ambient operating temperature	-	–40	105	°C
T_{STG}	Storage temperature	-	–65	150	°C
T_{LEAD}	Lead temperature during soldering[(1)]		-	260	°C

1. SO8N and UFDFPN8 lead temperature during soldering shall be compliant with JEDEC Std J-STD-020D (for small body, Sn-Pb or Pb assembly), ST ECOPACK® 7191395 specification, and the European directive on Restrictions on Hazardous Substances (ROHS directive 2011/65/EU, July 2011).

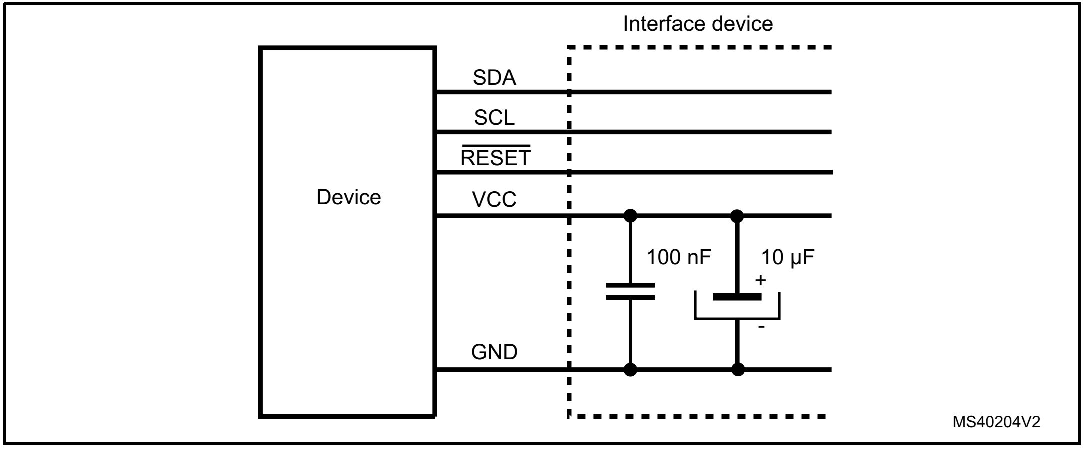
5.2 Power supply

The circuit includes a DC/DC converter that supplies the internal logic and memories with a low operating voltage. The device can operate with external voltages of 1.62 V to 5.5 V nominally, through GND and V_{CC} pins.

In order to filter spurious spikes on the supply voltage pins, decoupling capacitors (100 nF and 10 µF) must be added to the interface device as shown on *Figure 16*. They must be wired between GND and V_{CC} pins.

Note: For each device, the 100 nF decoupling capacitor must be located as close as possible to the device (within a few millimeters). If there are multiple power supplies, a 10 µF filtering capacitor must be located on each one.

Figure 16. Filtering capacitors on V_{CC}



5.2.1 Power supply specifications

Table 3 provides the detailed description of the power requirements of STSAFE-A100.

Table 3. Power supply specifications

Name	Description	Conditions	Min.	Typ.	Max.	Units
V_{POR}	Power on reset voltage	-	1.35	1.45	1.55	V
V_{CC}	Supply voltage	V_{CC} to GND	1.62	-	5.5	V
$V_{CC\text{-}HIPS}$	High power supply detection	Ambient temperature (25 °C)	5.6	6.3	6.9	V
$I_{CC\text{-}PROC}$	Supply current while processing a command	Ambient temperature (25 °C)	14	18	21	mA
$I_{CC\text{-}STDBY}$	Supply current in standby	IO pulled up to V_{CC}, T_A = 25 °C, 3 V to 5 V	160	245	460	µA
$I_{CC\text{-}RESET}$	Supply current during reset	$\overline{\text{RESET}}$ = 0	200	450	800	µA
$I_{CC\text{-}HIBERNATE}$	Supply current during hibernate	$\overline{\text{RESET}}$ = 1 [1] T_A = 25 °C	0.2	1.1	3	µA

[1] $\overline{\text{RESET}}$ must be tied to V_{CC} ± 200mV in case of Wake-up from Hibernate on Reset event selected. RESET, SDA and SCL must be tied to V_{CC} ± 200mV in case of Wake-up from Hibernate on Reset event or I²C start condition selected.

5.2.2 Power-on and power-off sequences, and power supply glitch tolerance

The power-on sequence on STSAFE-A100 products need to follow the requirements mentioned below:

- The $\overline{\text{RESET}}$ pin must not be tight to High prior to the V_{CC} power pin.
- The $\overline{\text{RESET}}$ pin must be tied low prior to or simultaneously with the V_{CC} pin.
- The voltage applied to the V_{CC} pin must be less than or equal to 0.3 V prior to starting a new power-on sequence.

For security purposes, the STSAFE-A100 embeds detectors. When these are trigged, the STSAFE-A100 device enters the Reset state until a power cycle or a reset event occurs.

5.2.3 Reset pin (external reset)

The circuit is in reset state when the Reset signal available on the $\overline{\text{RESET}}$ pin is at logical level ‘0’. If this signal is low for less than t_{WL}, it is not taken into account.

When the $\overline{\text{RESET}}$ pin is floating, an external reset is not available and the device will remain in a Reset state as the pin is connected to an internal weak pull-down.

When pin V_{CC} is tied high, if the $\overline{\text{RESET}}$ pin switches from high to low and then to high again, a warm reset occurs. For more information, refer to *Figure 18: Warm reset sequence on page 29*.

5.2.4 Power-on and reset sequence

Figure 17. Power-on and reset sequence

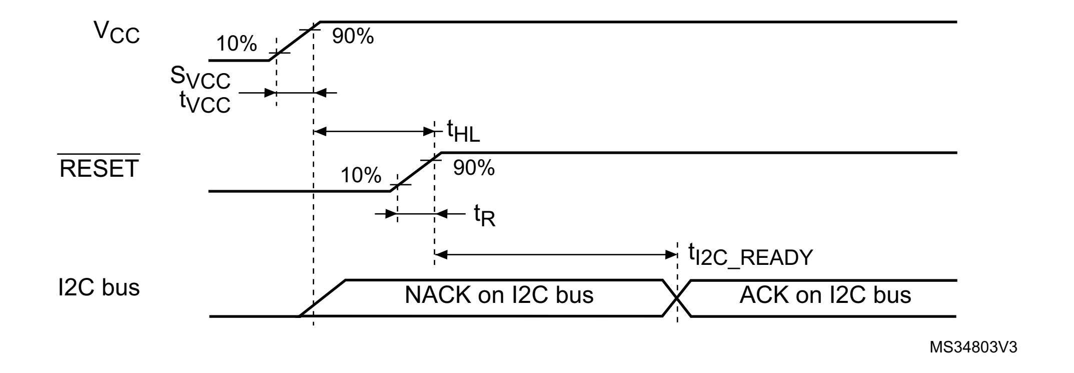


Figure 18. Warm reset sequence

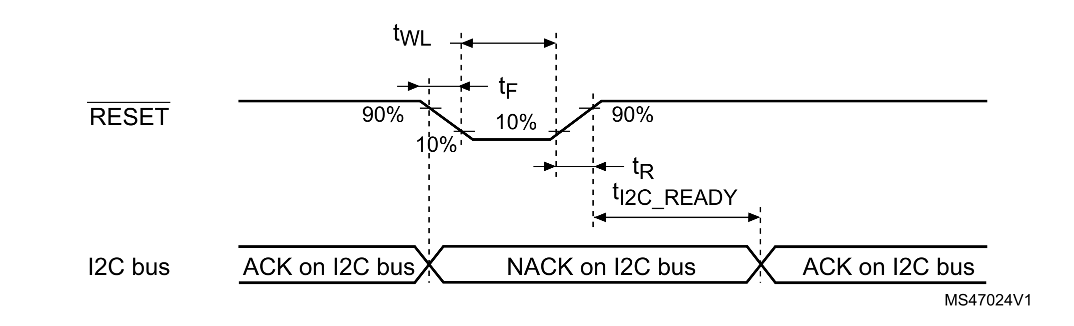


Table 4. Power-on and reset sequence timings

Name	Description	Conditions	Min.	Typ.	Max.	Units
t_{HL}	Minimum time before de-asserting $\overline{\text{RESET}}$ after power-up	-	0	-	-	µs
S_{VCC}	V_{CC} rising slope (from 10% to 90% of nominal value)	-	0.05	-	5	V/µs
T_{set_4mA}	Minimum time required to supply 4 mA	From POWER OFF	-	-	500	ns
		From IDLE	-	-	150	

Table 4. Power-on and reset sequence timings (continued)

Name	Description	Conditions	Min.	Typ.	Max.	Units
t_{WL}	Pulse width for Reset	-	1	-	-	µs
t_R/t_F Reset	Reset rise and fall time	$V_{CC} > V_{POR}$	-	-	1	µs
$t_{I2C\_}$ READY	Delay for STSAFE-A100 to accept I^2C commands after a reset sequence.	-	20	-	50	ms

5.2.5 Power consumption optimization

When the STSAFE-A100 is not in use, it is possible to decrease its power consumption by removing the power supply properly, knowing that the power supply must remain for operations that require a context, for instance during sessions.

This could be achieved by using a transistor to pilot the STSAFE-A100 power supply, or by using a GPIO able to provide a $I_{CC\text{-}PROC}$ current that respects the STSAFE-A100 powering conditions.

5.3 DC characteristics

The following tables provide the detailed description of the DC operating conditions of STSAFE-A100 from 1.62 V to 5.5 V voltages.

Table 5. DC operating specifications and input parameters

Name	Description	Conditions	Min.	Max.	Units
V_{IH}	Input high voltage	T = 25 °C	$0.7 \times V_{CC}$	-	V
V_{IL}	Input low voltage	T = 25 °C	0	$0.3 \times V_{CC}$	V
I_{IH}	Input high current	RST	0	20	µA
		SDA, SCL	−1	1	
I_{IL}	Input low current	RST	0	2	µA
		SDA, SCL	−1	1	µA
V_{OL}	Output low voltage	I_{OL} = 1 mA	-	0.54	V
I_{OL}	Output low current	V_{CC} = 3.3 V and V_{OL} = 0.4 V	3	-	mA
CIN1	SCL input capacitance	V_{IN} = 0 to $V_{CC\ Max}$	-	30	pF
CIN2	SDA input capacitance	V_{IN} = 0 to $V_{CC\ Max}$	-	30	pF

Note: $V_{CC\ Max}$ is the maximum V_{CC} as defined in Table 3: Power supply specifications.

5.4 AC characteristics

Table 6. AC characteristics

Name	Description	Min.	Typ.	Max.	Units
t_R, t_F Reset	Reset rise and fall time	-	-	1	µs
t_{WL}	Pulse width for Reset	1	-	-	µs

Table 7. I²C operating conditions

Name	Description	Standard mode		Fast mode		Units
		Min.	Max.	Min.	Max.	
f_{SCL}	SCL frequency of sub-device: processor	-	100	-	400	kHz
$t_{HD;STA}$	Input low to Clock low (Start condition hold time)	4.0	-	0.6	-	µs
t_{LOW}	Low period of SCL clock	4.7	-	1.3	-	µs
t_{HIGH}	High period of SCL clock	4.0	-	0.6	-	µs
$t_{SU;STA}$	Clock high to Input Transition / Setup time for a (repeated) Start condition See Note	4.7	-	0.6	-	µs
$t_{HD;DAT}$	Clock low to Input transition	0[1]	[2]	0[1]	[2]	µs
$t_{SU;DAT}$	Input transition to Clock transition Data setup time	250	-	100	-	ns
$t_{SU;STO}$	Clock high to Input high (Stop)	4.0	-	0.6	-	µs
t_{BUF}	Input high to Input low (bus free between stop and start)	4.7	-	1.3	-	µs
t_R	Clock and Data rise time on load capacitance of 30 pF	-	1000	20	300	ns
t_F	Clock and Data fall time on load capacitance of 30 pF	-	300	10	300	ns

1. The device must internally provide a hold time of at least 300 ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.
2. The maximum $t_{HD;DAT}$ could be 3.45 µs and 0.9 µs for Standard mode and Fast mode, but must be less than the maximum of $t_{VD;DAT}$ or $t_{VD;ACK}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL signal, the data must be valid by the setup time before it releases the clock.

Table 8. I²C filter characteristics

Symbol	Parameter	Min	Max	Unit
t_{SP}[1]	Pulse width of spikes that are suppressed by filter	0	50	ns

1. Guaranteed by design, not tested in production

Figure 19. AC clock and data timings

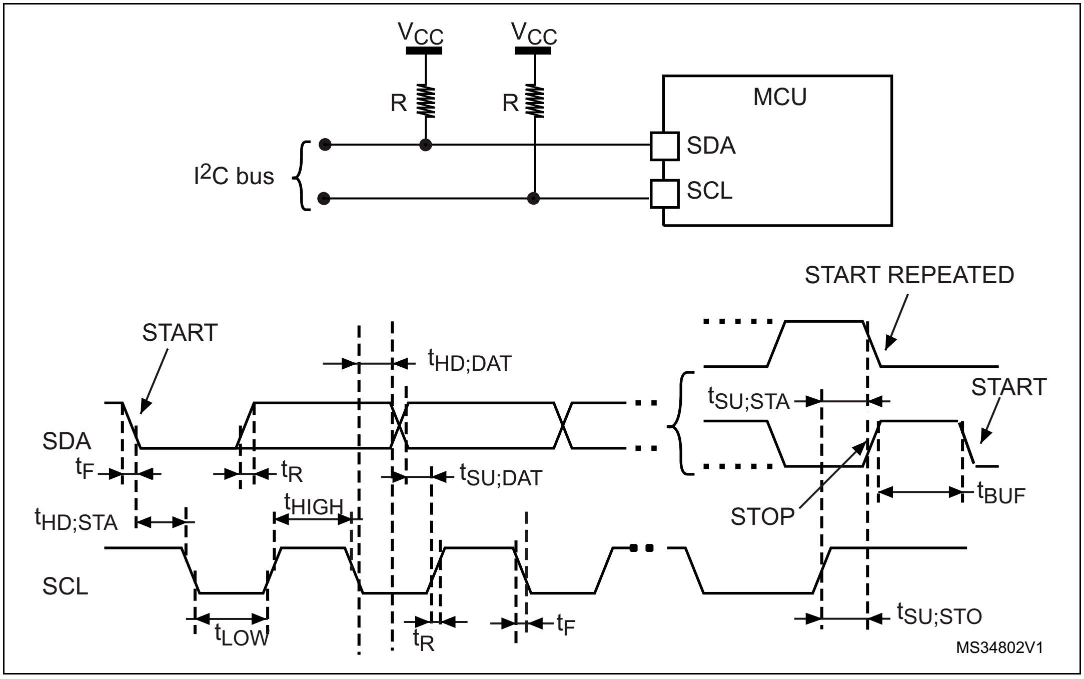


Table 9. AC measurement conditions

Description	Range	Units
Input pulse voltages	0.2 × V_{CC} to 0.8 × V_{CC}	V
Input and Output timing reference voltages	0.3 × V_{CC} to 0.7 × V_{CC}	V

6 Package information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

6.1 SO8N package information

Figure 20. SO8N – 8-lead plastic small outline, 150 mils body width, package outline

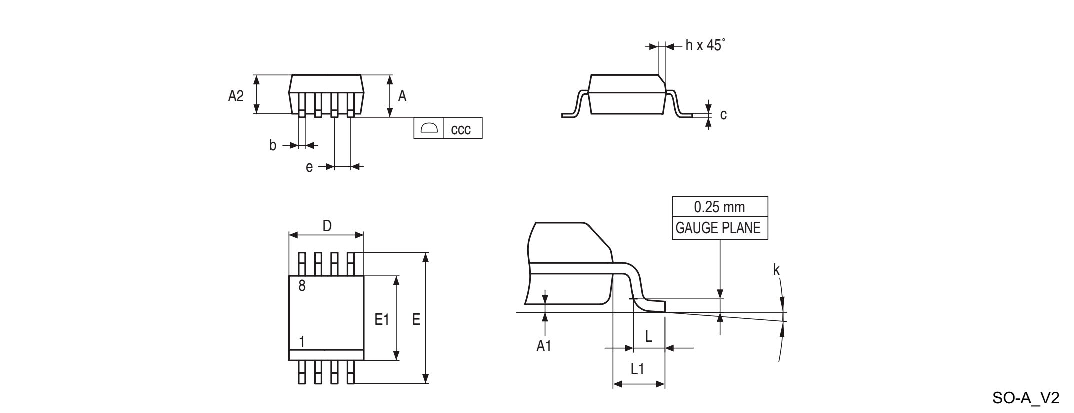


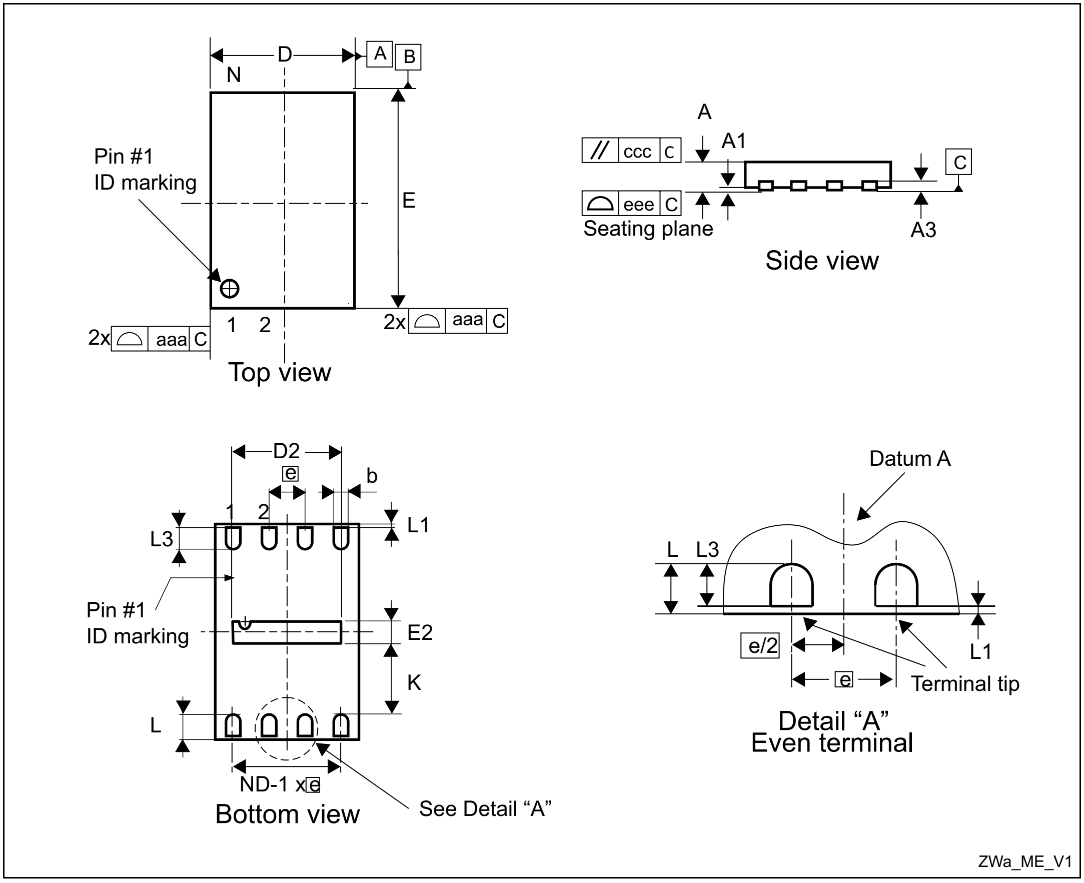
Table 10. SO8N – 8-lead plastic small outline, 150 mils body width, package mechanical data

Symbol	millimeters			inches [(1)]		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	-	-	1.750	-	-	0.0689
A1	0.100	-	0.250	0.0039	-	0.0098
A2	1.250	-	-	0.0492	-	-
b	0.280	-	0.480	0.0110	-	0.0189
c	0.170	-	0.230	0.0067	-	0.0091
ccc	-	-	0.100	-	-	0.0039
D	4.800	4.900	5.000	0.1890	0.1929	0.1969
E	5.800	6.000	6.200	0.2283	0.2362	0.2441
E1	3.800	3.900	4.000	0.1496	0.1535	0.1575
e	-	1.270	-	-	0.0500	-
h	0.250	-	0.500	0.0098	-	0.0197
k	0°	-	8°	0°	-	8°
L	0.400	-	1.270	0.0157	-	0.0500
L1	-	1.040	-	-	0.0409	-

1. Values in inches are converted from mm and rounded to four decimal digits.

6.2 UFDFPN8 package information

Figure 21. UFDFPN8 - 8-lead, 2 × 3 mm, 0.5 mm pitch ultra thin profile fine pitch dual flat package outline



1. Max. package warpage is 0.05 mm.
2. Exposed copper is not systematic and can appear partially or totally according to the cross section.
3. Drawing is not to scale.

Table 11. UFDFPN8 - 8-lead, 2 × 3 mm, 0.5 mm pitch ultra thin profile fine pitch dual flat package mechanical data

Symbol	millimeters			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	0.450	0.550	0.600	0.0177	0.0217	0.0236
A1	0.000	0.020	0.050	0.0000	0.0008	0.0020
b[2]	0.200	0.250	0.300	0.0079	0.0098	0.0118
D	1.900	2.000	2.100	0.0748	0.0787	0.0827
D2	1.500	1.600	1.700	0.0591	0.0630	0.0669
E	2.900	3.000	3.100	0.1142	0.1181	0.1220
E2	0.100	0.200	0.300	0.0039	0.0079	0.0118
e	-	0.500	-		0.0197	
K	0.800	-	-	0.0315	-	-
L	0.400	0.450	0.500	0.0157	0.0177	0.0197
L1	-	-	0.150	-	-	0.0059
L3	0.300	-	-	0.0118	-	-
aaa	-	-	0.150	-	-	0.0059
bbb	-	-	0.100	-	-	0.0039
ccc	-	-	0.100	-	-	0.0039
ddd	-	-	0.050	-	-	0.0020
eee[3]	-	-	0.080	-	-	0.0031

[1] Values in inches are converted from mm and rounded to 4 decimal digits.

[2] Dimension b applies to plated terminal and is measured between 0.15 and 0.30 mm from the terminal tip.

[3] Applied for exposed die paddle and terminals. Exclude embedding part of exposed die paddle from measuring.

7 Ordering code

Example: STSAFA100 X SR xxx

Product name

STSAFA100 = STSAFE-A100

Option

X

Package codification

SR8 = SO8N tape and reel
DF = UFDFPN8

Personalization revision code

xxx = Personalization setup

Note: For a list of available options (speed, package, etc.) or for further information on any aspect of this device, please contact your nearest STMicroelectronics sales office.

8 Revision history

Table 12. Document revision history

Date	Revision	Changes
18-Feb-2016	1	Initial release.
29-Apr-2016	2	Corrected *Figure 7: Key Establishment command flow*. Updated *Table 5: DC operating specifications and input parameters*. Updated *Section 7: Ordering code*.
18-Nov-2016	3	Removed V_{CC} *rising slope* table. Updated NC description in *Table 1: Pin description*. Updated SVCC description in *Table 4: Power-on and reset sequence timings*. Added *Section 5.2.5: Power consumption optimization*. Updated IIH and IIL in *Table 5: DC operating specifications and input parameters*.
08-Dec-2016	4	Updated operating temperature in *Section : Hardware features* and in *Table 2: Absolute maximum ratings*.
10-Jul-2017	5	The note on TLEAD in *Table 2: Absolute maximum ratings* also applies to UFDFPN8 packages. Updated *Section 5.2.2: Power-on and power-off sequences, and power supply glitch tolerance*. Updated *Figure 17: Power-on and reset sequence* and *Figure 18: Warm reset sequence*. Updated *Section 5.2.5: Power consumption optimization*.
24-Nov-2017	6	Updated introductory text in *Table 5: Electrical characteristics*. Moved content of *Section 5.1 Signal description* to *Section 1.3: Pin descriptions*. Added V_{IO} to *Table 2: Absolute maximum ratings*. In *Table 5: DC operating specifications and input parameters*: – updated Max. value of V_{IL} – updated conditions for V_{OL} – added I_{OL} Small text changes.
29-Nov-2018	7	Updated confidentiality of document from ST Restricted to Public. Updated *Section 5.2.2: Power-on and power-off sequences, and power supply glitch tolerance*.
05-Feb-2019	8	Updated document reference from DS_STSAFE-A100 to DS12911. Added STSAFE-A logo.