

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 August 2022

C. Bormann
Universität Bremen TZI
Y. Li
Huawei Technologies
7 February 2022

SWORN: Secure Wake on Radio Nudging
draft-bormann-t2trg-sworn-05

Abstract

Normally off devices ([RFC7228](#)) would need to expend considerable energy resources to be reachable at all times. Instead, MAC layer mechanisms are often employed that allow the last hop router of the device to "wake" the device via radio when needed. Activating these devices even for a short time still does expend energy and thus should be available to authorized correspondents only. Traditionally, this has been achieved by heavy firewalling, allowing only authorized hosts to reach the device at all. This may be too inflexible for an Internet of Things.

The present report describes how to use a combination of currently standardized technologies to securely effect this authorization.

We also discuss how the general approach of the original SWORN protocol can be extended to cover additional use cases and implementation environments.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-bormann-t2trg-sworn/>.

Discussion of this document takes place on the Thing-to-Thing (T2TRG) Research Group mailing list (<mailto:t2trg@irtf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/t2trg/>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Draft

SWORN: Secure Wake on Radio Nudging

February 2022

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | | |
|------------------------|---|-------------------|
| 1. | Introduction | 3 |
| 1.1. | Terminology | 3 |
| 2. | The original SWORN proposal | 3 |
| 2.1. | Assumptions and Requirements | 3 |
| 2.2. | Security goals | 4 |
| 2.3. | Mechanism | 4 |
| 2.3.1. | Wake-Grant | 5 |
| 2.3.2. | Wake-Token | 5 |
| 2.3.3. | Finding the wake token | 5 |
| 3. | Generalizing SWORN towards Token-Based In-Network Authorization | 6 |
| 3.1. | Position of router R | 7 |
| 3.2. | Position of Token in Packet | 8 |
| 3.3. | Range of Checking | 8 |
| 3.4. | Information (attributes) to be used on the way | 9 |

| | | |
|------------------------|---------------------------------------|--------------------|
| 3.5. | Example arrangements | 10 |
| 3.5.1. | Multiple Independent Grants | 11 |
| 3.5.2. | Hierarchical Grants | 13 |
| 4. | IANA Considerations | 15 |
| 4.1. | Original SWORN model | 15 |

| | | |
|----------------------|-----------------------------------|--------------------|
| 4.2. | Generalized model | 16 |
| 5. | Security Considerations | 16 |
| 6. | References | 16 |
| 6.1. | Normative References | 16 |
| 6.2. | Informative References | 17 |
| | Acknowledgements | 18 |
| | Authors' Addresses | 18 |

[1.](#) Introduction

(See Abstract.)

[1.1.](#) Terminology

The term "byte" is used in its now customary sense as a synonym for "octet".

Messages defined in this document employ CBOR [[RFC8949](#)] and are described in CDDL [[RFC8610](#)].

Terms used in this draft:

C: Client, or Correspondent host. The node that wants to effect "Wake on Radio" on D by sending a message to D.

D: Device. This is typically battery operated and "Normally off" [[RFC7228](#)].

R: Router. The router that is adjacent to D, sharing an energy-saving link with D, and serving as a ("parent") router to D.

MAC: Message Authentication Code (when discussing authentication mechanisms)

MAC: Media Access Control (when discussing protocol layers)

[2.](#) The original SWORN proposal

[2.1.](#) Assumptions and Requirements

D is a normally off [[RFC7228](#)] device, waking up very briefly to communicate with its first hop router R. R and D share a MAC layer that allows R to keep D in extended wake periods.

R and D have a security association. (This may have been created in network onboarding, or be setup dynamically from the device-to-network security association when D chose R as a parent router.)

D wants to authorize a client (or correspondent host) C to ask R to initiate wake periods in D.

Because of changes in the radio environment, D needs to be able to change its parent router from R1 to R2 occasionally. This should not cause a need to notify all its clients; which parent router is used by D is therefore opaque to its clients as usual in IP.

[2.2.](#) Security goals

Since packets with wake tokens are kept in R for extended periods, the limited size buffer provided in R for this is a resource that needs to be protected to protect availability.

D uses up battery for a wake period, which would make it susceptible to battery depletion attacks. To protect availability, D should only undergo wake periods that R has commanded based on previous authorization by D.

There may be confidentiality requirements (e.g., for privacy); this is not addressed in the present version of this report.

[2.3.](#) Mechanism

```
+-----+ . +-----+
| Client   +-- 0 Share some context\ . | Device   |
| Authorization +----- . \_ | Authorization +
| Manager CAM |          \    | Manager DAM   |
+-----+-----+          \    +-----+-----+
```

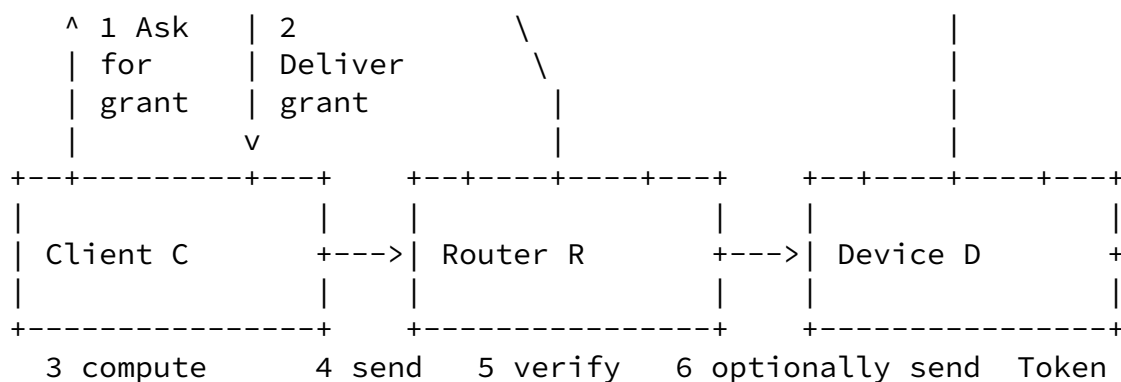


Figure 1: Illustration of a potential setup with Authorization Managers

[2.3.1.](#) Wake-Grant

A wake grant is a CWE [RFC8152], packaging a grant key, provided from D or D's authorization manager to C. (Possibly the grant key can be conveyed within a larger confidentiality protected data structure or channel, such as a CWT [RFC8392] employing a cnf claim for the key [RFC8747].)

A wake grant may then be used by C for initiating (a possibly limited number or total duration of) wake periods, employing Wake-Tokens.

Information about the wake grant is also made available to R, so it knows the grant key and the parameters of the wake grant. (Upon a change of parent router, D will need to make that information available to its new parent router as well.)

[2.3.2.](#) Wake-Token

A wake token is a CWS, in a COSE_MAC0 [RFC8152] message built with the Wake-Grant's key, containing a CBOR data item of the form:

```
[serial: uint, wake-period: duration]
```

The CWS is additionally marked by tagging it with a CBOR tag 1398230866 (a value that becomes visible in a packet dump as ASCII "SWOR").

(Discussion: Should this be a CWE for confidentiality?)

The serial is used for replay detection, based on the usual window mechanism. Wake-Tokens for a fresh wake grant start out with serial numbers at zero.

A Wake-Token instructs R to use MAC mechanisms to provide an extended wake period to D the next time it wakes up.

The wake token is sent from C to D; R finds it by examining packets that it would need to forward to D.

[2.3.3.](#) Finding the wake token

As C is addressing D with the wake token, R needs to find it in traffic purportedly for D.

As described in [[I-D.bormann-intarea-alfi](#)], this cannot be reasonably done with IP options (which originally would have carried this kind of information in the IP architecture).

Instead, R finds the wake token by deep packet inspection. The wake token is found by a heuristic that may have false positives; this is not a problem as the wake token is then verified by its MAC.

SWORN requests are carried in UDP packets that also may have a payload function. To this end, they are conveyed as CoAP messages [[RFC7252](#)]. The wake token is carried in a CoAP option, Wake-Token. R can find the option by decoding the CoAP packet in the UDP payload or simply by scanning for the 5-byte signature 0xda53574f52 created by the CBOR wake token tag. Any potential wake token so found is then validated as a CWS.

This works well with [[RFC8613](#)] as the CoAP security mechanism for any payload function that this packet may have. To be able to use DTLS as well, we define a media type "application/dtls-payload" that can

be used in a CoAP POST request to send a DTLS payload as payload of a CoAP message (in other words, the CoAP POST request carries a Wake-Token and a Content-Format option). (Any return packet can be similarly sent back in the POST response.) (TODO: This media type has to define the port number juggling needed.)

[3.](#) Generalizing SWORN towards Token-Based In-Network Authorization

The original SWORN protocol described so far was designed to solve a specific use case in a specific implementation environment.

We can open up the design space in a number of dimensions, which will be discussed in the following subsections.

The general idea of SWORN can be described as:

- * giving a packet sender a way to send authenticating information with the packet,
- * that is not intended for the recipient of the packet, but instead
- * to be checked at particular enforcement points on the way,
- * which are not necessarily known by the sender,
- * to derive authorization to forward.

Generalizing the terms used so far, we can identify the following players and components:

- * Packets P that require authorization (more generally, the reliable derivation of attributes) by entities on their way,

- * a sender (C) and a recipient (D), where the packets P go from C to D,
- * SWORN policy enforcement points (SPEPs) that are in the network in places where the generalized SWORN authorization of and derivation of attributes for the packets is performed; the ones that are active for a packet P are called R, generalizing the concept of the last-hop router R in the SWORN model,

- * the "grant" G that provides setup information for the SPEP (wake grant in SWORN), and
- * the "token" T that is sent with a specific packet P to carry information that will be checked by R/SPEP (wake token in SWORN).

This model can accommodate additional entities, "authorization managers" (AM), that pair with C (CAM) and D (DAM) for purposes of creating setup information and potentially distributing it among C and D and to network elements that might play the role of R. The distribution may be further facilitated by adding Router AMs (RAM).

The roles of C and D can also be played by tunnel ingress/egress points; this can enable the use of unmodified client and device implementations (note that D, if a DAM is used, need not implement anything special at all, but can of course benefit from information in the token).

[3.1.](#) Position of router R

The original SWORN protocol is based on a 6LoWPAN-like environment where a host (D) has a defined relationship (often including a MAC layer security association) to the last-hop router(s) (R) supplying it with packets. This enables D to provide R with information about wake grants in a secure way, or to delegate this to its authorization manager.

Within a limited domain [[RFC8799](#)], routers that act as SWORN policy enforcement points (SPEPs) may be known by configuration. Security associations from D's authorization manager (DAM) to each such PEP may be set up already, providing a way for DAM to broadcast policy information to all such SPEPs. This enables policy distribution without any involvement from D.

Alternatively, C can have a security association with the SPEPs (possibly indirectly through a client authorization manager CAM) and can send special SWORN probe packets towards D that carry grant information that the SPEPs on the way can extract and copy into their local (soft?) state.

A SPEP could also react to tokens it does not understand by asking

authorization managers for the applicable grant; this may be facilitated by identifying information in a token such as a "key ID". It remains a quality of implementation issue whether the packet can be buffered while the relevant grant is obtained. (On-demand retrieval of grants also adds an obvious attack vector.)

[3.2.](#) Position of Token in Packet

The original SWORN protocol goes to some lengths to avoid the need for C to influence properties below the application layer of the packets it sends.

However, routers may be able to provide more efficient implementations of SWORN when the token information is in a header. So if C has a platform API that could create/influence such a header, this could be used instead.

While penultimate hop processing is in use in a number of architectures (see, e.g., [Section 3.16 of \[RFC3031\]](#) or [\[I-D.ietf-bier-php\]](#)), it is not currently part of the IPv6 architecture, so there is no obvious IPv6 header that could carry this information.

[3.3.](#) Range of Checking

Tokens in the original SWORN protocol only check a serial number; attackers might be able to extract such a token from a packet, suppress the packet, and use the token in a different packet. We call this form of attack "malicious reuse". Replay protection cannot prevent this.

One way to reduce this attack vector is to make the grant very specific to only a certain kind of packets, for instance by effectively including an ACL (carrying, say, elements of the 5-tuple) with the grant. (The grant could also include quantitative aspects, such as a rate limit.) The cryptographic makeup of the grant then must be such that a client cannot modify the ACL during generation of a token from the grant.

Alternatively, some kinds of malicious reuse can be made harder to perform by including more information in the signing input of the MAC checked for a token. If a wake grant is shared between multiple instances of D and C and therefore can only have a weak ACL, including the source and destination addresses (IP address and possibly port) in the signing input for a specific token T prevents reuse of T for a different D/C pair.

Including the payload in the signing input (i.e., essentially authenticating the entire packet) makes malicious reuse of tokens less useful for an attacker as no new information can be injected by it, but requires considerable processing power in the SPEP. We assume that any authentication of the whole packet will most likely be performed by its recipient D, based on a security association it has with the sender C; R or any other SPEPs are not burdened with this authentication. (However, some authentication offload in a last-hop router R may be desirable for a very constrained device D, at least if R and D have a robust security association that can provide the level of authentication needed.)

Most likely, a token should include some information about what is included in the signing input. (This information itself should then also be included in the signing input.) This coverage information could take the form of a bitmap, or of a set of offset-length pairs. If coverage information and covered information is in a form that can be processed by a network processor, higher processing and forwarding rates can be achieved.

3.4. Information (attributes) to be used on the way

The original SWORN protocol provides one attribute beyond the authentication itself: a wake period to be used in the MAC layer that connects R and D. This can be generalized to other parameters that control routing and forwarding, e.g., information that would be equivalent to a DSCP.

An ambitious use case might be a path from a sender to a recipient that crosses multiple domains, where each domain boundary normally bleaches DSCP information coming in from the previous domain.

The domain ingress router might check the SWORN information to be able to apply a policy not to bleach, or to install DSCP information specific to the domain it belongs to, or to operate on some DSCP-like information outside the DSCP field.

Note that the model used by this extended version of SWORN is based on an assumption that it is in the interest of all players to cooperate in achieving the objectives. Security is used to ensure that only actually authorized players can participate. But, in general, the assumed trust and incentive models work best within a limited domain [[RFC8799](#)], which spans one or maybe two security domains only.

[3.5.](#) Example arrangements

This subsection discusses a number of example arrangements with more complex relationships between the players. Once the set of clients starts to scale, the issue of key setup becomes significant. In the best case, the SPEPs do not need to be provisioned (set up) separately for each separate client authorization.

The arrangements presented here are based on symmetric cryptography, specifically MACs (message authentication codes). The MACs are based on grant keys GK shared between the origin or a packet (or of a grant key, see below) and the enforcement point. Several grant keys may need to be employed; they are differentiated in the names of variables defined in this section by using suffixes such as `_z` for authorization and `_n` for authentication, or `_1` and `_2` for consecutive steps.

In these arrangements, grant keys are managed by "authorization managers" (AM, compare Figure 1); we simplify the discussion by mentioning only the client authorization manager (CAM), which actually may be in contact with other authorization managers closer to the SPEPs. Some grant keys are shared only between CAM and the SPEPs, requiring the CAM to send a pre-computed MAC to the client; some grant keys are shared between CAM, Client C, and the SPEPs, allowing the client to compute MACs covering variable parts of the packets.

The MACs of the example arrangements employ a mechanism to indicate which parts of the packet go into the signing input for the MAC, the `_coverage area indication_`, or `cai` for short, to be included in each actual packet. The realization for this could contain a bitmap with bits that enable individual predefined fields (e.g., elements of the 5-tuple), plus pointer/length information for the payload or other dynamic attributes that need to be checked. As discussed above, the coverage area indication itself should also go into the signing input for the MAC.

The notation `"fields(cai_x)"` stands for the actual content of the fields in the packet as enumerated by the coverage area indication

cai_x. If these fields are constant for the authorization conveyed by a grant key (as is often required by an ACL), they can be included in the derivation of the grant key; otherwise they are included in the computation of the MAC keyed by the grant key.

Multiple MACs (based on a cai per MAC and separate GK) can be used in each packet. This can be realized by including all of these MACs in a packet (possibly by simply XORing them), or by building one of the MACs into the signing input of another MAC.

We make the simplifying assumption that the IP address of the client can be used as a client ID; this is always available in the 5-tuple of the packet and can be included in a MAC via the coverage area indication. We use Cx (C1, C2, ...) to discuss a specific client.

Based on these common mechanisms, we discuss two arrangements in the rest of this section. Other arrangements can be built; the information in the grants should enable the indication of the exact structure of the arrangement, enabling the AM to determine which specific arrangement is in use.

[3.5.1.](#) Multiple Independent Grants

In this example arrangement, the AM generates two different kinds of grants, GK_z for authorization checking over a range of clients, and GK_n_x that focuses only on authenticating the packet as originating from a specific client Cx.

The corresponding MACs are used as follows: MAC_z contains information that the AM wants to define, i.e. AM sets the values of these fields; therefore MAC_z can be pre-computed by the AM. MAC_n also contains information that the AM wants Cx to define, i.e. AM identifies the fields (sets cai) and lets Cx set the values of these fields. (As an example, the wake_period of original SWORN would be in MAC_n.)

GK_z is not provided to the clients, instead the CAM provides the client with a ready-made token complete with a MAC, called MAC_z, based on the grant key. This token indicates authorization for its coverage area (typically elements of the 5-tuple, plus possibly some more dynamic attributes). It is pre-computed for each client Cx and conveyed to Cx from the CAM; as there is no replay protection of the

authorization, this token can be used essentially as a bearer token, until dynamic attributes or keys need to be updated (then a new authorization token needs to be sent from CAM to each Cx that shall continue to communicate).

The SPEP also has GK_z and can check the tokens based on the information in the packet (and possibly cache e.g. MAC_z for the flow, if not pre-computed from a table of client IDs).

This approach alone does not protect against replay attacks. For this, we introduce GK_n_x in corresponding client grants, carrying a client-specific authentication key. Each GK_n_x is provided by the CAM to both an individual Cx and (indirectly via GK_z) to the SPEPs. The authentication key is used against a selected subset of the five-tuple, the dynamic attributes, and the payload. It effectively proves that the holder of this key, Cx, is the actual originator of the packet. It does not prove authorization of a specific five-tuple plus attributes; this is the job of the GK_z/MAC_z.

Each packet carries MAC_z based on the shared grant key (bearer token supplied by CAM for a specific selection of elements from the 5-tuple, proving that this communication with this selection of 5-tuple elements is generally authorized.) A typical selection from the 5-tuple would comprise the source (client) and destination (device) IP addresses as well as the destination port number and protocol; the source port number would remain dynamic to be chosen by the client in this case. (MAC_z could also cover further attributes relevant to the authorization, such as an application-id or performance parameters such as intended maximum bitrate and traffic priority; some of these attributes could be included in the token, others could be derived from policy and an attribute such as an application-id.)

We assume that the mechanisms discussed here need to provide source

authentication for the packets from Cx. If a separate source authentication mechanism is available, GK_n_x and MAC_n are not needed. Otherwise, for authentication, each packet also carries MAC_n based on GK_n_x, which proves the actual origin of the packet; the authorization that applies to the packet needs to be checked via the authorization token. The coverage area of MAC_n is indicated by cai_n; this will generally be a superset of cai_z and will include the source port number as well, so this can be freely chosen by the client but is still authenticated.

The payload may or may not be included into the signing input to MAC_n (this is also indicated by cai_n). If the payload is included, this ensures that an attacker can only replay completely identical packets; in situations where this kind of replay would not be a problem, no replay window mechanism needs to be employed.

In summary (bracket notation [..., ...] stands for an array of values):

```
MAC_z = MAC([cai_z, fields(cai_z), attributes, cai_n, timestamp],
            GK_z)
```

```
MAC_n = MAC([serial, cai_n, fields(cai_n)], GK_n_x)
```

where the CAM provides to Cx:

- * GK_n_x = KDF(["key for MAC_n", CxID, GK_z])
- * all other inputs to MAC_z except for timestamp
- * MAC_z

The actual information in the packet contains the token:

```
Token =
    [cai_z, cai_n, GK_z-kid, CxID, attributes_conveyed,
     (MAC_z xor MAC_n_x)]
```

The provisioning needed is limited to distributing grant keys GK_z for each specific authorization (e.g., Device IP address/port number) plus a base key GK_n_x to derive authentication keys for each client Cx. These are connected by the common parts of the 5-tuple and a

CxID.

The role of the different keys can be summarized as follows:

- * GK_z, known only by AM and the SPEPs, allows the AM to specify the authorization information, which then can be checked by the SPEP using MAC_z under the key GK_z.

Since the AM can compute the MAC_z ahead of time, from the point of view of the client MAC_z essentially serves as a bearer token, which however is only useful for packets with the fields(cai_z) that are being authorized by AM.

- * GK_{n_x} provides client x with a way to authenticate a wider coverage, including values of the client's choice such as the source port number or even the payload. Here, client x needs to have the key so it can dynamically vary these fields without needing to recur to AM.

3.5.2. Hierarchical Grants

[Section 3.5.1](#) requires distributing separate authentication grants for each client Cx to each SPEP, which may however be orthogonal to the specific authorizations granted.

We cannot completely eliminate per-client "provisioning" as SPEP needs to know about the authorization it needs to enforce. However, by giving clients (e.g., C1/C2) more specific information than SPEP, we can relieve SPEP from needing to know each of C1/C2 a priori, as follows:

Let GK₀ be the _generic grant Key_, which is generated by CAM and told to the SPEPs in a generic grant. Clients never obtain GK₀.

Let GK_x be a _specific grant Key_, which is generated by CAM for each Cx and told to Cx in a _specific grant_. Client x only knows its own GK_x, not that of other clients Cy.

For a complete exposition, we also introduce cai₁ and cai₂:

- * Both cai₁ and fields(cai₁) are shared between CAM and SPEP in a generic grant; fields(cai₁) are the constant parts of the packets

to be authorized (e.g., device IP address, device port number).

- * `cai_2` is a superset of `cai_1`; it indicates the fields that need to be checked by SPEP. The generic grant provides `cai_2`, but not `fields(cai_2)`, as these can differ per packet and are chosen by Cx.

Ignoring nonces, serials, and key identifier/lifetime/rollover mechanisms, the following computations can be done:

- * by CAM or SPEP (who both know `GK0` and `cai_1`, i.e. generic information about all packets covered by this grant):

`Grant_x = [GK0-kid, cai_1, cai_2, fields(cai_1), GKx]`

where `GKx = KDF([cai_1, cai_2, fields(cai_1), GK0])`

- * by Cx based on its provisioned `GKx` and a specific packet, but also by SPEP based on the `GKx` the SPEP can compute for Cx

`token = [GK0-kid, cai_1, cai_2, MAC([serial, cai_2, fields(cai_2)]), GKx]`

(These computations do not explicitly mention a client ID, which is implied by the source IP address included in `fields(cai_n_)`.)

Note that the token can be verified by SPEP as it can compute `GKx` from the secret key `GK0` that is only shared between CAM and SPEP, as long as it knows which `GK0` to apply (e.g., using `GK0-kid`, which here simply is copied into the token). As they don't know `GK0`, the

individual clients Cx cannot compute GKx -- they need to obtain it from CAM as part of Grant_x, which also gives them cai_1 and cai_2 -- containing the information for Cx how to apply the GKx. The token, computed from GKx, therefore authenticates Cx as the holder of xGKx, and signifies the authorization of Cx to send a packet with fields(cai_1) as indicated in GKx, with the additional requirement on Cx that it also authenticates a serial number (used for replay protection) and additional fields(cai_2).

Since the token includes the relevant parts of the 5-tuple, and cai_1 is part of the first MAC and therefore cannot be changed by the Cxes, there is no need to apply any additional ACL beyond feeding the cai-selected fields of the packet into the MAC.

So GK0 only needs to be provisioned once to the SPEPs for all Cxes; each GKx is provisioned only to Cx. An SPEG can compute GKx from cai_1, cai_2 (part of the token) and fields(cai_1) (taken from the packet). GKx is eminently cacheable, but in any case does not need to be provisioned as it can be derived from GK0 and information that is in the packet.

C1 and C2 have GK1 and GK2, respectively; the coverage area is easily extended to include D1-address and D1-port etc., by setting the bits in either cai_1 (if these are fixed for Cx) or cai_2 (if these can be chosen by Cx).

[4.](#) IANA Considerations

[4.1.](#) Original SWORN model

Define CBOR Wake-Token Tag 1398230866 in [[IANA.cbor-tags](#)].

Define CoAP option Wake-Token in the CoAP Option Numbers Registry of [[IANA.core-parameters](#)] ([Section 12.2 of \[RFC7252\]](#)). (The option is safe, no-cache-key, elective, repeatable, of type opaque 0-255 bytes.)

Define media-type "application/dtls-payload", with an associated CoAP Content-Format in the CoAP Content-Formats Registry of [[IANA.core-parameters](#)] ([Section 12.3 of \[RFC7252\]](#)).

[4.2.](#) Generalized model

TBD

[5.](#) Security Considerations

The purpose of the security mechanisms described is primarily to protect availability (obviously, any symmetric keys employed also need to be confidentiality protected for the sake of the integrity of the mechanism). For the purposes of this kind of availability protection, occasional false positives of the per-packet authorization mechanisms may be acceptable, as long as they don't reach a threshold of probability of success that is application dependent (say, success in one out of a million of brute force attempts, equivalent to 20-bit security). This may offer optimization opportunities that need further study.

TBD

[6.](#) References

[6.1.](#) Normative References

[IANA.cbor-tags]

IANA, "Concise Binary Object Representation (CBOR) Tags",
<<https://www.iana.org/assignments/cbor-tags>>.

[IANA.core-parameters]

IANA, "Constrained RESTful Environments (CoRE) Parameters",
<<https://www.iana.org/assignments/core-parameters>>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", [RFC 7252](#), DOI 10.17487/RFC7252, June 2014,
<<https://www.rfc-editor.org/info/rfc7252>>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", [RFC 8152](#), DOI 10.17487/RFC8152, July 2017,
<<https://www.rfc-editor.org/info/rfc8152>>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", [RFC 8392](#), DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.

Internet-Draft

SWORN: Secure Wake on Radio Nudging

February 2022

- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", [RFC 8610](#), DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", [RFC 8747](#), DOI 10.17487/RFC8747, March 2020, <<https://www.rfc-editor.org/info/rfc8747>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, [RFC 8949](#), DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

[6.2.](#) Informative References

- [I-D.bormann-intarea-alfi] Bormann, C., "Adaptation Layer Fragmentation Indication", Work in Progress, Internet-Draft, [draft-bormann-intarea-alfi-04](#), 9 September 2013, <<https://www.ietf.org/archive/id/draft-bormann-intarea-alfi-04.txt>>.
- [I-D.ietf-bier-php] Zhang, Z., "BIER Penultimate Hop Popping", Work in Progress, Internet-Draft, [draft-ietf-bier-php-07](#), 7 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-bier-php-07.txt>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", [RFC 3031](#), DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", [RFC 7228](#), DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", [RFC 8613](#), DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", [RFC 8799](#), DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.

Acknowledgements

☞☞ (Bo Chen) provided input for [Section 3](#).

TBD

Authors' Addresses

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

