Internet Engineering Task Force (IETF)
Request for Comments: 7519
Category: Standards Track
ISSN: 2070-1721

M. Jones
Microsoft
J. Bradley
Ping Identity
N. Sakimura
NRI

May 2015

JSON Web Token (JWT)

Abstract

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

JWT(JSON Web Token)는 두 개체 간에 전송되는 클레임을 나타내는
간결하고 URL 안전한 수단입니다.
JWT의 클레임은 JSON 객체로 인코딩되며,
이는 JSON 웹 서명(JWS) 구조의 페이로드로 사용되거나
JSON 웹 암호화(JWE) 구조의 평문으로 사용됩니다.
이를 통해 클레임은 디지털 서명 또는 메시지 인증 코드(MAC)로
디지털 서명되거나 무결성이 보호될 수 있습니다.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7519.

이 문서는 인터넷 표준 추적 문서입니다.

이 문서는 인터넷 공학 태스크 포스(IETF)의 제품입니다.
이는 IETF 커뮤니티의 합의를 반영합니다.
이 문서는 공개 검토를 받았으며 인터넷 공학 지도 그룹(IESG)에 의해 게시 승인을 받았습니다.
인터넷 표준에 대한 자세한 정보는 RFC 5741의 2절에서 확인할 수 있습니다.

이 문서의 현재 상태, 모든 정정표 및 피드백 제공 방법에 대한 정보는
http://www.rfc-editor.org/info/rfc7519에서 확인할 수 있습니다.

Table of Contents

1. Introduction 4
1.1. Notational Conventions 4
2. Terminology 4
3. JSON Web Token (JWT) Overview 6
3.1. Example JWT 7
4. JWT Claims 8
4.1. Registered Claim Names 9
4.1.1. "iss" (Issuer) Claim 9
4.1.2. "sub" (Subject) Claim 9
4.1.3. "aud" (Audience) Claim 9
4.1.4. "exp" (Expiration Time) Claim 9
4.1.5. "nbf" (Not Before) Claim 10
4.1.6. "iat" (Issued At) Claim 10
4.1.7. "jti" (JWT ID) Claim 10
4.2. Public Claim Names 10
4.3. Private Claim Names 10
5. JOSE Header 11
5.1. "typ" (Type) Header Parameter 11
5.2. "cty" (Content Type) Header Parameter 11
5.3. Replicating Claims as Header Parameters 12
6. Unsecured JWTs 12
6.1. Example Unsecured JWT 12
7. Creating and Validating JWTs 13
7.1. Creating a JWT 13
7.2. Validating a JWT 14
7.3. String Comparison Rules 15
8. Implementation Requirements 16
9. URI for Declaring that Content is a JWT 17
10. IANA Considerations 17

10.1. JSON Web Token Claims Registry 17
10.1.1. Registration Template 18
10.1.2. Initial Registry Contents 18
10.2. Sub-Namespace Registration of
urn:ietf:params:oauth:token-type:jwt 19
10.2.1. Registry Contents 19
10.3. Media Type Registration 20
10.3.1. Registry Contents 20
10.4. Header Parameter Names Registration 20
10.4.1. Registry Contents 21
11. Security Considerations 21
11.1. Trust Decisions 21
11.2. Signing and Encryption Order 21
12. Privacy Considerations 22
13. References 22
13.1. Normative References 22
13.2. Informative References 23
Appendix A. JWT Examples 26
A.1. Example Encrypted JWT 26
A.2. Example Nested JWT 26
Appendix B. Relationship of JWTs to SAML Assertions 28
Appendix C. Relationship of JWTs to Simple Web Tokens (SWTs) . . 28
Acknowledgements 28
Authors' Addresses 29

---

1. Introduction

JSON Web Token (JWT) is a compact claims representation format intended for space constrained environments such as HTTP Authorization headers and URI query parameters. JWTs encode claims to be transmitted as a JSON [RFC7159] object that is used as the payload of a JSON Web Signature (JWS) [JWS] structure or as the plaintext of a JSON Web Encryption (JWE) [JWE] structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted. JWTs are always represented using the JWS Compact Serialization or the JWE Compact Serialization.

The suggested pronunciation of JWT is the same as the English word "jot".

JSON Web Token (JWT)는 HTTP 인증 헤더 및 URI 쿼리 매개변수와 같은 공간이 제한된 환경을 위한 간결한 클레임 표현 형식입니다.
JWT는 클레임을 전송하기 위해 JSON [RFC7159] 객체로 인코딩되며,
이는 JSON 웹 서명(JWS) [JWS] 구조의 페이로드로 사용되거나 JSON 웹 암호화(JWE) [JWE] 구조의 평문으로 사용됩니다.
이를 통해 클레임은 디지털 서명 또는 메시지 인증 코드(MAC) 및/또는 암호화로 무결성이 보호될 수 있습니다.
JWT는 항상 JWS Compact Serialization 또는 JWE Compact Serialization을 사용하여 표현됩니다.

JWT의 권장 발음은 영어 단어 "jot"과 동일합니다.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119]. The interpretation should only be applied when the terms appear in all capital letters.

2. Terminology

The terms "JSON Web Signature (JWS)", "Base64url Encoding", "Header Parameter", "JOSE Header", "JWS Compact Serialization", "JWS Payload", "JWS Signature", and "Unsecured JWS" are defined by the JWS specification [JWS].

The terms "JSON Web Encryption (JWE)", "Content Encryption Key (CEK)", "JWE Compact Serialization", "JWE Encrypted Key", and "JWE Initialization Vector" are defined by the JWE specification [JWE].

The terms "Ciphertext", "Digital Signature", "Message Authentication Code (MAC)", and "Plaintext" are defined by the "Internet Security Glossary, Version 2" [RFC4949].

"JSON Web Signature (JWS)", "Base64url Encoding", "Header Parameter", "JOSE Header", "JWS Compact Serialization", "JWS Payload", "JWS Signature", 및 "Unsecured JWS" 용어는 JWS 사양 [JWS]에 의해 정의됩니다.

"JSON Web Encryption (JWE)", "Content Encryption Key (CEK)", "JWE Compact Serialization", "JWE Encrypted Key", 및 "JWE Initialization Vector" 용어는 JWE 사양 [JWE]에 의해 정의됩니다.

"Ciphertext", "Digital Signature", "Message Authentication Code (MAC)", 및 "Plaintext" 용어는 "인터넷 보안 용어집, 버전 2" [RFC4949]에서 정의됩니다.

These terms are defined by this specification:

JSON Web Token (JWT)
A string representing a set of claims as a JSON object that is encoded in a JWS or JWE, enabling the claims to be digitally signed or MACed and/or encrypted.

JSON Web Token (JWT)은 클레임 집합을 나타내는 문자열로, 이는 JWS 또는 JWE에 인코딩된 JSON 객체로 표현됩니다. 이를 통해 클레임은 디지털 서명이나 MAC 또는 암호화로 보호될 수 있습니다.

JWT Claims Set
A JSON object that contains the claims conveyed by the JWT.

JWT에 의해 전달된 클레임을 포함하는 JSON 객체입니다.

Claim
A piece of information asserted about a subject. A claim is represented as a name/value pair consisting of a Claim Name and a Claim Value.

주체에 대해 주장된 정보입니다.
클레임은 클레임 이름과 클레임 값으로 구성된 이름/값 쌍으로 표현됩니다.

Claim Name
The name portion of a claim representation. A Claim Name is always a string.

클레임 표현의 이름 부분입니다.
클레임 이름은 항상 문자열입니다.

Claim Value
The value portion of a claim representation. A Claim Value can be any JSON value.

클레임 표현의 값 부분입니다.
클레임 값은 어떤 JSON 값이든 될 수 있습니다.

Nested JWT
A JWT in which nested signing and/or encryption are employed. In Nested JWTs, a JWT is used as the payload or plaintext value of an enclosing JWS or JWE structure, respectively.

중첩된 서명 및/또는 암호화가 사용된 JWT입니다.
중첩된 JWT에서 JWT는 각각 포함된 JWS
또는 JWE 구조의 페이로드 또는 평문 값으로 사용됩니다.

Unsecured JWT
A JWT whose claims are not integrity protected or encrypted.

클레임이 무결성 보호되지 않거나 암호화되지 않은 JWT입니다.

Collision-Resistant Name
A name in a namespace that enables names to be allocated in a manner such that they are highly unlikely to collide with other names. Examples of collision-resistant namespaces include: Domain Names, Object Identifiers (OIDs) as defined in the ITU-T X.660 and X.670 Recommendation series, and Universally Unique IDentifiers (UUIDs) [RFC4122]. When using an administratively delegated namespace, the definer of a name needs to take reasonable precautions to ensure they are in control of the portion of the namespace they use to define the name.

충돌 방지 이름은 이름을 다른 이름과 충돌할 가능성이 매우 낮도록 할 수 있는
네임스페이스의 이름입니다.
충돌 방지 네임스페이스의 예로는 도메인 이름, ITU-T X.660 및
X.670 권장 시리즈에서 정의된 객체 식별자(OID),
그리고 RFC4122에 정의된 범용 고유 식별자(UUID)가 있습니다.

관리적으로 위임된 네임스페이스를 사용할 때,
이름을 정의하는 사용자는 이름을 정의하는 데 사용되는 네임스페이스 부분을 통제하기 위해
합리적인 주의를 기울여야 합니다.

StringOrURI
A JSON string value, with the additional requirement that while arbitrary string values MAY be used, any value containing a ":" character MUST be a URI [RFC3986]. StringOrURI values are compared as case-sensitive strings with no transformations or canonicalizations applied.

JSON 문자열 값으로, 임의의 문자열 값이 사용될 수 있지만, ":" 문자를 포함하는 모든 값은 반드시 URI여야 합니다 [RFC3986]. StringOrURI 값은 대소문자를 구분하여 문자열로 비교되며, 변환 또는 표준화가 적용되지 않습니다.

NumericDate
A JSON numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds. This is equivalent to the IEEE Std 1003.1, 2013 Edition [POSIX.1] definition "Seconds Since the Epoch", in which each day is accounted for by exactly 86400 seconds, other than that non-integer values can be represented. See RFC 3339 [RFC3339] for details regarding date/times in general and UTC in particular.

지정된 UTC 날짜/시간부터 1970년 1월 1일 00:00:00Z UTC까지의 초 수를 나타내는 JSON 숫자 값입니다.
이는 윤초를 무시하며, 각 날은 정확히 86400초로 계산되는 IEEE Std 1003.1, 2013 Edition [POSIX.1]의
"에포크부터의 초" 정의와 동등합니다.

이 정의에서는 정수가 아닌 값도 표현할 수 있습니다.
RFC 3339 [RFC3339]에서는 일반적인 날짜/시간 및 특히 UTC에 대한 자세한 내용을 참조하십시오.

3. JSON Web Token (JWT) Overview

JWTs represent a set of claims as a JSON object that is encoded in a JWS and/or JWE structure. This JSON object is the JWT Claims Set. As per Section 4 of RFC 7159 [RFC7159], the JSON object consists of zero or more name/value pairs (or members), where the names are strings and the values are arbitrary JSON values. These members are the claims represented by the JWT. This JSON object MAY contain whitespace and/or line breaks before or after any JSON values or structural characters, in accordance with Section 2 of RFC 7159 [RFC7159].

JWT는 JWS 및/또는 JWE 구조에 인코딩된 JSON 객체로서 클레임 집합을 나타냅니다.
이 JSON 객체는 JWT 클레임 집합입니다.
RFC 7159 [RFC7159]의 섹션 4에 따르면,
이 JSON 객체는 이름이 문자열이고 값이 임의의 JSON 값인 제로 개 이상의 이름/값 쌍(또는 멤버)으로 구성됩니다.
이러한 멤버들은 JWT에 의해 표현된 클레임입니다.
이 JSON 객체는 RFC 7159 [RFC7159]의 섹션 2에 따라,

임의의 JSON 값 또는 구조적 문자 앞 또는 뒤에 공백 및/또는 줄 바꿈을 포함할 수 있습니다.

The member names within the JWT Claims Set are referred to as Claim Names. The corresponding values are referred to as Claim Values.

JWT 클레임 집합 내의 멤버 이름은 클레임 이름으로 참조됩니다.
해당하는 값은 클레임 값으로 참조됩니다.

The contents of the JOSE Header describe the cryptographic operations applied to the JWT Claims Set. If the JOSE Header is for a JWS, the JWT is represented as a JWS and the claims are digitally signed or MACed, with the JWT Claims Set being the JWS Payload. If the JOSE Header is for a JWE, the JWT is represented as a JWE and the claims are encrypted, with the JWT Claims Set being the plaintext encrypted by the JWE. A JWT may be enclosed in another JWE or JWS structure to create a Nested JWT, enabling nested signing and encryption to be performed.

JOSE 헤더의 내용은 JWT 클레임 집합에 적용된 암호화 작업을 설명합니다.
JOSE 헤더가 JWS를 위한 것인 경우, JWT는 JWS로 표현되고
클레임은 디지털 서명되거나 MAC이 적용되며, JWT 클레임 집합은 JWS 페이로드가 됩니다.
JOSE 헤더가 JWE를 위한 것인 경우, JWT는 JWE로 표현되고 클레임은 암호화되며,
JWT 클레임 집합은 JWE에 의해 암호화된 평문이 됩니다.
JWT는 중첩된 서명 및 암호화를 수행하는 중첩된 JWT를 만들기 위해
다른 JWE 또는 JWS 구조에 포함될 수 있습니다.

A JWT is represented as a sequence of URL-safe parts separated by period ('.') characters. Each part contains a base64url-encoded value. The number of parts in the JWT is dependent upon the representation of the resulting JWS using the JWS Compact Serialization or JWE using the JWE Compact Serialization.

JWT는 URL 안전한 부분들의 시퀀스로 표현되며, 이들은 마침표('.') 문자로 구분됩니다.
각 부분은 base64url로 인코딩된 값이 포함되어 있습니다.
JWT의 부분 수는 JWS Compact Serialization 또는 JWE Compact Serialization을 사용하여
결과 JWS의 표현에 따라 달라집니다.

3.1. Example JWT

The following example JOSE Header declares that the encoded object is a JWT, and the JWT is a JWS that is MACed using the HMAC SHA-256 algorithm:

다음 예시의 JOSE 헤더는 인코딩된 객체가 JWT임을 선언하고,
해당 JWT가 HMAC SHA-256 알고리즘을 사용하여 MAC이 적용된 JWS임을 나타냅니다.

```
{"typ":"JWT",
 "alg":"HS256"}
```

To remove potential ambiguities in the representation of the JSON

object above, the octet sequence for the actual UTF-8 representation used in this example for the JOSE Header above is also included below. (Note that ambiguities can arise due to differing platform representations of line breaks (CRLF versus LF), differing spacing at the beginning and ends of lines, whether the last line has a terminating line break or not, and other causes. In the representation used in this example, the first line has no leading or trailing spaces, a CRLF line break (13, 10) occurs between the first and second lines, the second line has one leading space (32) and no trailing spaces, and the last line does not have a terminating line break.) The octets representing the UTF-8 representation of the JOSE Header in this example (using JSON array notation) are:

위의 JSON 객체 표현에서 잠재적인 모호성을 제거하기 위해,
위의 예시에서 사용된 JOSE 헤더의 실제 UTF-8 표현에 대한 옥텟 시퀀스도 아래에 포함되어 있습니다.
(모호성은 줄 바꿈의 다른 플랫폼 표현 (CRLF 대 LF), 줄의 시작과 끝에 다른 공간,
마지막 줄에 종결 줄 바꿈이 있는지 여부 등으로 인해 발생할 수 있습니다.
이 예시에서 사용된 표현에서 첫 번째 줄에는 선행 또는 후행 공백이 없으며,
첫 번째 줄과 두 번째 줄 사이에는 CRLF 줄 바꿈 (13, 10)이 발생하며,
두 번째 줄에는 선행 공백 하나 (32)만 있고 후행 공백이 없으며,
마지막 줄에는 종결 줄 바꿈이 없습니다.)

이 예시에서 JOSE 헤더의 UTF-8 표현을 나타내는 옥텟은 다음과 같습니다 (JSON 배열 표기법 사용):

```
[123, 34, 116, 121, 112, 34, 58, 34, 74, 87, 84, 34, 44, 13, 10, 32,
34, 97, 108, 103, 34, 58, 34, 72, 83, 50, 53, 54, 34, 125]
```

Base64url encoding the octets of the UTF-8 representation of the JOSE Header yields this encoded JOSE Header value:

UTF-8 표현의 옥텟을 Base64url로 인코딩하면
다음과 같은 인코딩된 JOSE 헤더 값이 생성됩니다:

```
  eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
```

The following is an example of a JWT Claims Set:

다음은 JWT 클레임 집합의 예시입니다:

```
  {"iss":"joe",
   "exp":1300819380,
   "http://example.com/is_root":true}
```

The following octet sequence, which is the UTF-8 representation used in this example for the JWT Claims Set above, is the JWS Payload:

다음 옥텟 시퀀스는 위의 JWT 클레임 집합에 사용된 UTF-8 표현으로,
이는 JWS 페이로드입니다.

```
[123, 34, 105, 115, 115, 34, 58, 34, 106, 111, 101, 34, 44, 13, 10,
32, 34, 101, 120, 112, 34, 58, 49, 51, 48, 48, 56, 49, 57, 51, 56,
48, 44, 13, 10, 32, 34, 104, 116, 116, 112, 58, 47, 47, 101, 120, 97,
109, 112, 108, 101, 46, 99, 111, 109, 47, 105, 115, 95, 114, 111,
111, 116, 34, 58, 116, 114, 117, 101, 125]
```

Base64url encoding the JWS Payload yields this encoded JWS Payload (with line breaks for display purposes only):

JWS 페이로드를 Base64url로 인코딩하면
다음과 같은 인코딩된 JWS 페이로드가 생성됩니다
(표시 목적으로만 줄 바꿈 추가):

```
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly
9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

Computing the MAC of the encoded JOSE Header and encoded JWS Payload with the HMAC SHA-256 algorithm and base64url encoding the HMAC value in the manner specified in [JWS] yields this encoded JWS Signature:

[JWS]에서 지정된 방식으로 HMAC SHA-256 알고리즘을 사용하여
인코딩된 JOSE 헤더와 인코딩된 JWS 페이로드의 MAC을 계산하고,
HMAC 값을 base64url로 인코딩하면 다음과 같은 인코딩된 JWS 서명이 생성됩니다:

```
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

Concatenating these encoded parts in this order with period ('.') characters between the parts yields this complete JWT (with line breaks for display purposes only):

이러한 인코딩된 부분들을 위에서부터 이 순서로 마침표 ('.') 문자로 연결하면
다음과 같은 완전한 JWT가 생성됩니다 (표시 목적으로만 줄 바꿈 추가):

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

This computation is illustrated in more detail in Appendix A.1 of [JWS]. See Appendix A.1 for an example of an encrypted JWT.

이 계산은 [JWS]의 부록 A.1에서 더 자세히 설명되어 있습니다.
암호화된 JWT의 예시는 부록 A.1을 참조하십시오.

4. JWT Claims

The JWT Claims Set represents a JSON object whose members are the claims conveyed by the JWT. The Claim Names within a JWT Claims Set MUST be unique; JWT parsers MUST either reject JWTs with duplicate Claim Names or use a JSON parser that returns only the lexically last duplicate member name, as specified in Section 15.12 ("The JSON Object") of ECMAScript 5.1 [ECMAScript].

JWT 클레임 집합은 JWT에 의해 전달된 클레임을 나타내는 멤버들로 구성된 JSON 객체를 나타냅니다.
JWT 클레임 집합 내의 클레임 이름은 고유해야 합니다.

JWT 파서는 중복된 클레임 이름이 있는 JWT를 거부하거나,
ECMAScript 5.1 [ECMAScript]의 섹션 15.12 ("JSON 객체")에서 지정된 대로
렉시컬로 마지막 중복 멤버 이름만 반환하는 JSON 파서를 사용해야 합니다.

The set of claims that a JWT must contain to be considered valid is context dependent and is outside the scope of this specification. Specific applications of JWTs will require implementations to understand and process some claims in particular ways. However, in the absence of such requirements, all claims that are not understood by implementations MUST be ignored.

JWT가 유효하다고 간주되기 위해 포함해야 하는 클레임 집합은 컨텍스트에 따라 달라지며,
이 사양의 범위를 벗어납니다.
JWT의 구체적인 응용 프로그램은 구현이 특정 방식으로 특정 클레임을 이해하고 처리해야 할 수 있습니다.
그러나 이러한 요구 사항이 없는 경우, 구현이 이해하지 못하는 모든 클레임은 무시해야 합니다.

There are three classes of JWT Claim Names: Registered Claim Names, Public Claim Names, and Private Claim Names.

JWT 클레임 이름에는 등록된 클레임 이름, 공개 클레임 이름
및 개인 클레임 이름 세 가지 클래스가 있습니다.

4.1. Registered Claim Names

The following Claim Names are registered in the IANA "JSON Web Token Claims" registry established by Section 10.1. None of the claims defined below are intended to be mandatory to use or implement in all cases, but rather they provide a starting point for a set of useful, interoperable claims. Applications using JWTs should define which specific claims they use and when they are required or optional. All the names are short because a core goal of JWTs is for the representation to be compact.

4.1. 등록된 클레임 이름

다음 클레임 이름들은 섹션 10.1에 의해 설정된 IANA "JSON Web Token Claims"
레지스트리에 등록되어 있습니다.
아래 정의된 클레임 중 어떤 것도 모든 경우에 필수적으로 사용하거나 구현할 의도가 아니라,
대신 유용하고 상호 운용 가능한 클레임 집합의 시작점을 제공합니다.
JWT를 사용하는 응용 프로그램은 사용하는 특정 클레임과 필요한 경우
또는 선택적인 경우를 정의해야 합니다.
모든 이름은 짧습니다.
왜냐하면 JWT의 핵심 목표 중 하나는 표현이 간결해야 하기 때문입니다.

4.1.1. "iss" (Issuer) Claim

The "iss" (issuer) claim identifies the principal that issued the JWT. The processing of this claim is generally application specific.

The "iss" value is a case-sensitive string containing a StringOrURI value. Use of this claim is OPTIONAL.

4.1.1. "iss" (발급자) 클레임

"iss" (발급자) 클레임은 JWT를 발급한 주체를 식별합니다.
이 클레임의 처리는 일반적으로 응용 프로그램별로 구체적입니다.
"iss" 값은 StringOrURI 값을 포함하는 대소문자를 구분하는 문자열입니다.
이 클레임의 사용은 선택 사항입니다.

4.1.2. "sub" (Subject) Claim

The "sub" (subject) claim identifies the principal that is the subject of the JWT. The claims in a JWT are normally statements about the subject. The subject value MUST either be scoped to be locally unique in the context of the issuer or be globally unique. The processing of this claim is generally application specific. The "sub" value is a case-sensitive string containing a StringOrURI value. Use of this claim is OPTIONAL.

4.1.2. "sub" (주체) 클레임

"sub" (주체) 클레임은 JWT의 주제인 주체를 식별합니다.
JWT의 클레임은 일반적으로 주체에 관한 진술입니다.
주제 값은 일반적으로 발급자의 컨텍스트에서 지역적으로 고유하거나 전역적으로 고유해야 합니다.
이 클레임의 처리는 일반적으로 응용 프로그램별로 구체적입니다.
"sub" 값은 StringOrURI 값을 포함하는 대소문자를 구분하는 문자열입니다.
이 클레임의 사용은 선택 사항입니다.

4.1.3. "aud" (Audience) Claim

The "aud" (audience) claim identifies the recipients that the JWT is intended for. Each principal intended to process the JWT MUST identify itself with a value in the audience claim. If the principal processing the claim does not identify itself with a value in the "aud" claim when this claim is present, then the JWT MUST be rejected. In the general case, the "aud" value is an array of case-sensitive strings, each containing a StringOrURI value. In the special case when the JWT has one audience, the "aud" value MAY be a single case-sensitive string containing a StringOrURI value. The interpretation of audience values is generally application specific. Use of this claim is OPTIONAL.

4.1.3. "aud" (대상 청중) 클레임

"aud" (대상 청중) 클레임은 JWT가 대상으로 하는 수령자를 식별합니다.
JWT를 처리하기로 의도된 각 주체는 반드시 대상 청중 클레임에서 값을 지정해야 합니다.
이 클레임이 존재할 때 클레임을 처리하는 주체가 자신을 "aud" 클레임의 값으로 식별하지 않으면,
JWT는 반드시 거부되어야 합니다.
일반적인 경우, "aud" 값은 대소문자를 구분하는 문자열을 포함한 StringOrURI 값을 갖는 배열입니다.
JWT에 대상 청중이 하나만 있는 특별한 경우에는 "aud" 값이 대소문자를 구분하는
문자열 하나를 포함한 StringOrURI 값을 가질 수 있습니다.
청중 값의 해석은 일반적으로 응용 프로그램별로 구체적입니다.
이 클레임의 사용은 선택 사항입니다.

4.1.4. "exp" (Expiration Time) Claim

The "exp" (expiration time) claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing. The processing of the "exp" claim requires that the current date/time MUST be before the expiration date/time listed in the "exp" claim.

4.1.4. "exp" (만료 시간) 클레임

"exp" (만료 시간) 클레임은 JWT가 처리되지 않아야 하는 만료 시간을 식별합니다.
"exp" 클레임의 처리에는 현재 날짜/시간이 "exp" 클레임에 명시된 만료 날짜/시간보다 이전이어야 합니다.

Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value MUST be a number containing a NumericDate value. Use of this claim is OPTIONAL.

구현자는 시계 오차를 고려하여 일반적으로 몇 분 이상이 아닌 작은 여유를 제공할 수 있습니다.
해당 값은 NumericDate 값을 포함하는 숫자여야 합니다.
이 클레임의 사용은 선택 사항입니다.

4.1.5. "nbf" (Not Before) Claim

The "nbf" (not before) claim identifies the time before which the JWT MUST NOT be accepted for processing. The processing of the "nbf" claim requires that the current date/time MUST be after or equal to the not-before date/time listed in the "nbf" claim. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value MUST be a number containing a NumericDate value. Use of this claim is OPTIONAL.

4.1.5. "nbf" (Not Before) 클레임

"nbf" (Not Before, 허용 시작 시간) 클레임은 JWT가 처리되기 전에 허용되지 않아야 하는 시간을 식별합니다.
"nbf" 클레임의 처리에는 현재 날짜/시간이 "nbf" 클레임에 명시된 시작 이전이거나 동일해야 합니다.
구현자는 시계 오차를 고려하여 일반적으로 몇 분 이상이 아닌 작은 여유를 제공할 수 있습니다.
해당 값은 NumericDate 값을 포함하는 숫자여야 합니다.
이 클레임의 사용은 선택 사항입니다.

4.1.6. "iat" (Issued At) Claim

The "iat" (issued at) claim identifies the time at which the JWT was issued. This claim can be used to determine the age of the JWT. Its value MUST be a number containing a NumericDate value. Use of this claim is OPTIONAL.

4.1.6. "iat" (발급 시간) 클레임

"iat" (Issued At, 발급 시간) 클레임은 JWT가 발급된 시간을 식별합니다.
이 클레임은 JWT의 연령을 결정하는 데 사용될 수 있습니다.
해당 값은 NumericDate 값을 포함하는 숫자여야 합니다.
이 클레임의 사용은 선택 사항입니다.

4.1.7. "jti" (JWT ID) Claim

The "jti" (JWT ID) claim provides a unique identifier for the JWT. The identifier value MUST be assigned in a manner that ensures that there is a negligible probability that the same value will be accidentally assigned to a different data object; if the application uses multiple issuers, collisions MUST be prevented among values produced by different issuers as well. The "jti" claim can be used to prevent the JWT from being replayed. The "jti" value is a case-sensitive string. Use of this claim is OPTIONAL.

4.1.7. "jti" (JWT ID) 클레임

"jti" (JWT ID, JWT 식별자) 클레임은 JWT에 대한 고유한 식별자를 제공합니다.
식별자 값은 다른 데이터 개체에 실수로 동일한 값이 할당되는 가능성이
무시할 수 있는 수준으로 할당되어야 합니다.
응용 프로그램이 여러 발급자를 사용하는 경우,
서로 다른 발급자에 의해 생성된 값들 사이에서 충돌이 발생하지 않도록 해야 합니다.
"jti" 클레임은 JWT의 재생을 방지하는 데 사용될 수 있습니다.
"jti" 값은 대소문자를 구분하는 문자열입니다.
이 클레임의 사용은 선택 사항입니다.

4.2. Public Claim Names

Claim Names can be defined at will by those using JWTs. However, in order to prevent collisions, any new Claim Name should either be registered in the IANA "JSON Web Token Claims" registry established by Section 10.1 or be a Public Name: a value that contains a Collision-Resistant Name. In each case, the definer of the name or value needs to take reasonable precautions to make sure they are in control of the part of the namespace they use to define the Claim Name.

4.2. 공개 클레임 이름

클레임 이름은 JWT를 사용하는 사람들이 자유롭게 정의할 수 있습니다.
그러나 충돌을 방지하기 위해 새로운 클레임 이름은 IANA가 설정한 "JSON Web Token Claims" 레지스트리에
등록되어야 하거나, 충돌 방지를 위한 이름을 포함하는 Public Name이어야 합니다.
이 두 경우 모두, 이름 또는 값의 정의자는 정의한 클레임 이름의 일부를 제어하는 데,
합리적인 주의를 기울여야 합니다.

4.3. Private Claim Names

A producer and consumer of a JWT MAY agree to use Claim Names that are Private Names: names that are not Registered Claim Names (Section 4.1) or Public Claim Names (Section 4.2). Unlike Public

4.3. 사설 클레임 이름

JWT의 생성자와 소비자는 등록된 클레임 이름(4.1절)이나 공개 클레임 이름(4.2절)이 아닌
사설 이름을 사용하기로 합의할 수 있습니다.

공개 클레임 이름과 달리, 사설 클레임 이름은 해당 JWT를 생성하고 소비하는 측 사이에만 약속되어 있습니다.

Claim Names, Private Claim Names are subject to collision and should be used with caution.

클레임 네임, 프라이빗 클레임 네임은 충돌이 발생할 수 있으므로 주의해서 사용해야 합니다.

5. JOSE Header

For a JWT object, the members of the JSON object represented by the JOSE Header describe the cryptographic operations applied to the JWT and optionally, additional properties of the JWT. Depending upon whether the JWT is a JWS or JWE, the corresponding rules for the JOSE Header values apply.

This specification further specifies the use of the following Header Parameters in both the cases where the JWT is a JWS and where it is a JWE.

5. JOSE 헤더

JWT 객체의 경우, JOSE 헤더로 표시된 JSON 객체의 멤버는 JWT에 적용된 암호화 작업과 선택적으로 JWT의 추가 속성을 설명합니다. JWT가 JWS 또는 JWE인지에 따라 JOSE 헤더 값에 대한 해당 규칙이 적용됩니다.

이 사양은 JWT가 JWS인 경우와 JWE인 경우 모두에서 다음 헤더 매개변수의 사용을 더 명시화합니다.

5.1. "typ" (Type) Header Parameter

The "typ" (type) Header Parameter defined by [JWS] and [JWE] is used by JWT applications to declare the media type [IANA.MediaTypes] of this complete JWT. This is intended for use by the JWT application when values that are not JWTs could also be present in an application data structure that can contain a JWT object; the application can use this value to disambiguate among the different kinds of objects that might be present. It will typically not be used by applications when it is already known that the object is a JWT. This parameter is ignored by JWT implementations; any processing of this parameter is performed by the JWT application. If present, it is RECOMMENDED that its value be "JWT" to indicate that this object is a JWT. While media type names are not case sensitive, it is RECOMMENDED that "JWT" always be spelled using uppercase characters for compatibility with legacy implementations. Use of this Header Parameter is OPTIONAL.

5.1. "typ" (Type) 헤더 매개변수

[JWS] 및 [JWE]에서 정의된 "typ" (타입) 헤더 매개변수는 JWT 애플리케이션이 이 전체 JWT의 미디어 타입 [IANA.MediaTypes]을 선언하는 데

사용됩니다.
이는 JWT 애플리케이션이 JWT가 아닌 값도 존재할 수 있는
애플리케이션 데이터 구조에 JWT 객체가 포함될 때 사용됩니다.
애플리케이션은 이 값을 사용하여 존재할 수 있는
다양한 종류의 객체들 사이를 구분할 수 있습니다.
일반적으로 애플리케이션은 이미 해당 객체가 JWT임을 알고 있을 때
이를 사용하지 않습니다.
이 매개변수는 JWT 구현에서 무시되며,
이 매개변수의 처리는 JWT 애플리케이션에 의해 수행됩니다.
있을 경우, 이 값이 "JWT"임을 나타내기 위해 이 매개변수의 값이 권장됩니다.
미디어 타입 이름은 대소문자를 구별하지 않지만,
호환성을 위해 "JWT"가 항상 대문자로 표기되는 것이 권장됩니다.
이 헤더 매개변수의 사용은 선택 사항입니다.

5.2. "cty" (Content Type) Header Parameter

The "cty" (content type) Header Parameter defined by [JWS] and [JWE] is used by this specification to convey structural information about the JWT.

이 명세서에서는 [JWS]와 [JWE]에 의해 정의된 "cty" (컨텐츠 유형) 헤더 매개변수가
JWT에 대한 구조적 정보를 전달하는 데 사용됩니다.

In the normal case in which nested signing or encryption operations are not employed, the use of this Header Parameter is NOT RECOMMENDED. In the case that nested signing or encryption is employed, this Header Parameter MUST be present; in this case, the value MUST be "JWT", to indicate that a Nested JWT is carried in this JWT. While media type names are not case sensitive, it is RECOMMENDED that "JWT" always be spelled using uppercase characters for compatibility with legacy implementations. See Appendix A.2 for an example of a Nested JWT.

일반적으로 중첩된 서명 또는 암호화 작업이 사용되지 않는 경우에는
이 헤더 매개변수의 사용을 권장하지 않습니다.
중첩된 서명 또는 암호화가 사용되는 경우에는 이 헤더 매개변수가 반드시 있어야 합니다.
이 경우 값은 반드시 "JWT"여야 하며, 이는 이 JWT에 중첩된 JWT가 포함되어 있음을 나타냅니다.
미디어 유형 이름은 대소문자를 구분하지 않지만,
호환성을 위해 "JWT"는 항상 대문자로 표기하는 것이 권장됩니다.
중첩된 JWT의 예시는 부록 A.2를 참조하세요.

5.3. Replicating Claims as Header Parameters

In some applications using encrypted JWTs, it is useful to have an unencrypted representation of some claims. This might be used, for instance, in application processing rules to determine whether and how to process the JWT before it is decrypted.

5.3. 클레임을 헤더 매개변수로 복제하기

암호화된 JWT를 사용하는 일부 애플리케이션에서는
몇 가지 클레임의 암호화되지 않은 표현을 가지는 것이 유용합니다.
예를 들어, 이는 JWT가 복호화되기 전에 JWT를 처리할지 여부 및 방법을 결정하는
응용 프로그램 처리 규칙에 사용될 수 있습니다.

This specification allows claims present in the JWT Claims Set to be replicated as Header Parameters in a JWT that is a JWE, as needed by the application. If such replicated claims are present, the application receiving them SHOULD verify that their values are identical, unless the application defines other specific processing rules for these claims. It is the responsibility of the application to ensure that only claims that are safe to be transmitted in an unencrypted manner are replicated as Header Parameter values in the JWT.

이 명세서는 JWT Claims Set에 있는 클레임을 JWE인 JWT의 헤더 매개변수로 필요에 따라
복제할 수 있도록 허용합니다. 만약 이와 같은 복제된 클레임이 존재한다면,
받는 응용 프로그램은 해당 값이 동일한지 확인해야 합니다.
응용 프로그램이 이러한 클레임에 대한 다른 특정 처리 규칙을 정의하지 않은 경우,
복제된 클레임의 값이 동일한지 확인하는 것이 권장됩니다.
암호화되지 않은 방식으로 전송해도 안전한 클레임만이 JWT의 헤더 매개변수 값으로 복제되도록
응용 프로그램이 보장하는 것이 책임입니다.

Section 10.4.1 of this specification registers the "iss" (issuer), "sub" (subject), and "aud" (audience) Header Parameter names for the purpose of providing unencrypted replicas of these claims in encrypted JWTs for applications that need them. Other specifications MAY similarly register other names that are registered Claim Names as Header Parameter names, as needed.

이 명세서의 10.4.1 절은 "iss" (발급자), "sub" (주체),
그리고 "aud" (청중) 헤더 매개변수 이름을 등록하여
이러한 클레임의 암호화되지 않은 복제본을 필요로 하는 애플리케이션을 위한
암호화된 JWT에서 제공하는 목적으로 등록합니다.
다른 명세서에서는 필요한 경우 등록된 클레임 이름으로서
헤더 매개변수 이름을 유사하게 등록할 수 있습니다.

6. Unsecured JWTs

To support use cases in which the JWT content is secured by a means other than a signature and/or encryption contained within the JWT (such as a signature on a data structure containing the JWT), JWTs MAY also be created without a signature or encryption. An Unsecured JWT is a JWS using the "alg" Header Parameter value "none" and with the empty string for its JWS Signature value, as defined in the JWA specification [JWA]; it is an Unsecured JWS with the JWT Claims Set as its JWS Payload.

6. 안전하지 않은 JWT

JWT 내부에 포함된 서명 및/또는 암호화와 같은 수단 이외의 방법으로
JWT 콘텐츠가 보호되는 사용 사례를 지원하기 위해
JWT는 서명 또는 암호화 없이도 생성될 수 있습니다.
안전하지 않은 JWT는 "alg" 헤더 매개변수 값으로 "none"을 사용하고,
JWA 명세서 [JWA]에서 정의된 대로 그 JWS 서명 값으로 빈 문자열을 사용하는 JWS입니다.
이는 JWT Claims Set을 그 JWS 페이로드로 사용하는 안전하지 않은 JWS입니다.

6.1. Example Unsecured JWT

The following example JOSE Header declares that the encoded object is an Unsecured JWT:

6.1. 예시 안전하지 않은 JWT

다음 예제 JOSE 헤더는 인코딩된 객체가 안전하지 않은 JWT임을 선언합니다:

```
{"alg":"none"}
```

Base64url encoding the octets of the UTF-8 representation of the JOSE Header yields this encoded JOSE Header value:

JOSE 헤더의 UTF-8 표현의 옥텟을 Base64url로 인코딩하면 다음과 같은 인코딩된 JOSE 헤더 값이 생성됩니다:

```
eyJhbGciOiJub25lIn0
```

The following is an example of a JWT Claims Set:

다음은 JWT 클레임 세트의 예시입니다:

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

Base64url encoding the octets of the UTF-8 representation of the JWT Claims Set yields this encoded JWS Payload (with line breaks for display purposes only):

JWT 클레임 세트의 UTF-8 표현의 옥텟을 Base64url로 인코딩하면 다음과 같은 인코딩된 JWS 페이로드가 생성됩니다 (표시 용도로 줄 바꿈이 포함되어 있습니다):

```
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

The encoded JWS Signature is the empty string.
인코딩된 JWS 서명은 빈 문자열입니다.

Concatenating these encoded parts in this order with period ('.') characters between the parts yields this complete JWT (with line breaks for display purposes only):

이러한 인코딩된 부분들을 순서대로 마침표('.') 문자로 연결하면 다음과 같은 완전한 JWT가 생성됩니다 (표시 용도로 줄 바꿈이 포함되어 있습니다):

```
eyJhbGciOiJub25lIn0
.
```

```
      eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
      cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
      .
```

7. Creating and Validating JWTs
7. JWT 생성 및 유효성 검사

7.1. Creating a JWT
7.1. JWT 생성

To create a JWT, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps.

JWT를 생성하기 위해 다음 단계를 수행합니다.
단계 간의 순서는 입력과 출력 간에 의존 관계가 없는 경우에는 중요하지 않습니다.

1. Create a JWT Claims Set containing the desired claims. Note that whitespace is explicitly allowed in the representation and no canonicalization need be performed before encoding.

1. 원하는 클레임을 포함하는 JWT 클레임 세트를 생성합니다.
표현에서 공백은 명시적으로 허용되며
인코딩 전에 정규화를 수행할 필요가 없음을 유의하세요.

2. Let the Message be the octets of the UTF-8 representation of the JWT Claims Set.

2. 메시지를 JWT 클레임 세트의 UTF-8 표현의 옥텟으로 설정합니다.

3. Create a JOSE Header containing the desired set of Header Parameters. The JWT MUST conform to either the [JWS] or [JWE] specification. Note that whitespace is explicitly allowed in the representation and no canonicalization need be performed before encoding.

3. 원하는 헤더 매개변수 집합을 포함하는 JOSE 헤더를 생성합니다.
JWT는 [JWS] 또는 [JWE] 명세에 준수해야 합니다.
표현에서 공백은 명시적으로 허용되며
인코딩 전에 정규화를 수행할 필요가 없음을 유의하세요.

---

4. Depending upon whether the JWT is a JWS or JWE, there are two cases:

   * If the JWT is a JWS, create a JWS using the Message as the JWS Payload; all steps specified in [JWS] for creating a JWS MUST be followed.

   * Else, if the JWT is a JWE, create a JWE using the Message as the plaintext for the JWE; all steps specified in [JWE] for creating a JWE MUST be followed.

4. JWT가 JWS인지 JWE인지에 따라 두 가지 경우가 있습니다:

   * JWT가 JWS인 경우, 메시지를 JWS 페이로드로 사용하여 JWS를 생성합니다.
     JWS를 생성하는 데 필요한 모든 단계는 [JWS]에서 지정된 대로 따라야 합니다.

   * 그렇지 않으면, JWT가 JWE인 경우,
     메시지를 JWE의 평문으로 사용하여 JWE를 생성합니다.
     JWE를 생성하는 데 필요한 모든 단계는 [JWE]에서 지정된 대로 따라야 합니다.

5. If a nested signing or encryption operation will be performed,
   let the Message be the JWS or JWE, and return to Step 3, using a
   "cty" (content type) value of "JWT" in the new JOSE Header
   created in that step.

5. 중첩된 서명 또는 암호화 작업을 수행할 경우,
   메시지를 해당 JWS 또는 JWE로 설정하고,
   "cty" (컨텐츠 유형) 값을 새로 생성된 JOSE 헤더에서 "JWT"로 설정하여
   해당 단계로 돌아갑니다.

6. Otherwise, let the resulting JWT be the JWS or JWE.

6. 그렇지 않은 경우, 결과로 얻은 JWT를 해당하는 JWS 또는 JWE로 설정합니다.

7.2. Validating a JWT

When validating a JWT, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps. If any of the listed steps fail, then the JWT MUST be rejected -- that is, treated by the application as an invalid input.

JWT를 유효성 검사할 때는 다음 단계를 수행합니다.
입력과 출력 간에 의존 관계가 없는 경우 단계의 순서는 중요하지 않습니다.
나열된 단계 중 어느 하나라도 실패할 경우 JWT는 반드시 거부되어야 합니다.
다시 말해, 응용 프로그램에서 잘못된 입력으로 처리되어야 합니다.

1. Verify that the JWT contains at least one period ('.')
   character.

   1. JWT에 적어도 하나의 마침표 ('.') 문자가 포함되어 있는지 확인합니다.

2. Let the Encoded JOSE Header be the portion of the JWT before the
   first period ('.') character.

   2. 인코딩된 JOSE 헤더를 JWT의 첫 번째 마침표 ('.') 문자 이전의 부분으로 정의합니다.

3. Base64url decode the Encoded JOSE Header following the
   restriction that no line breaks, whitespace, or other additional
   characters have been used.

   3. 추가적인 줄 바꿈, 공백 또는 다른 추가 문자가 사용되지 않은 제약 조건을 따르면서
      인코딩된 JOSE 헤더를 Base64url로 디코딩합니다.

4. Verify that the resulting octet sequence is a UTF-8-encoded

representation of a completely valid JSON object conforming to RFC 7159 [RFC7159]; let the JOSE Header be this JSON object.

4. 결과적인 옥텟 시퀀스가 RFC 7159 [RFC7159]에 준하는 완전히 유효한 JSON 객체의 UTF-8로 인코딩된 표현인지 확인합니다. 이 JSON 객체를 JOSE 헤더로 지정합니다.

5. Verify that the resulting JOSE Header includes only parameters and values whose syntax and semantics are both understood and supported or that are specified as being ignored when not understood.

5. 결과적인 JOSE 헤더가 문법과 의미 모두 이해되고 지원되는 매개변수와 값만 포함하거나, 이해되지 않을 때 무시되어야 하는 것으로 지정된 매개변수와 값을만 포함하는지 확인합니다.

6. Determine whether the JWT is a JWS or a JWE using any of the methods described in Section 9 of [JWE].

6. [JWE]의 섹션 9에 설명된 방법 중 하나를 사용하여, JWT가 JWS인지 JWE인지 결정합니다.

---

7. Depending upon whether the JWT is a JWS or JWE, there are two cases:

7. JWT가 JWS인지 JWE인지에 따라 두 가지 경우가 있습니다:

* If the JWT is a JWS, follow the steps specified in [JWS] for validating a JWS. Let the Message be the result of base64url decoding the JWS Payload.

* JWT가 JWS인 경우, JWS를 유효성 검사하는 [JWS]에 지정된 단계를 따릅니다. 메시지는 JWS 페이로드를 base64url로 디코딩한 결과입니다.

* Else, if the JWT is a JWE, follow the steps specified in [JWE] for validating a JWE. Let the Message be the resulting plaintext.

* 그렇지 않으면, JWT가 JWE인 경우, JWE를 유효성 검사하는 [JWE]에 지정된 단계를 따릅니다. 메시지는 결과 평문입니다.

8. If the JOSE Header contains a "cty" (content type) value of "JWT", then the Message is a JWT that was the subject of nested signing or encryption operations. In this case, return to Step 1, using the Message as the JWT.

8. JOSE 헤더에 "cty" (컨텐츠 유형) 값이 "JWT"인 경우, 메시지는 중첩된 서명 또는 암호화 작업의 대상이 된 JWT입니다. 이 경우, 메시지를 JWT로 사용하여 단계 1로 돌아갑니다.

9. Otherwise, base64url decode the Message following the restriction that no line breaks, whitespace, or other additional characters have been used.

9. 그렇지 않으면, 추가적인 줄 바꿈, 공백 또는 다른 추가 문자가 사용되지 않은 제약 조건을 따르면서 메시지를 base64url로 디코딩합니다.

10. Verify that the resulting octet sequence is a UTF-8-encoded representation of a completely valid JSON object conforming to RFC 7159 [RFC7159]; let the JWT Claims Set be this JSON object.

10. 결과적인 옥텟 시퀀스가 RFC 7159 [RFC7159]에 준하는 완전히 유효한 JSON 객체의 UTF-8로 인코딩된 표현인지 확인합니다. 이 JSON 객체를 JWT 클레임 세트로 지정합니다.

Finally, note that it is an application decision which algorithms may be used in a given context. Even if a JWT can be successfully validated, unless the algorithms used in the JWT are acceptable to the application, it SHOULD reject the JWT.

마지막으로, 주어진 상황에서 어떤 알고리즘이 사용될 수 있는지는 응용 프로그램 결정입니다.
JWT를 성공적으로 검증하더라도 JWT에서 사용된 알고리즘이 응용 프로그램에서 허용되지 않는 경우,
해당 JWT를 거부해야 합니다.

7.3. String Comparison Rules
7.3. 문자열 비교 규칙

Processing a JWT inevitably requires comparing known strings to members and values in JSON objects. For example, in checking what the algorithm is, the Unicode [UNICODE] string encoding "alg" will be checked against the member names in the JOSE Header to see if there is a matching Header Parameter name.

JWT를 처리하는 것은 불가피하게 알려진 문자열을 JSON 객체의 멤버 및 값과 비교하는 작업을 필요로 합니다. 예를 들어, 알고리즘을 확인할 때, Unicode [UNICODE] 문자열 인코딩 "alg"는 JOSE 헤더의 멤버 이름과 일치하는 헤더 매개변수 이름이 있는지 확인하기 위해 확인될 것입니다.

The JSON rules for doing member name comparison are described in Section 8.3 of RFC 7159 [RFC7159]. Since the only string comparison operations that are performed are equality and inequality, the same rules can be used for comparing both member names and member values against known strings.

멤버 이름 비교를 수행하는 JSON 규칙은 RFC 7159 [RFC7159]의 섹션 8.3에 설명되어 있습니다.
수행되는 문자열 비교 작업이 등호 및 부등호인 경우,
멤버 이름 및 멤버 값 모두 알려진 문자열과 비교하는 데 동일한 규칙을 사용할 수 있습니다.

These comparison rules MUST be used for all JSON string comparisons except in cases where the definition of the member explicitly calls out that a different comparison rule is to be used for that member value. In this specification, only the "typ" and "cty" member values

do not use these comparison rules.

이러한 비교 규칙은 멤버 값에 대해 다른 비교 규칙을 사용해야 하는 경우를 제외하고 모든 JSON 문자열 비교에 사용되어야 합니다.
이 명세서에서는 "typ" 및 "cty" 멤버 값만이 이러한 비교 규칙을 사용하지 않습니다.

Some applications may include case-insensitive information in a case-sensitive value, such as including a DNS name as part of the "iss" (issuer) claim value. In those cases, the application may need to define a convention for the canonical case to use for representing the case-insensitive portions, such as lowercasing them, if more than one party might need to produce the same value so that they can be compared. (However, if all other parties consume whatever value the producing party emitted verbatim without attempting to compare it to an independently produced value, then the case used by the producer will not matter.)

일부 응용 프로그램에서는 "iss" (발급자) 클레임 값의 일부로 DNS 이름을 포함하는 등의 대소문자를 구분하는 값에 대소문자를 구분하지 않는 정보를 포함할 수 있습니다.
이러한 경우 응용 프로그램은 비교를 위해 같은 값을 생성할 필요가 있는 경우를 위해 대소문자를 표현하는 규범을 정의해야 할 수 있습니다.
예를 들어, 이러한 부분을 소문자로 변환하는 것입니다.
(그러나 다른 모든 당사자가 생산 당사자가 독립적으로 생성된 값과 비교하려고 하지 않고 생산 당사자가 출력한 값을 소비하는 경우, 생산자가 사용한 경우는 중요하지 않습니다.)

8. Implementation Requirements
8. 구현 요구 사항

This section defines which algorithms and features of this specification are mandatory to implement. Applications using this specification can impose additional requirements upon implementations that they use. For instance, one application might require support for encrypted JWTs and Nested JWTs, while another might require support for signing JWTs with the Elliptic Curve Digital Signature Algorithm (ECDSA) using the P-256 curve and the SHA-256 hash algorithm ("ES256").

이 섹션은 이 명세의 알고리즘과 기능 중 필수로 구현해야 하는 부분을 정의합니다.
이 명세를 사용하는 응용 프로그램은 사용하는 구현에 대해 추가 요구 사항을 부과할 수 있습니다.

예를 들어, 한 응용 프로그램은 암호화된 JWT와 중첩된 JWT를 지원해야 하고,
다른 응용 프로그램은 P-256 곡선을 사용하여 타원 곡선 디지털 서명 알고리즘 (ECDSA)로 JWT에 서명하는 지원을 필요로 할 수 있습니다.
그리고 SHA-256 해시 알고리즘 ("ES256")을 사용합니다.

Of the signature and MAC algorithms specified in JSON Web Algorithms [JWA], only HMAC SHA-256 ("HS256") and "none" MUST be implemented by conforming JWT implementations. It is RECOMMENDED that implementations also support RSASSA-PKCS1-v1_5 with the SHA-256 hash algorithm ("RS256") and ECDSA using the P-256 curve and the SHA-256

hash algorithm ("ES256"). Support for other algorithms and key sizes is OPTIONAL.

JSON Web Algorithms [JWA]에서 지정된 서명 및 MAC 알고리즘 중,
준수하는 JWT 구현에서는 HMAC SHA-256 ("HS256") 및 "none"만 필수로 구현되어야 합니다.
구현에서는 또한 RSASSA-PKCS1-v1_5 및 SHA-256 해시 알고리즘을 사용하는 RSA 알고리즘 ("RS256")
및 P-256 곡선 및 SHA-256 해시 알고리즘을 사용하는 ECDSA ("ES256")를 지원하는 것이 권장됩니다.
다른 알고리즘 및 키 크기에 대한 지원은 선택 사항입니다.

Support for encrypted JWTs is OPTIONAL. If an implementation provides encryption capabilities, of the encryption algorithms specified in [JWA], only RSAES-PKCS1-v1_5 with 2048-bit keys ("RSA1_5"), AES Key Wrap with 128- and 256-bit keys ("A128KW" and "A256KW"), and the composite authenticated encryption algorithm using AES-CBC and HMAC SHA-2 ("A128CBC-HS256" and "A256CBC-HS512") MUST be implemented by conforming implementations. It is RECOMMENDED that implementations also support using Elliptic Curve Diffie-Hellman Ephemeral Static (ECDH-ES) to agree upon a key used to wrap the Content Encryption Key ("ECDH-ES+A128KW" and "ECDH-ES+A256KW") and AES in Galois/Counter Mode (GCM) with 128- and 256-bit keys ("A128GCM" and "A256GCM"). Support for other algorithms and key sizes is OPTIONAL.

암호화된 JWT 지원은 선택 사항입니다.
구현이 암호화 기능을 제공하는 경우, [JWA]에서 지정된 암호화 알고리즘 중 준수하는 구현에서는
RSAES-PKCS1-v1_5 및 2048비트 키를 사용하는 RSA 알고리즘 ("RSA1_5"), 128비트 및 256비트 키를
사용하는 AES 키 랩 ("A128KW" 및 "A256KW"), 그리고 AES-CBC와 HMAC SHA-2를 사용하는
복합 인증 암호화 알고리즘 ("A128CBC-HS256" 및 "A256CBC-HS512")을 반드시 구현해야 합니다.
구현은 또한 Content Encryption Key를 래핑하는 데 사용되는 키를 합의하기 위해
타원 곡선 디피-헬만 (ECDH-ES) 사용을 지원하는 것이 권장됩니다
("ECDH-ES+A128KW" 및 "ECDH-ES+A256KW"), 그리고 128비트 및 256비트 키를 사용하는
AES Galois/Counter Mode (GCM) ("A128GCM" 및 "A256GCM")을 지원하는 것이 권장됩니다.
다른 알고리즘 및 키 크기에 대한 지원은 선택 사항입니다.

Support for Nested JWTs is OPTIONAL.
중첩된 JWT 지원은 선택 사항입니다.

---

9. URI for Declaring that Content is a JWT

This specification registers the URN "urn:ietf:params:oauth:token-type:jwt" for use by applications that declare content types using URIs (rather than, for instance, media types) to indicate that the content referred to is a JWT.

9. 콘텐츠가 JWT임을 선언하는 URI

이 명세서는 콘텐츠 유형을 URI를 사용하여 선언하는 응용 프로그램에서
사용하기 위해 "urn:ietf:params:oauth:token-type:jwt" URN을 등록합니다.
이는 콘텐츠가 JWT임을 나타냅니다.

10. IANA Considerations
10. IANA 고려사항

10.1. JSON Web Token Claims Registry
10.1. JSON 웹 토큰 클레임 레지스트리

This section establishes the IANA "JSON Web Token Claims" registry
for JWT Claim Names. The registry records the Claim Name and a
reference to the specification that defines it. This section
registers the Claim Names defined in Section 4.1.

이 섹션은 JWT 클레임 이름을 위한 IANA "JSON Web Token Claims" 레지스트리를 설정합니다.
이 레지스트리에는 클레임 이름과 해당 클레임을 정의하는 명세에 대한 참조가 기록됩니다.
이 섹션에서는 섹션 4.1에 정의된 클레임 이름을 등록합니다.

Values are registered on a Specification Required [RFC5226] basis
after a three-week review period on the jwt-reg-review@ietf.org
mailing list, on the advice of one or more Designated Experts.
However, to allow for the allocation of values prior to publication,
the Designated Experts may approve registration once they are
satisfied that such a specification will be published.

값은 명세가 요구하는 대로 등록됩니다.
등록 요청은 jwt-reg-review@ietf.org 메일링 리스트에서 3주의 검토 기간을 거친 후
하나 이상의 지정된 전문가의 조언을 받아서 이루어집니다.
그러나 게시 전에 값을 할당할 수 있도록 하기 위해 지정된 전문가는
해당 명세가 게시될 것이라고 확신할 경우 등록을 승인할 수 있습니다.

Registration requests sent to the mailing list for review should use
an appropriate subject (e.g., "Request to register claim: example").

검토를 위해 메일링 리스트로 보내진 등록 요청은 적절한 제목을 사용해야 합니다
(예: "클레임 등록 요청: 예제").

Within the review period, the Designated Experts will either approve
or deny the registration request, communicating this decision to the
review list and IANA. Denials should include an explanation and, if
applicable, suggestions as to how to make the request successful.
Registration requests that are undetermined for a period longer than
21 days can be brought to the IESG's attention (using the
iesg@ietf.org mailing list) for resolution.

검토 기간 내에, 지정된 전문가들은 등록 요청을 승인하거나 거부하고
이 결정을 검토 목록과 IANA에 통보합니다.
거부된 경우 설명을 포함하고, 해당하는 경우 요청을 성공적으로 처리할 수 있는 방법에 대한
제안을 포함해야 합니다.
21일 이상 미결정된 등록 요청은 IESG의 관심을 끌 수 있으며
(iesg@ietf.org 메일링 리스트를 사용하여), 해결할 수 있습니다.

Criteria that should be applied by the Designated Experts includes
determining whether the proposed registration duplicates existing

functionality, whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration description is clear.

지정된 전문가들이 적용해야 할 기준은 제안된 등록이 기존 기능을 중복하는지 여부, 일반적으로 적용 가능한지 여부 또는 단일 응용 프로그램에만 유용한지 여부, 등록 설명이 명확한지 여부를 결정하는 것입니다.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

IANA는 지정된 전문가들로부터만 레지스트리 업데이트를 수락해야 하며, 모든 등록 요청은 검토 메일링 리스트로 전달되어야 합니다.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed review of registration decisions.

해당 명세서를 사용하는 다양한 응용 프로그램의 관점을 대표할 수 있는 여러 지정 전문가를 지명하는 것이 제안되며, 이는 등록 결정에 대한 넓은 범위의 정보를 제공하기 위한 것입니다.

In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

특정 전문가에게 이해 관계의 충돌을 초래할 수 있는 등록 결정이 있는 경우, 해당 전문가는 다른 전문가들의 판단에 따라야 합니다.

10.1.1. Registration Template
10.1.1. 등록 템플릿

Claim Name:
The name requested (e.g., "iss"). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the name be short -- that is, not to exceed 8 characters without a compelling reason to do so. This name is case sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.

클레임 이름:
요청된 이름(예: "iss").
이 명세의 주요 목표 중 하나는 결과적인 표현이 간결하게 유지되는 것이므로, 이름이 짧은 것이 권장됩니다.
즉, 특별한 이유가 없는 한 8자를 초과하지 않아야 합니다.
이 이름은 대소문자를 구분합니다.
지정된 전문가들이 예외를 허용할만한 강력한 이유가 있다고 설명하지 않는 한,

이름은 대소문자를 구분하지 않는 방식으로 다른 등록된 이름과 일치하지 않을 수 있습니다.

Claim Description:
Brief description of the claim (e.g., "Issuer").

주장 설명:
주장의 간략한 설명 (예: "발행자").

Change Controller:
For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

변경 관리자:
표준 트랙 RFC의 경우 "IESG"를 나열하십시오.
다른 경우에는 책임 당사자의 이름을 제공하십시오.
(예: 우편 주소, 이메일 주소, 홈페이지 URI 등의 기타 세부 정보도 포함 가능)

Specification Document(s):
Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

사양 문서:
매개변수를 지정하는 문서 또는 문서에 대한 참조를 제공하십시오.
가능하면 문서 사본을 검색할 수 있는 URI를 포함하십시오.
관련 섹션의 표시도 포함할 수 있지만 필수는 아닙니다.

10.1.2. Initial Registry Contents
10.1.2. 초기 레지스트리 내용

- Claim Name: "iss"
- Claim Description: Issuer
- Change Controller: IESG
- Specification Document(s): Section 4.1.1 of RFC 7519

- 클레임 이름: "iss"
- 클레임 설명: 발행자
- 변경 관리자: IESG
- 사양 문서: RFC 7519의 섹션 4.1.1

- Claim Name: "sub"
- Claim Description: Subject
- Change Controller: IESG
- Specification Document(s): Section 4.1.2 of RFC 7519

- 클레임 이름: "sub"
- 클레임 설명: 주제
- 변경 관리자: IESG
- 사양 문서: RFC 7519의 섹션 4.1.2

- Claim Name: "aud"
- Claim Description: Audience
- Change Controller: IESG
- Specification Document(s): Section 4.1.3 of RFC 7519

   o 클레임 이름: "aud"
   o 클레임 설명: 청중
   o 변경 관리자: IESG
   o 사양 문서: RFC 7519의 섹션 4.1.3

---

   o Claim Name: "exp"
   o Claim Description: Expiration Time
   o Change Controller: IESG
   o Specification Document(s): Section 4.1.4 of RFC 7519

   o 클레임 이름: "exp"
   o 클레임 설명: 만료 시간
   o 변경 관리자: IESG
   o 사양 문서: RFC 7519의 섹션 4.1.4

   o Claim Name: "nbf"
   o Claim Description: Not Before
   o Change Controller: IESG
   o Specification Document(s): Section 4.1.5 of RFC 7519

   o 클레임 이름: "nbf"
   o 클레임 설명: 시작 시간
   o 변경 관리자: IESG
   o 사양 문서: RFC 7519의 섹션 4.1.5

   o Claim Name: "iat"
   o Claim Description: Issued At
   o Change Controller: IESG
   o Specification Document(s): Section 4.1.6 of RFC 7519

   o 클레임 이름: "iat"
   o 클레임 설명: 발행 시간
   o 변경 관리자: IESG
   o 사양 문서: RFC 7519의 섹션 4.1.6

   o Claim Name: "jti"
   o Claim Description: JWT ID
   o Change Controller: IESG
   o Specification Document(s): Section 4.1.7 of RFC 7519

   o 클레임 이름: "jti"
   o 클레임 설명: JWT 식별자
   o 변경 관리자: IESG
   o 사양 문서: RFC 7519의 섹션 4.1.7

10.2. Sub-Namespace Registration of urn:ietf:params:oauth:token-type:jwt

10.2. urn:ietf:params:oauth:token-type:jwt의 하위 네임스페이스 등록

10.2.1. Registry Contents

10.2.1. 레지스트리 내용

This section registers the value "token-type:jwt" in the IANA "OAuth URI" registry established by "An IETF URN Sub-Namespace for OAuth" [RFC6755], which can be used to indicate that the content is a JWT.

이 섹션에서는 "An IETF URN Sub-Namespace for OAuth" [RFC6755]에서 설정된 IANA "OAuth URI" 레지스트리에 값 "token-type:jwt"를 등록합니다. 이는 콘텐츠가 JWT임을 나타낼 수 있습니다.

- URN: urn:ietf:params:oauth:token-type:jwt
- Common Name: JSON Web Token (JWT) Token Type
- Change Controller: IESG
- Specification Document(s): RFC 7519

- URN: urn:ietf:params:oauth:token-type:jwt
- 일반 이름: JSON Web Token (JWT) 토큰 유형
- 변경 관리자: IESG
- 사양 문서: RFC 7519

10.3. Media Type Registration
10.3. 미디어 유형 등록

10.3.1. Registry Contents
10.3.1. 레지스트리 내용

This section registers the "application/jwt" media type [RFC2046] in the "Media Types" registry [IANA.MediaTypes] in the manner described in RFC 6838 [RFC6838], which can be used to indicate that the content is a JWT.

이 섹션에서는 RFC 6838 [RFC6838]에서 설명된 방식으로 "미디어 유형(Media Type)" 레지스트리 [IANA.MediaTypes]에 "application/jwt" 미디어 유형 [RFC2046]을 등록합니다. 이는 콘텐츠가 JWT임을 나타낼 수 있습니다.

- Type name: application
- Subtype name: jwt
- Required parameters: n/a
- Optional parameters: n/a
- Encoding considerations: 8bit; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- Security considerations: See the Security Considerations section of RFC 7519
- Interoperability considerations: n/a
- Published specification: RFC 7519
- Applications that use this media type: OpenID Connect, Mozilla Persona, Salesforce, Google, Android, Windows Azure, Amazon Web Services, and numerous others
- Fragment identifier considerations: n/a

- Additional information:

- 유형 이름: application
- 서브타입 이름: jwt
- 필수 매개변수: 없음
- 선택적 매개변수: 없음
- 인코딩 고려 사항: 8비트;
  JWT 값은 기본적으로 base64url로 인코딩된 값들의 시리즈로 구성되어 있으며,
  이들 값은 점('.') 문자로 구분됩니다.
- 보안 고려 사항: RFC 7519의 보안 고려 사항 섹션을 참조하십시오.
- 상호 운용성 고려 사항: 없음
- 공개된 사양: RFC 7519
- 이 미디어 유형을 사용하는 애플리케이션:
  OpenID Connect, Mozilla Persona, Salesforce, Google,
  Android, Windows Azure, Amazon Web Services 등 많은 다른 애플리케이션
- 프래그먼트 식별자 고려 사항: 없음
- 추가 정보:

  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a

  매직 넘버: 없음
  파일 확장자: 없음
  맥킨토시 파일 유형 코드: 없음

- Person & email address to contact for further information:
  Michael B. Jones, mbj@microsoft.com
- Intended usage: COMMON
- Restrictions on usage: none
- Author: Michael B. Jones, mbj@microsoft.com
- Change controller: IESG
- Provisional registration? No

- 추가 정보를 위해 연락할 사람 및 이메일 주소: Michael B. Jones, mbj@microsoft.com
- 의도된 사용: 일반
- 사용 제한: 없음
- 저자: Michael B. Jones, mbj@microsoft.com
- 변경 관리자: IESG
- 임시 등록 여부: 아니요

10.4. Header Parameter Names Registration
10.4. 헤더 매개변수 이름 등록

This section registers specific Claim Names defined in Section 4.1 in the IANA "JSON Web Signature and Encryption Header Parameters" registry established by [JWS] for use by claims replicated as Header Parameters in JWEs, per Section 5.3.

이 섹션에서는 섹션 4.1에서 정의된 특정 클레임 이름을
JWEs의 헤더 매개변수로 복제되는 클레임으로 사용하기 위해
[JWS]에 의해 설정된 IANA "JSON Web Signature and Encryption Header Parameters"
레지스트리에 등록합니다.

10.4.1. Registry Contents
10.4.1. 레지스트리 내용

- Header Parameter Name: "iss"
- Header Parameter Description: Issuer
- Header Parameter Usage Location(s): JWE
- Change Controller: IESG
- Specification Document(s): Section 4.1.1 of RFC 7519

- 헤더 매개변수 이름: "iss"
- 헤더 매개변수 설명: 발급자
- 헤더 매개변수 사용 위치: JWE
- 변경 컨트롤러: IESG
- 명세서 문서: RFC 7519의 섹션 4.1.1

- Header Parameter Name: "sub"
- Header Parameter Description: Subject
- Header Parameter Usage Location(s): JWE
- Change Controller: IESG
- Specification Document(s): Section 4.1.2 of RFC 7519

- 헤더 매개변수 이름: "sub"
- 헤더 매개변수 설명: 주체
- 헤더 매개변수 사용 위치: JWE
- 변경 컨트롤러: IESG
- 명세서 문서: RFC 7519의 섹션 4.1.2

- Header Parameter Name: "aud"
- Header Parameter Description: Audience
- Header Parameter Usage Location(s): JWE
- Change Controller: IESG
- Specification Document(s): Section 4.1.3 of RFC 7519

- 헤더 매개변수 이름: "aud"
- 헤더 매개변수 설명: 청중
- 헤더 매개변수 사용 위치: JWE
- 변경 컨트롤러: IESG
- 명세서 문서: RFC 7519의 섹션 4.1.3

11. Security Considerations
11. 보안 고려 사항

All of the security issues that are pertinent to any cryptographic application must be addressed by JWT/JWS/JWE/JWK agents. Among these issues are protecting the user's asymmetric private and symmetric secret keys and employing countermeasures to various attacks.

모든 암호화 응용 프로그램에 관련된 보안 문제는
JWT/JWS/JWE/JWK 에이전트에 의해 처리되어야 합니다.
이러한 문제 중에는 사용자의 비대칭 개인 및 대칭 비밀 키를 보호하고
다양한 공격에 대응하기 위한 대책을 채택하는 것이 포함됩니다.

All the security considerations in the JWS specification also apply to JWT, as do the JWE security considerations when encryption is employed. In particular, Sections 10.12 ("JSON Security

Considerations") and 10.13 ("Unicode Comparison Security Considerations") of [JWS] apply equally to the JWT Claims Set in the same manner that they do to the JOSE Header.

JWS 사양에서의 모든 보안 고려 사항은 JWT에도 적용되며, 암호화가 사용될 때에는 JWE 보안 고려 사항도 적용됩니다. 특히, [JWS]의 섹션 10.12 ("JSON 보안 고려 사항") 및 10.13 ("유니코드 비교 보안 고려 사항")은 JOSE 헤더와 마찬가지로 JWT Claims Set에도 동일하게 적용됩니다.

11.1. Trust Decisions
11.1. 신뢰 결정

The contents of a JWT cannot be relied upon in a trust decision unless its contents have been cryptographically secured and bound to the context necessary for the trust decision. In particular, the key(s) used to sign and/or encrypt the JWT will typically need to verifiably be under the control of the party identified as the issuer of the JWT.

JWT의 내용은 그 내용이 암호적으로 보호되고 신뢰 결정에 필요한 컨텍스트에 바인딩되지 않는 한 신뢰 결정에서 의존해서는 안 됩니다. 특히, JWT를 서명하고/또는 암호화하는 데 사용되는 키는 일반적으로 JWT의 발급자로 식별된 당사자의 제어 하에 있어야 합니다.

11.2. Signing and Encryption Order
11.2. 서명 및 암호화 순서

While syntactically the signing and encryption operations for Nested JWTs may be applied in any order, if both signing and encryption are necessary, normally producers should sign the message and then

중첩 JWT에 대한 구문상 서명 및 암호화 작업은 어떤 순서로든 적용될 수 있지만, 서명과 암호화가 모두 필요한 경우에는 일반적으로 생산자는 메시지를 서명한 다음

---

encrypt the result (thus encrypting the signature). This prevents attacks in which the signature is stripped, leaving just an encrypted message, as well as providing privacy for the signer. Furthermore, signatures over encrypted text are not considered valid in many jurisdictions.

결과를 암호화합니다 (따라서 서명을 암호화함으로써 서명을 남겨두고 메시지만 암호화하는 공격을 방지하며, 서명자의 개인 정보를 보호합니다. 또한, 암호화된 텍스트 위의 서명은 많은 관할 지역에서 유효하지 않은 것으로 간주되지 않습니다.

Note that potential concerns about security issues related to the order of signing and encryption operations are already addressed by the underlying JWS and JWE specifications; in particular, because JWE

only supports the use of authenticated encryption algorithms, cryptographic concerns about the potential need to sign after encryption that apply in many contexts do not apply to this specification.

서명 및 암호화 작업의 순서와 관련된 보안 문제에 대한 잠재적인 우려에 대한 처리는 이미 기본 JWS 및 JWE 사양에서 처리되어 있음을 유의하십시오.
특히, JWE는 인증된 암호화 알고리즘의 사용만 지원하므로, 많은 문맥에서 암호화 후에 서명이 필요한 경우에 적용되는 암호학적 우려는 이 사양에 적용되지 않습니다.

12. Privacy Considerations
12. 개인 정보 보호 고려 사항

A JWT may contain privacy-sensitive information. When this is the case, measures MUST be taken to prevent disclosure of this information to unintended parties. One way to achieve this is to use an encrypted JWT and authenticate the recipient. Another way is to ensure that JWTs containing unencrypted privacy-sensitive information are only transmitted using protocols utilizing encryption that support endpoint authentication, such as Transport Layer Security (TLS). Omitting privacy-sensitive information from a JWT is the simplest way of minimizing privacy issues.

JWT에는 개인 정보를 포함할 수 있습니다.
이 경우, 이 정보가 부적절한 당사자에게 노출되지 않도록 조치를 취해야 합니다.
이를 달성하는 한 가지 방법은 암호화된 JWT를 사용하고 수신자를 인증하는 것입니다.
또 다른 방법은 암호화되지 않은 개인 정보를 포함하는 JWT가 암호화를 지원하고 엔드포인트 인증을 제공하는 프로토콜을 사용하여만 전송되도록 하는 것입니다.
개인 정보를 생략하는 것은 개인 정보 문제를 최소화하는 가장 간단한 방법입니다.

Appendix A. JWT Examples
부록 A. JWT 예제

This section contains examples of JWTs. For other example JWTs, see Section 6.1 of this document and Appendices A.1 - A.3 of [JWS].

이 섹션에는 JWT의 예제가 포함되어 있습니다.
다른 예제 JWT에 대해서는 이 문서의 섹션 6.1 및 [JWS]의 부록 A.1 - A.3을 참조하십시오.

A.1. Example Encrypted JWT
A.1. 암호화된 JWT 예제

This example encrypts the same claims as used in Section 3.1 to the recipient using RSAES-PKCS1-v1_5 and AES_128_CBC_HMAC_SHA_256.

이 예제는 RSAES-PKCS1-v1_5 및 AES_128_CBC_HMAC_SHA_256을 사용하여 Section 3.1에서 사용한 것과 동일한 클레임을 수신자에게 암호화합니다.

The following example JOSE Header declares that:

다음 예제 JOSE 헤더는 다음을 선언합니다:

o The Content Encryption Key is encrypted to the recipient using the RSAES-PKCS1-v1_5 algorithm to produce the JWE Encrypted Key.

o 컨텐츠 암호화 키는 수신자에게 RSAES-PKCS1-v1_5 알고리즘을 사용하여 JWE 암호화된 키를 생성합니다.

o Authenticated encryption is performed on the plaintext using the AES_128_CBC_HMAC_SHA_256 algorithm to produce the JWE Ciphertext and the JWE Authentication Tag.

o 평문은 AES_128_CBC_HMAC_SHA_256 알고리즘을 사용하여 인증된 암호화가 수행되어 JWE 암호문과 JWE 인증 태그를 생성합니다.

```
{"alg":"RSA1_5","enc":"A128CBC-HS256"}
```

Other than using the octets of the UTF-8 representation of the JWT Claims Set from Section 3.1 as the plaintext value, the computation of this JWT is identical to the computation of the JWE in Appendix A.2 of [JWE], including the keys used.

이 JWT의 계산은 Section 3.1의 JWT 클레임 세트의 UTF-8 표현의 옥텟을 평문 값으로 사용한다는 점을 제외하고는 [JWE]의 부록 A.2의 JWE 계산과 동일하며, 사용된 키도 동일합니다.

The final result in this example (with line breaks for display purposes only) is:
이 예제의 최종 결과(표시 목적을 위한 줄 바꿈 포함)는 다음과 같습니다:

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.
QR1Owv2ug2WyPBnbQrRARTeEk9kDO2w8qDcjiHnSJflSdv1iNqhWXaKH4MqAkQtM
oNfABIPJaZm0HaA415sv3aeuBWnD8J-Ui7Ah6cWafs3ZwwFKDFUUsWHSK-IPKxLG
TkND09XyjORj_CHAgOPJ-Sd8ONQRnJvWn_hXV1BNMHzUjPyYwEsRhDhzjAD26ima
sOTsgruobpYGoQcXUwFDn7moXPRfDE8-NoQX7N7ZYMmpUDkR-Cx9obNGwJQ3nM52
YCitxoQVPzjbl7WBuB7AohdBoZOdZ24WlN1lVIeh8v1K4krB8xgKvRU8kgFrEn_a
1rZgN5TiysnmzTROF869lQ.
AxY8DCtDaGlsbGljb3RoZQ.
MKOle7UQrG6nSxTLX6Mqwt0orbHvAKeWnDYvpIAeZ72deHxz3roJDXQyhxx0wKaM
HDjUEOKIwrtkHthpqEanSBNYHZgmNOV7sln1Eu9g3J8.
fiK51VwhsxJ-siBMR-YFiA
```

A.2. Example Nested JWT
A.2. 중첩 JWT 예제

This example shows how a JWT can be used as the payload of a JWE or JWS to create a Nested JWT. In this case, the JWT Claims Set is first signed, and then encrypted.

이 예제는 JWT를 JWE 또는 JWS의 페이로드로 사용하여 중첩 JWT를 생성하는 방법을 보여줍니다.
이 경우 JWT 클레임 세트가 먼저 서명되고, 그 다음 암호화됩니다.

The inner signed JWT is identical to the example in Appendix A.2 of [JWS]. Therefore, its computation is not repeated here. This example then encrypts this inner JWT to the recipient using RSAES-PKCS1-v1_5 and AES_128_CBC_HMAC_SHA_256.

내부 서명된 JWT는 [JWS]의 부록 A.2 예제와 동일합니다.
따라서 그 계산은 여기서 반복되지 않습니다.
이 예제에서는 이 내부 JWT를 RSAES-PKCS1-v1_5 및 AES_128_CBC_HMAC_SHA_256을 사용하여 수신자에게 암호화합니다.

The following example JOSE Header declares that:
다음 예제 JOSE 헤더는 다음을 선언합니다:

- The Content Encryption Key is encrypted to the recipient using the RSAES-PKCS1-v1_5 algorithm to produce the JWE Encrypted Key.
- 내용 암호화 키는 RSAES-PKCS1-v1_5 알고리즘을 사용하여 수신자에게 암호화됩니다. 이를 통해 JWE 암호화 키가 생성됩니다.

- Authenticated encryption is performed on the plaintext using the AES_128_CBC_HMAC_SHA_256 algorithm to produce the JWE Ciphertext and the JWE Authentication Tag.
- 평문은 AES_128_CBC_HMAC_SHA_256 알고리즘을 사용하여 인증 암호화가 수행되어 JWE 암호문과 JWE 인증 태그가 생성됩니다.

- The plaintext is itself a JWT.
- 평문 자체가 JWT입니다.

```
{"alg":"RSA1_5","enc":"A128CBC-HS256","cty":"JWT"}
```

Base64url encoding the octets of the UTF-8 representation of the JOSE Header yields this encoded JOSE Header value:

JOSE 헤더의 UTF-8 표현의 옥텟을 Base64url로 인코딩하면
위와 같은 인코딩된 JOSE 헤더 값이 생성됩니다.

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5IjoiSldUIn0
```

The computation of this JWT is identical to the computation of the JWE in Appendix A.2 of [JWE], other than that different JOSE Header, plaintext, JWE Initialization Vector, and Content Encryption Key values are used. (The RSA key used is the same.)

이 JWT의 계산은 [JWE]의 부록 A.2의 JWE 계산과 동일하며,
다른 JOSE Header, 평문, JWE 초기화 벡터 및 내용 암호화 키 값이 사용됩니다.
(사용된 RSA 키는 동일합니다.)

The plaintext used is the octets of the ASCII [RFC20] representation of the JWT at the end of Appendix A.2.1 of [JWS] (with all whitespace and line breaks removed), which is a sequence of 458 octets.

사용된 평문은 [JWS]의 부록 A.2.1 끝의 JWT의 ASCII [RFC20] 표현의 옥텟으로,
이는 공백 및 줄 바꿈이 모두 제거된 458개의 옥텟 시퀀스입니다.

The JWE Initialization Vector value used (using JSON array notation) is:
사용된 JWE 초기화 벡터 값(JSON 배열 표기법 사용)은 다음과 같습니다:

```
[82, 101, 100, 109, 111, 110, 100, 32, 87, 65, 32, 57, 56, 48, 53,
```

50]

This example uses the Content Encryption Key represented by the base64url-encoded value below:

이 예시에서 사용된 내용 암호화 키는 아래의 base64url로 인코딩된 값입니다:

```
GawgguFyGrWKav7AX4VKUg
```

The final result for this Nested JWT (with line breaks for display purposes only) is:

이 중첩된 JWT의 최종 결과(표시 목적을 위한 줄 바꿈 포함)는 다음과 같습니다:

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5IjoiSldU
In0.
g_hEwksO1Ax8Qn7HoN-BVeBoa8FXe0kpyk_XdcSmxvcM5_P296JXXtoHISr_DD_M
qewaQSH4dZOQHoUgKLeFly-9RI11TG-_Ge1bZFazBPwKC5lJ6OLANLMd0QSL4fYE
b9ERe-epKYE3xb2jfY1AltHqBO-PM6j23Guj2yDKnFv6WO72tteVzm_2n17SBFvh
DuR9a2nHTE67pe0XGBUS_TK7ecA-iVq5COeVdJR4U4VZGGlxRGPLRHvolVLEHx6D
YyLpw30Ay9R6d68YCLi9FYTq3hIXPK_-dmPlOUlKvPr1GgJzRoeC9G5qCvdcHWsq
JGTO_z3Wfo5zsqwkxruxwA.
UmVkbW9uZCBXQSA5ODA1Mg.
VwHERHPvCNcHHpTjkoigx3_ExK0Qc71RMEParpatm0X_qpg-w8kozSjfNIPPXiTB
BLXR65CIPkFqz4l1Ae9w_uowKiwyi9acgVztAi-pSL8GQSXnaamh9kX1mdh3M_TT
-FZGQFQsFhu0Z72gJKGdfGE-OE7hS1zuBD5oEUfk0Dmb0VzWEzpxxiSSBbBAzP10
l56pPfAtrjEYw-7ygeMkwBl6Z_mLS6w6xUgKlvW6ULmkV-uLC4FUiyKECK4e3WZY
Kw1bpgIqGYsw2v_grHjszJZ-_I5uM-9RA8ycX9KqPRp9gc6pXmoU_-27ATs9XCvr
ZXUtK2902AUzqpeEUJYjWWxSNsS-r1TJ1I-FMJ4XyAiGrfmo9hQPcNBYxPz3GQb2
8Y5CLSQfNgKSGt0A4isp1hBUXBHAndgtcslt7ZoQJaKe_nNJgNliWtWpJ_ebuOpE
l8jdhehdccnRMIwAmU1n7SPkmhIl1HlSOpvcvDfhUN5wuqU955vOBvfkBOh5A11U
zBuo2WlgZ6hYi9-e3w29bR0C2-pp3jbqxEDw3iWaf2dc5b-LnR0FEYXvI_tYk5rd
_J9N0mg0tQ6RbpxNEMNoA9QWk5lgdPvbh9BaO195abQ.
AVO9iT5AV4CzvDJCdhSFlQ
```

Appendix B. Relationship of JWTs to SAML Assertions
부록 B. JWT와 SAML 어설션의 관계

Security Assertion Markup Language (SAML) 2.0 [OASIS.saml-core-2.0-os] provides a standard for creating security tokens with greater expressivity and more security options than supported by JWTs. However, the cost of this flexibility and expressiveness is both size and complexity. SAML's use of XML [W3C.CR-xml11-20060816] and XML Digital Signature (DSIG) [RFC3275] contributes to the size of SAML Assertions; its use of XML and especially XML Canonicalization [W3C.REC-xml-c14n-20010315] contributes to their complexity.

보안 주장 표시 언어(Security Assertion Markup Language, SAML) 2.0은 JWT보다 더 큰 표현력과 보안 옵션을 갖는 보안 토큰을 생성하기 위한 표준을 제공합니다. 그러나 이러한 유연성과 표현력은 크기와 복잡성의 측면에서 비용이 발생합니다.

SAML은 XML 및 XML 디지털 서명(DSIG)의 사용으로 인해 SAML 어설션의 크기가 커지고, 특히 XML 및 특히 XML 정규화(XML Canonicalization)의 사용으로 인해 복잡성이 증가합니다.

JWTs are intended to provide a simple security token format that is small enough to fit into HTTP headers and query arguments in URIs. It does this by supporting a much simpler token model than SAML and using the JSON [RFC7159] object encoding syntax. It also supports securing tokens using Message Authentication Codes (MACs) and digital signatures using a smaller (and less flexible) format than XML DSIG.

JWT는 HTTP 헤더 및 URI의 쿼리 인수에 적합한 크기의 간단한 보안 토큰 형식을 제공하기 위해 설계되었습니다. 이는 SAML보다 훨씬 간단한 토큰 모델을 지원하고 JSON [RFC7159] 객체 인코딩 구문을 사용함으로써 이루어집니다. 또한 XML DSIG보다 더 작고(그러나 유연성이 떨어지는) 형식을 사용하여 메시지 인증 코드(MAC) 및 디지털 서명을 지원합니다.

Therefore, while JWTs can do some of the things SAML Assertions do, JWTs are not intended as a full replacement for SAML Assertions, but rather as a token format to be used when ease of implementation or compactness are considerations.

따라서 JWT가 SAML 어설션의 일부 기능을 수행할 수 있지만 JWT는 SAML 어설션의 완전한 대체가 아니라 구현의 용이성이나 간결성이 고려되어야 할 때 사용되는 토큰 형식으로 고려되었습니다.

SAML Assertions are always statements made by an entity about a subject. JWTs are often used in the same manner, with the entity making the statements being represented by the "iss" (issuer) claim, and the subject being represented by the "sub" (subject) claim. However, with these claims being optional, other uses of the JWT format are also permitted.

SAML 주장은 항상 주체에 대한 엔터티의 발언입니다. JWT(Javascript Web Token)는 종종 같은 방식으로 사용되며, 발언을 하는 엔터티는 "iss" (발행자) 클레임으로, 주체는 "sub" (주제) 클레임으로 나타냅니다. 그러나 이러한 클레임이 선택적이므로 JWT 형식의 다른 사용도 허용됩니다.

Appendix C. Relationship of JWTs to Simple Web Tokens (SWTs)
부록 C. JWT와 간단한 웹 토큰(SWT)의 관계

Both JWTs and SWTs [SWT], at their core, enable sets of claims to be communicated between applications. For SWTs, both the claim names and claim values are strings. For JWTs, while claim names are strings, claim values can be any JSON type. Both token types offer cryptographic protection of their content: SWTs with HMAC SHA-256 and JWTs with a choice of algorithms, including signature, MAC, and encryption algorithms.

JWT와 SWT는 핵심적으로 애플리케이션 간에 클레임 집합을 통신할 수 있게 합니다.
SWT의 경우, 클레임 이름과 클레임 값 모두 문자열입니다.
JWT의 경우, 클레임 이름은 문자열이지만, 클레임 값은 JSON 타입일 수 있습니다.
두 토큰 유형 모두 그들의 내용에 대한 암호화 보호를 제공합니다:
SWT는 HMAC SHA-256으로,
JWT는 서명, MAC 및 암호화 알고리즘을 포함한 알고리즘 선택을 통해 이루어집니다.

Acknowledgements

The authors acknowledge that the design of JWTs was intentionally influenced by the design and simplicity of SWTs [SWT] and ideas for JSON tokens that Dick Hardt discussed within the OpenID community.

Solutions for signing JSON content were previously explored by Magic Signatures [MagicSignatures], JSON Simple Sign [JSS], and Canvas Applications [CanvasApp], all of which influenced this document.

This specification is the work of the OAuth working group, which includes dozens of active and dedicated participants. In particular, the following individuals contributed ideas, feedback, and wording that influenced this specification:

Dirk Balfanz, Richard Barnes, Brian Campbell, Alissa Cooper, Breno de Medeiros, Stephen Farrell, Yaron Y. Goland, Dick Hardt, Joe Hildebrand, Jeff Hodges, Edmund Jay, Warren Kumari, Ben Laurie, Barry Leiba, Ted Lemon, James Manger, Prateek Mishra, Kathleen Moriarty, Tony Nadalin, Axel Nennker, John Panzer, Emmanuel Raviart, David Recordon, Eric Rescorla, Jim Schaad, Paul Tarjan, Hannes Tschofenig, Sean Turner, and Tom Yu.

Hannes Tschofenig and Derek Atkins chaired the OAuth working group and Sean Turner, Stephen Farrell, and Kathleen Moriarty served as Security Area Directors during the creation of this specification.

Authors' Addresses

Michael B. Jones
Microsoft

EMail: mbj@microsoft.com
URI: http://self-issued.info/

John Bradley
Ping Identity

EMail: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Nat Sakimura

Nomura Research Institute

EMail: n-sakimura@nri.co.jp
URI: http://nat.sakimura.org/