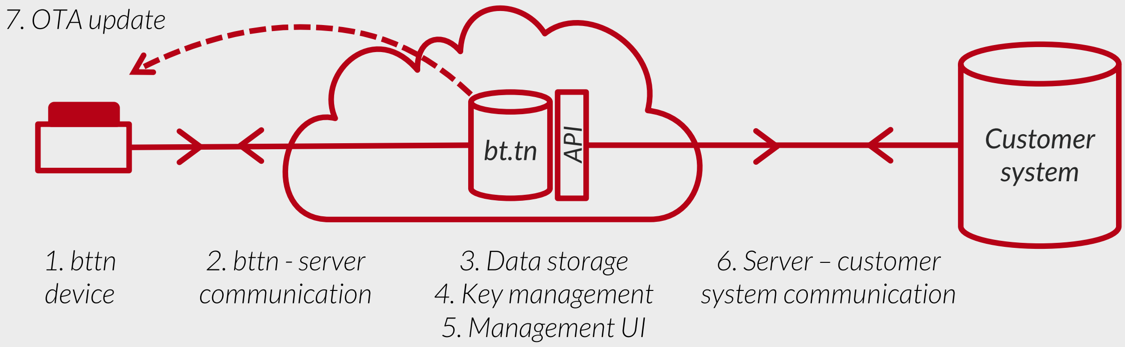
bt.tn®

Connected buttons and data security:

bttn – the IoT solution that doesn't compromise your data security.

The professional team behind **bttn** has vast experience in developing and installing strong encryption systems for enterprise and government customers. Similarly, the bttn device and bt.tn cloud service have been **designed and built to be inherently secure and fit for business use**. The key to bttn's unrivalled security is simple: The system does not store or transmit any critical data where it could be compromised.

Overview of the secure end-to-end architecture:



1. bttn device

The device contains no information about its user or owner or any access keys to customer systems. The memory holds only very long random key material for secure authentication with server.

2. bttn – server communication

The device communicates over HTTP with two-way message authentication. Communication is always initiated by the device. Random keys are generated on-demand per transaction to prevent device spoofing and message replay.

3. Server data storage & availability

The bt.tn cloud service runs in AWS cloud (Ireland) and implements a reliable and failsafe cloud infrastructure with database replication. The service is monitored 24/7 for load, functionality and any exceptions.

4. Device key management

bt.tn server can revoke compromised device access keys and update keys securely.

5. Management UI

The my.bt.tn management interface is protected with HTTPS and username/password authentication. Social Sign-in is also available.

6. Server – customer system communication

Machine-to-machine communication takes place via HTTP(S), using webhooks or bt.tn REST API. Available authentication options are OAUTH1/2, username/password, HTTP basic/digest authentication, or API keys.

7. OTA update

Device firmware can be updated via a secure, opt-in Over-The-Air process with built-in error-checking.

For further information, please contact:
Juuso Pesola, Co-Founder, CISSP
juuso.pesola@bt.tn

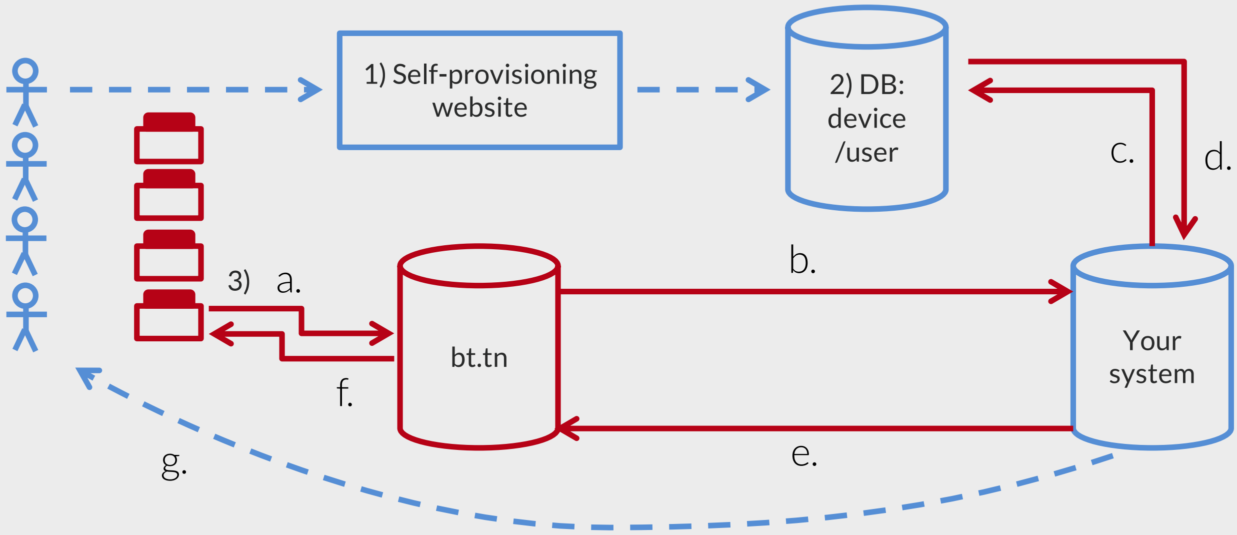
bt.tn®

Best practice:

Recommended way of working with customer data.

A large end-user deployment is easy to do with bttns. This example shows a way to link a bttn device to a customer ID in your backend, enabling one-push ordering of your product or service. All critical data and processes such as billing take place inside your own system.

Architecture for a secure self-provisioning deployment (example):



1. Self-provisioning website

When you are distributing thousands of bttns, the best way to tie a bttn device to an end-user is to give end-users a simple website form to enter their details:

- bttn device's unique code (printed on a sticker on the bottom of the bttn).
- A piece of information that identifies the user at your end, e.g. customer ID and/or email address.
- Optionally a selection of items / services to order when pushing the bttn.

The website can be hosted by you, or you can get it custom-made and hosted by us.

2. Database for linking device to a user

The website writes (but cannot read) the entered end-user information to a database.

The database can be hosted by you, or you can get it custom-made and hosted by us.

3. Transaction logic, example

a. "bttn #1234 has been pushed".
b. bt.tn server sends a request to your system, including device ID for bttn #1234 .
c. Your system asks whether there is valid user data in the database for bttn #1234.
d. If yes, the database returns e.g. the user's ID or other piece of data required for completing the process.
e. After your system has processed the request, it returns "OK" or "FAIL" to bt.tn server.
f. bt.tn server tells bttn device to show green or red lights based on the result.
g. Your system sends an email receipt to the end-user "Thank you for purchasing".

For further information, please contact:
Juuso Pesola, Co-Founder, CISSP
juuso.pesola@bt.tn