OPENBOUNCER Authentication Protocol

Milosch Meriac Henryk Plötz

October 2, 2008

1 Protocol

The door (identified by 32 bit identifier ID) and the tag (not explicitly identified, herein referred to as T) share a common key Key(ID, T). The tag can store multiple (ID, Key(ID, T)) pairs to be used for multiple different doors. The door will store multiple Key(ID, x) for all tags x in the set of allowed tags. No explicit tag identifier is used in the protocol or stored on the tag and it's up to the key management at the door to provide a set of keys for all allowed tags (and allow to remove specific tags from this set).

The tag authenticates to the door with the shared key in a challenge/response scheme. The main cryptographic primitive used for the protocol is xxTea, used as a keyed hash function.

1.1 Protocol goals

The challenge/response protocol should provide the following (all are assuming that the attacker does not possess the relevant keys):

Authentication An attacker can not successfully perform the protocol.

Freshness An attacker can not simply replay a tag's response to authenticate to the door.

Anonymity on the radio channel

1. An attacker that passively listens in on successful door/tag transactions can not distinguish between different tags or recognize the identity of any tag.
2. An attacker that actively interrogates tags can not distinguish between different tags or recognize the identity of any tag.

Limited protection against forwarding An attacker can not successfully convince the door that a tag is in radio range of the door when it in reality is not, without using advanced equipment.

Protection of the door associations An attacker that is in possession of a tag can not determine whether that tag shares a common key with a specific door without trying to use that tag at that door.

1.2 Protocol description

Idea Door and tag each generate and exchange a 64 bit nonce, the door nonce is called challenge, the tag nonce is called salt. The tag uses xxTea with the door specific key $\text{Key}(\texttt{ID}, T)$ to encrypt a concatenation of the tag salt, door challenge and door identifier into a 256 bit (32 bytes) response. This response is never fully transmitted over the air. Instead the door generates 8 random byte offsets into the response (i_0 through i_7) and transmits this list of offsets to the tag. The tag responds immediately by transmitting the 8 (out of 32) bytes of the precomputed response indicated by these offsets (and then overwrites the full response in its memory). Under no circumstance will the tag transmit more than 8 bytes of response per protocol run. The door measures the response time between sending the offset list and receiving the response bytes and rejects all responses that take 'too long' (how long exactly is 'too long' remains to be determined, but it is expected that this time does not exceed 1 ms, since no additional cryptographic operations are necessary on the tag).

The door should then determine if the tag is in the list of allowed tags by replicating this computation for all stored tag keys.

Note: Care should be taken to ensure that all calculations on tag and door do not differ in timing or power consumption for the different cases of "tag has a matching key for the door `ID` (or not)" and "door has a matching key for the tag response (or not)". See the section 2.3.

Protocol run A successful run of the protocol is as follows*:

1. *Tag* comes up with random numbers Salt_A^4 and Salt_B^4, see section 2.1
2. *Tag* transmits to *Door*

$$\texttt{HELLO}(\text{Retrycounter}^1, \text{Salt}_A^4)$$

3. *Door* transmits to *Tag*

$$\texttt{SETUP}(\texttt{ID}^4, \text{Challenge}^8)$$

4. *Door* waits $\delta t > 42\,\text{ms}$, while
5. *Tag* computes

$$\text{Response}_T^{32} \leftarrow \text{xxTea}_{\text{Key}(\texttt{ID},T)}\left(\text{Salt}_A^4 \parallel \text{Salt}_B^4 \parallel \text{Challenge}^8 \parallel \texttt{ID}^4 \parallel \overrightarrow{0}^{12}\right)$$

*All superscripts indicate the length of the respective field in bytes; square brackets indicate a selection of indexed byte offsets from a larger field; $\overrightarrow{0}$ is a sequence of zero bytes of the indicated length

6. *Door* transmits to *Tag*

$$\texttt{CHALLENGE}(i_0^1, i_1^1, i_2^1, \ldots, i_7^1)$$

7. *Door* starts timer

8. *Tag* transmits to *Door*

$$\texttt{RESPONSE}(\text{Response}_T[i_0]^1, \text{Response}_T[i_1]^1, \ldots \text{Response}_T[i_7]^1, \text{Salt}_B^4)$$

9. *Door* stops timer and checks that $\delta t < 1\,\text{ms}$

10. *Door* calculates Response_x^{32} for all Tags in the list of acceptable tags and checks that there is exists an allowed Tag x for which

$$(\text{Response}_x[i_0]^1, \text{Response}_x[i_1]^1, \ldots \text{Response}_x[i_7]^1)$$

matches what the Tag sent in its `RESPONSE` message.

Retries It is assumed that the tag will always transmit at full transmit power, while the receive sensitivity on the door side can be reduced if necessary. Radio transmissions from door to tag will not be protected specially (the door will eventually just time out), while all transmissions from tag to door will be protected with the automatic acknowledgement mechanism of the underlying radio protocol. This ensures that the tag will detect when either the `HELLO` or the `RESPONSE` packet were not received by the door.

If the `HELLO` packet is not acknowledged it can be retransmitted until it is either received or the attempt times out. If the `RESPONSE` packet is not acknowledged it can *not* be retransmitted. Instead the tag should start a new protocol run with new random numbers and a new `HELLO` packet. In this case the tag will increment the Retrycounter field in the `HELLO` packet (which otherwise is set to 0) so that the door is informed about the aborted previous attempt (which might indicate poor radio reception or other causes that should be investigated).

2 Tag considerations

2.1 Random number generator

Since the tag most likely does not have a proper physical random number generator it will use the xxTea cryptographic primitive with a counter stored in EEPROM and secret data from EEPROM (D_T and Counter) and Flash (FlashSalt):

$$\begin{aligned} \text{RND}^{32} &\leftarrow \text{xxTea}_{D_T}\left(\text{Counter}_T^4 \parallel \text{FlashSalt}_T^{28}\right) \\ \text{Counter}_T^4 &\leftarrow \text{Counter}_T^4 + 1 \end{aligned}$$

$$\begin{aligned} \text{Salt}_A^4 &\leftarrow \text{RND}[0\ldots3]^4 \\ \text{Salt}_B^4 &\leftarrow \text{RND}[4\ldots7]^4 \end{aligned}$$

FlashSalt represents real entropy that was generated at tag production time. D_T is the dummy door key (see section 2.3). Counter represents the state of this pseudo-random number sequence (guaranteeing a sequence length of 2^{32}).

2.2 Data stored in tag T

Flash FlashSalt_T^{12}: random confidential data, generated at production

EEPROM D_T and Counter are initialized to random confidential data at production. All unused $\texttt{ID}_x$ fields are initialized to `0x00000000`.

Figure 1 EEPROM memory map

Offset	Length	Contents
0	4	$\texttt{ID}_0$
4	16	$\text{Key}(\texttt{ID}_0, T) \oplus D_T$
20	4	$\texttt{ID}_1$
24	16	$\text{Key}(\texttt{ID}_1, T) \oplus D_T$
⋮	⋮	⋮
80	4	$\texttt{ID}_4$
84	16	$\text{Key}(\texttt{ID}_4, T) \oplus D_T$
100	16	D_T
116	4	Counter_T

2.3 Door lookup on tag

In order not to leak information about whether a given tag has an association with a given door (identified by `ID`), even when the attacker is in physical possession of the tag and can interrogate it, the door lookup algorithm (described in algorithm 1) is used.

This algorithm guarantees two things:

- The lookup takes the same time, regardless of whether an association for the requested `ID` is stored or not (except for two well-known invalid `ID`s).
- If no association for the requested `ID` is stored then the lookup returns a dummy key (random, confidential, generated at tag production time) that is unknown to the attacker. (And not, for example, a well-known key consisting only of zeros.)

Algorithm 1 Door lookup algorithm on tag

function LOOKUPDOORKEY(ID)
  if ID $=$ 0x00000000 **then**
    return Failure
  end if
  if ID $=$ 0xFFFFFFFF **then**
    return Failure
  end if
  $a \leftarrow 0$
  $b \leftarrow 0$
  for $(i, k) \in$ Associations **do** ▷ Not including dummy association (ID_D, D_T)
    if $i =$ ID **then**
      $a \leftarrow k$ ▷ a is now Key$(\mathrm{ID}, T) \oplus D_T$
    else ▷ Both alternatives shall take exactly the same execution time
      $b \leftarrow k$
    end if
  end for
  $a \leftarrow a \oplus D_T$ ▷ a is now the door key or D_T if the door is not associated
  return a
end function

3 Door considerations

4 Attacks and defenses