Network Working Group
Internet-Draft
Intended status: Informational
Expires: 9 April 2026

P. M. Hallam-Baker
ThresholdSecrets.com

6 October 2025

Encrypted Authenticated Resource Locator
draft-hallambaker-earl-01

Abstract

This document describes the Encrypted Authenticated Resource Locator (EARL) URI scheme and the encoding and decoding of the associated content data. An EARL is a bearer token that allows an encrypted data object to be located, decrypted and authenticated using a compact URI form designed for human readability. A range of work factors is supported from 2^112 to 2^252.

The plaintext data format consists of an initial header section containing metadata describing the payload, the payload itself and an optional section containing signature data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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 9 April 2026.

Copyright Notice

Table of Contents

1. Introduction

This document describes the Encrypted Authenticated Resource Locator (EARL) URI scheme. An EARL is a bearer token that allows a data object to be located, decrypted and authenticated using a compact URI form designed for human readability.

This document specifies three URI schemes using EARL construction and resolution:

* earl: A generic URI scheme for exchange of any type of data.

* contact: An application specific URI scheme for exchange of contact card information (e.g. JSContact or vCard information).

* device: An application specific URI scheme for exchange of device description information.

The processing and resolution schemes for generic and application specific schemes are identical. Use of an application specific form in a QR code or NFC transmission allows the URI to indicate which type of application should be used to process the associated data.

Additional application specific schemes MAY be defined in the future by registering the URI scheme prefix and specifying this document as the reference.

The following are examples of EARL URIs:

earl://example.com/eluv-woab-g7ih-onix-ybns-qdxk-rzqs
earl:eluv-woab-g7ih-onix-ybns-qdxk-rzqs
jscontact://example.com/eluv-woab-g7ih-onix-ybns-qdxk-rzqs

All three forms refer to the exact same data object. The first two forms are URLs that indicate that the corresponding ciphertext MAY be retreived via HTTPS at:

https://example.com/.well-known/earl/-utAO8IYsdcqmVGk2W15PCLDAFT1HL7M
    fWCWQ-s9qYU.earl

1.1. Construction

An EARL is a URI that contains a UDF value expressed as a Base32D string that specifies a multipurpose key that can be used to locate, decrypt (if necessary) and authenticate an associated data sequence presented as either plaintext or ciphertext.

The data sequence format MAY be the data object itself (verbatim) or the data object MAY be wrapped in an envelope format to provide content metadata or to apply additional security enhancements (signature, encryption) using public key cryptography.

The UDF value is an octet sequence whose leading octets are a Type Identifier indicating the authentication algorithm, the encryption algorithm (if the octet sequence is encrypted) and the data sequence format.

* All the EARLs specified in this document use digests from the SHA-3 family for authentication of the associated octet sequence, to form the locator and for derivation of the key and nonce (for ciphertext data sequences).

* All the encrypted EARLs specified in this document use AES-GCM to encrypted ciphertext data sequences.

* Two data sequence formats are specified, verbatim and wrapped in a DARE Envelope as specified in [draft-hallambaker-dare].

A Type Identifier is a variable length octet sequence in which every octet is odd (bit 0 is set) except for the last. Four Type Identifiers are defined in this document:

[32] Verbatim, Encrypted,

[34] Enveloped, Encrypted,

[80] Verbatim, Plaintext

[82] Enveloped, Plaintext

To prevent recovery of the content payload from the multi-purpose key through brute force attack, use of the enveloped encrypted format is preferred for applications where confidentiality is required. In such cases, the DARE envelope SHOULD include an unguessable salt value that is sufficiently large to achieve the desired work factor.

1.2. Publication

Publication of enveloped data using an EARL is a three-step process:

Derive multipurpose key The multipurpose key is calculated over the DARE envelope octets by means of a first digest function, the result being truncated to a multiple of 20 bits to achieve the desired work factor and the leading octets being replaced by the Type Identifier.

Encryption (optional) The enveloped plaintext data is encrypted under a key derived from the multi-purpose key by means of a key derivation function computed a second digest function, the result being the ciphertext data.

Publish The ciphertext data is published by a mechanism that allows retrieval by means of an authentication token derived from the multipurpose key by means of a third digest function and a locator derived by applying the applying the third digest function to the authenticator, the leading octets of the result being replaced by the Type Identifier and a precision specifier.

The default means of publishing the ciphertext data is through the HTTPS well-known service earl with the path being the locator value in base64url encoding [RFC4648].

This construction establishes the EARL as a ’bearer token’ that may be used to locate, decrypt and authenticate the associated payload and metadata.

1.3. URI Format

The URI is formed according to the usual URI syntax [RFC4648] as follows:

Scheme The scheme component of the URI, this MAY be either earl or an application specific scheme (e.g. contact).

Authority (optional) If the ciphertext data is to be retrieved using the earl well-known service, the authority section MUST be present and specify the host from which the ciphertext data MAY be retrieved.

Path The path value is the multipurpose key in BASE32-D encoding as specified in this document.

Query An EARL URI MAY contain a query component to provide information to the application that uses the EARL. If present, the query component MUST comply with the requirements of [RFC4648] but the interpretation of this data is outside the scope of this document.

Since the recovered payload is authenticated by means of the multi-purpose key, the source from which the ciphertext is obtained is immaterial. Ciphertexts MAY be cached for later retrieval. The double digest construction of the locator allows the result of the first pass to be used to authenticate retrieval requests by a party that does not have the ability to decrypt.

1.4. Resolution

Resolution of an EARL follows the same pattern as construction except that the ciphertext data is decrypted to recover the enveloped data and the resolver MUST verify that the digest value of the recovered enveloped data is consistent with the value of the key.

1.5. Applications

The EARL scheme is designed to support a wide range of applications including:

* A laboratory provides pathology results to a doctor in paper form which are in turn passed to a consultant who requires the results in electronic form for further analysis. Attaching a QR code containing an EARL allows the consultant to obtain the electronic form from the printed document without compromise to patient confidentiality. The paper document is a bearer token that can be exchanged for a paper form of itself

* A device carries a QR code containing an EARL linking to a description of the device for use in onboarding.

* A protocol requires that a trust anchor be passed as a compact URI without reliance on a Trusted Third Party. This is of particular concern when Post Quantum Cryptographic algorithms requiring very large public keys are involved.

1.6. JSContact

One important application of EARLs is to support the exchange of JSContact documents so that the trust context in which the contact is exchanged is preserved. Thus, ensuring that any cryptographic keys or trust anchors have the benefit of that context.

For example, Alice might print a QR code containing an EARL linking to her JSContact data on her business card. When she hands the card to Bob, he can scan the card to obtain Alice’s OpenPGP, SSH, S/MIME etc. credentials.

Use of the JSContact URI scheme for this application allows a camera application reading the QR code to hand off processing of the URI to a contacts application registered as the handler for the JSContact scheme.

Alternatively, if Alice and Bob are unable to meet in person, Alice might publish the JSContact URI as a prefixed TXT entry in her personal DNS Handle @alice.example.com. If Alice’s DNS domain is secured by means of DNSSEC, Bob has a degree of third-party attestation to the binding of the contact data to the domain.

In either case, the JSContact data MAY contain information specifying how updates MAY be received and validated by means of a digital signature contained within the enveloped plaintext data. Once established, the trust relationship is maintained end-to-end between Alice and Bob without the need to rely on any third party.

1.7. Issues to fix before -01

* Present udf value in lower case.
* Update code to produce new examples, in particular, apply prefix to locator, generate examples for all four formats from this draft. DARE signature examples.
* Convert jscontact URI method to contact throughout.
* Remove this section
* Audit the defined terms and ensure that they match those used in the text.
* Github link to the reference code.
* Fix up the section forward references.

2. Definitions

This section presents the related specifications and standards, the terms that are used as terms of art within the documents and the terms used as requirements language.

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Defined Terms

The following words and phrases are used as terms of art with the specified meanings within this document:

Cryptographic Digest Function A hash function that has the properties required for use as a cryptographic hash function. These include collision resistance, first pre-image resistance and second pre-image resistance.

Media Type An identifier indicating how a Data Value is to be interpreted as specified in the IANA registry Media Types.

Data Value The binary octet stream that is the input to the digest function used to calculate a digest value.

Data Object A Data Value and its associated Content Type

Digest Algorithm A synonym for Cryptographic Digest Function

Digest Value The output of a Cryptographic Digest Function

Data Digest Value The output of a Cryptographic Digest Function for a given Data Value input.

Work Factor A measure of computational effort required to perform an attack against some security property.

2.3. Related Specifications

This specification makes use of Base32 [RFC4648] encoding, the SHA-3 [SHA-3] digest function and AES-GCM encryption mode [RFC5288].

The DARE Envelope used to wrap the content payload and metadata is described in [draft-hallambaker-dare]

The URI scheme follows the approach described in [RFC3986].

The resource location scheme makes use of the Well-Known Resource scheme described in [RFC8615].

This work is based on the work originally presented in the Mathematical Mesh UDF [draft-hallambaker-mesh-udf].

2.4. Implementation Status

All the examples in this document were produced using the Mathematical Mesh Reference Library.

3. Architecture

An EARL is a bearer token that allows a data object to be located, decrypted and authenticated using a compact URI form designed for human readability.

The usability approach is based on the Uniform Data Fingerprint proposal [draft-hallambaker-mesh-udf] originally intended to extend and generalize the OpenPGP fingerprint format to allow greater compactness through use of Base32 encoding, algorithm agility by means of an embedded algorithm identifier and allow it to be applied to a wider field of application without the risk of semantic substitution through the incorporation of content metadata.

The UDF format is intended to support a wide range of cryptographic uses including:

* Nonces
* Symmetric keys
* Symmetric Shared Secret values
* Seeds used to generate private keys

These additional applications are outside the scope of this document.

3.1. Uniform Data Fingerprint (UDF)

A Uniform Data Fingerprint (UDF) is a sequence of octets that begins with a type identifier octet sequence.

In cases where a text representation is required, UDF values are presented in Base32 format with dashes separating each group of four characters (Base32D).

UDF values MAY be used as a component of a URI scheme. An EARL is a URI that contains a Multipurpose Key expressed as a UDF value.

The use of UDF values and type identifiers is not limited to EARL, additional applications of the UDF scheme that have been explored on an experimental basis include:

3.1.1. UDF Type Identifier Sequence

Type Identifiers are a sequence of zero or more odd octets (bit 0 is set) terminated by a single even octet (bit 0 is clear). This definition allows for an unlimited number of type identifiers to be defined.

Use of short Type Identifiers of one or two octets requires a Standards Action. Use of longer Type Identifiers (3 octets or more) is unrestricted.

3.2. URI Syntax

Two URI forms are defined: Name Form and Locator Form.

In each case the scheme name MUST be specified. This MAY be the base scheme name ’earl’ or a separately registered sub-scheme name used to specify a particular disposition for the referenced content.

For example, a QR code containing a contact record might specify a scheme name indicating that the linked data is contact data so that the content can be passed to the user’s contacts application.

3.2.1. Name Form

The name form of the EARL URI consists of the scheme name (e.g. earl) and a path value constructed from the Base32-D encoding of the key. For example:

earl:eluv-woab-g7ih-onix-ybns-qdxk-rzqs

The name form of the EARL allows a single URI to be used to decrypt the associated ciphertext and authenticate the resulting payload and metadata. It does not specify the means for retrieving the ciphertext.

3.2.2. Locator Form

The locator form of the EARL URI consists of the scheme name (e.g. earl), an authority section specifying a host name and a path value constructed from the Base32-D encoding of the key. For example:

earl://example.com/eluv-woab-g7ih-onix-ybns-qdxk-rzqs

The locator form of the EARL allows a single URI to be used to retrieve the associated ciphertext, decrypt it and authenticate the resulting payload and metadata.

3.2.3. Type Identifiers

The following type identifiers are currently defined:

Type	Initial	1st, 2nd Digest	3rd Digest	Encryption	Format
[32]	E	SHAKE-256	SHA-3-512	AES-GCM 256	Verbatim
[34]	E	SHAKE-256	SHA-3-512	AES-GCM 256	DARE Envelope
[80]	K	SHAKE-256	SHA-3-512	none	Verbatim
[82]	K	SHAKE-256	SHA-3-512	none	DARE Envelope

Table 1

The Type identifiers are chosen to provide a convenient mnemonic distinguishing ciphertext data sequences (Encrypted) from data authenticated with a digest (Keccak) when presented in Base32.

The digest code point is also chosen to avoid confusion with OpenPGP key fingerprints which can never begin with a K.

3.3. Multipurpose Key Derivation

The enveloped plaintext data is processed with the first digest. The first bytes are replaced by the type identifier byte sequence, and the result is presented as a Base32-dash string with enough 4-character segments to reach the desired work factor.

The Base32-dash encoded form is only used to create the path segment of the EARL URI. The locator and encryption key are derived from the binary form of the multi-purpose key obtained by decoding the key.

The content digest of the envelope shown in the first example is computed using SHAK256:

```
  81 E9 5B 38  01 37 D0 77  35 17 C0 5B  28 0E EA 8E
  61 23 F5 B5  D1 B1 54 91  BC 88 C4 18  29 71 AD 07
```

The first byte of the result is set to the type identifier 34, truncated to the desired precision (140 bits):

```
  22 E9 5B 38  01 37 D0 77  35 17 C0 5B  28 0E EA 8E
  61 23
```

The multipurpose key presentation is the result presented in Base32 with the characters grouped into sets of four with dashes:

```
eluv-woab-g7ih-onix-ybns-qdxk-rzqs
```

Note that since there is an odd number of 20-bit segments in this example, only the upper half of the last byte is recorded in the key:

```
  22 E9 5B 38  01 37 D0 77  35 17 C0 5B  28 0E EA 8E
  61 20
```

The EARL forms consist of the scheme name, authority section (if specified) and the presentation form of the multi-purpose key:

```
earl://example.com/eluv-woab-g7ih-onix-ybns-qdxk-rzqs
earl:eluv-woab-g7ih-onix-ybns-qdxk-rzqs
jscontact://example.com/eluv-woab-g7ih-onix-ybns-qdxk-rzqs
```

3.3.1. Nonces

If a low entropy payload is encoded, an attacker might be able to recover the payload content through a brute force attack. To prevent this attack, the metadata segment SHOULD include a nonce property containing a string containing a sufficient degree of unguessable information.

If a different nonce is specified in the metadata:

```
{
  "Nonce": "NCZ2-C6BE-IFM3-GHDN-LFDD-XZZ4-76KP",
  "cty": "text/plain"}
```

A different multi-purpose key is derived:

```
earl://example.com/eknc-x6cs-de3a-25vb-73si-2k6x-ni4a
```

3.4. Encryption

The enveloped payload is encrypted under a key and nonce derived from the multi-purpose key using the key derivation function.

Since the encryption algorithm is AES-GCM with a 96 nonce and a 256 bit key, it is necessary to generate 44 bytes to encrypt the enveloped data.

The key derrivation function, SHAKE-256 is applied to the multi-purpose key of the first example to obtain 44 bytes of output:

```
3C BA 88 A8  5B AC 99 E1  E5 D2 A3 28  40 9E F2 67
A9 D3 C2 0B  24 04 7B B0  C4 30 63 EA  23 45 B4 F2
DF D7 06 31  50 4D 2D EF  E8 82 79 93
```

The first 32 bytes of the result are the AES-GCM encryption key:

encryption key =

```
3C BA 88 A8  5B AC 99 E1  E5 D2 A3 28  40 9E F2 67
A9 D3 C2 0B  24 04 7B B0  C4 30 63 EA  23 45 B4 F2
```

The final 12 bytes of the result are the AES-GCM nonce:

encryption nonce =

```
DF D7 06 31  50 4D 2D EF  E8 82 79 93
```

The enveloped plaintext data is encrypted under AES-GCM with the specified key and nonce to produce the ciphertext:

ciphertext =

```
BC 46 D1 67  A2 08 D2 4B  2D F7 27 18  3B 1B 27 EF
88 23 EF E3  68 8C EA 53  A8 FB 0C CB  72 EB D3 11
74
```

3.5. Publication

The ciphertext MAY be published through the HTTPS well-known service earl.

The host for the HTTP service is specified by the authority section of the EARL URI. The final path section is the base64url encoding of the result of passing the multi-purpose key through the locator digest function twice and replacing the initial octets of the result with the Type Identifier or the EARL and a precision indicator byte calculated by dividing the length of the UDF value in bits by 20.

For example, a EARL using the verbatim SHA-3 encoding (type [80]) with a UDF of 140 bits will present a work factor of 2^132, the text encoding of the UDF will consist of seven, four character segments separated by six dashes for a total of 34 characters. Therefore, initial two bytes of the locator digest value will be replaced by the sequence [80, 07].

Including the type identifier and precision indicator in the locator digest allows the UDF of a plaintext EARL to be recovered from the locator digest. This in turn enables the use of the locator digest as an implicit authenticator as described in section XX below.

The first locator value is computed from the multi-purpose key using the locator digest function, in this case SHA-3-256:

locator1 =

```
  59 34 18 96  A1 60 AD 9C  13 DF A4 33  8A 2A 72 DD
  E7 EB DD 3F  6A DC 8D 0D  18 39 EB 51  2F 22 8C 03
```

The locator value is computed from the first locator value using the locator digest function, in this case SHA-3-256:

locator =

```
  59 34 18 96  A1 60 AD 9C  13 DF A4 33  8A 2A 72 DD
  E7 EB DD 3F  6A DC 8D 0D  18 39 EB 51  2F 22 8C 03
```

The locator URI is a HTTPS well-known service in which the final element of the path is the base64url encoded locator value.

EARL =
https://example.com/.well-known/earl/-utAO8IYsdcqmVGk2W15PCLDAFT1HL7M
    fWCWQ-s9qYU.earl

Note that publication using HTTPS is only one possible method of retrieval. Any retrieval method that recovers a plaintext sequence consistent with the UDF authenticator value MAY be used.

3.5.1. Access Authenticator

The first locator value MAY be used to authenticate access to the ciphertext. The first locator value is Base32 encoded without dashes to derive an access authenticator that can be used for password type authentication.

The same string is used as the username if the authentication mechanism requires one to be specified.

access authenticator =
LE2BRFVBMCWZYE67UQZYUKTS3XT6XXJ7NLOI2DIYHHVVCLZCRQBQ

Use of the first locator value is preferred over the multi-purpose key because the first locator value cannot be used to decrypt the ciphertext. Thus, a repository of ciphertext values can be provided with the access authenticator to implement access control without granting the ability to decrypt.

3.5.2. Additional Derived Properties

Use of the multi-purpose key is not limited to the applications specified in this document. The multi-purpose key MAY be used to derive any additional information that is needed in an application.

For example, a wireless device using a QR encoded EARL to provide a device description to be used for onboarding might support use of the multi-purpose key to derive a wireless network identifier and access credentials to enable initial access to the device.

The means by which these parameters are derived is outside the scope of this document, but any such applications MUST ensure that the mechanism employed does not disclose the multi-purpose key or access authenticator values.

3.6. Recovery

If the ciphertext is published as a HTTPS well-known service, recovery of the ciphertext is achieved by performing a HTTPS GET method on the specified host and path.

3.6.1. Decryption and Authentication

To recover the enveloped plaintext data from the ciphertext, the encryption key and nonce are derived from the EARL from the multi-purpose key in the same fashion as before and the content digest of the result verified against the original multi-purpose key.

Applications MUST authenticate the recovered plaintext against the multi-purpose key. This step is necessary even in the case that an authenticated encryption scheme such as AES GCM is used as anyone with knowledge of the multi-purpose key can create a ciphertext with a valid GCM tag.

3.7. Signed and Encrypted Envelopes

The DARE envelope format allows public key cryptography to be used to encrypt and sign the content payload and associated metadata. This security is orthogonal to the encryption and authentication enhancements provided by EARL allowing additional security controls to be provided.

For example, Alice includes a signature verification key in her profile for verifying updates to her contact information. She is also a member of a secret club whose membership is not to be disclosed to anyone else and she uses a separate contact card for this whose payload is encrypted using a service that only provides the decryption means to members. This ensures that Alice’s membership of the secret club is not disclosed even if she accidentally shows the wrong QR code to someone.

3.8. Using the Well-Known Service Form as an Implicit Authenticator.

EARLs provide a means of providing a compact, authenticated link to a static data resource. For example, a JSContact card might use 50-byte EARLs to specify a collection of a dozen 1184 byte PQC credentials without loss of authenticity, but this information is only available to clients that support the EARL format.

This is a particular problem with Web browsers where there is a recurring need to enable integrity checking but only the elements deemed to be most critical support the integrity tag. A self-authenticating URI form that renders content inaccessible to even 0.1% of users is likely to be unacceptable.

Use of the Locator URI to link to authenticated content provides full backwards compatibility. The locator URI can be resolved by any Web browser supporting TLS. Clients that understand the semantics of the earl .well-known service can verify that the content returned is authentic. The probability that the publisher of the document did not intend for the linked content to be authenticated in this way is the same as the probability for an incidental digest collision.

4. Security Considerations

4.1. Confidentiality

EARLs provide a mechanism for exchange of data objects with the option of applying encryption at three different levels:

At the transport layer, retrieving the data sequence via HTTPS. TLS encryption provides protection against traffic analysis attacks by third parties unrelated to the originator, publisher or recipient of the data.

Encrypting the data sequence under the multipurpose key. An EARL with an encrypted data sequence serves as a bearer token for the plaintext data sequence. The set of parties with access to the plaintext data sequence is exactly the same as the set of parties with access to the corresponding EARL.

Encrypting the Content Metadata and Payload within an enveloped data sequence. Encrypting the Content Metadata and Payload allows additional confidentiality controls to be imposed, restricting access to specific parties with access to the EARL.

Each layer provides different security enhancements and these should be considered complimentary rather than being mutually exclusive.

For example, Alice might have two contact cards that she distributes by means of a QR code on her business card, the first intended for a general audience, the second restricted for use by her business associates. Both are published through a cloud service. Use of an encrypted EARL effectively eliminates the risk of disclosure by the cloud service provider: they have no access to the plaintext. Encrypting the content payload in the second contact under a key effectively controlled by a confidential content management system, allows Alice to avoid the risk of unintended disclosure of the contact information by presenting the wrong QR code.

4.2. Integrity

Implementations MUST verify that the recovered plaintext data sequence is consistent with the UDF value specified in the EARL.

Verification of the recovered plaintext data sequence is REQUIRED even if the content metadata and payload is signed.

Failure to perform the verification attack allows an attacker with access to a publication host to perform a range of substitution attacks.

For example, Alice publishes her contact EARL through a service hosted by Mallet. When Bob attempts to retrieve Alice’s latest contact information using a faulty client, Mallet sends him her obsolete contact card that omit the public key credentials she has added to it since. The subterfuge works despite the fact that Alice signed both contacts. As a result, Bob attempts to contact Alice through an insecure channel rather than using the available security controls.

4.3. Availability

The EARL publication service may be unavailable. Deployments SHOULD carefully consider the possibility of service availability being lost due to equipment failure, misconfiguration, insufficiency of resources and external denial of service attack.

4.4. Semantic Substitution

Many applications record the fact that a data item is trusted, rather fewer record the circumstances in which the data item is trusted. This results in a semantic substitution vulnerability which an attacker may exploit by presenting the trusted data item in the wrong context.

The DARE envelope format allows the linked data object to be accompanied by metadata describing the intended semantics. For example, by specifying the IANA media type.

4.5. QR Code Scanning

The applications used to scan QR codes raise security concerns. Scanning a QR code with a poorly designed or implemented application can cause consequences unintended by the user or compromise the end point device itself.

The act of scanning a QR code SHOULD be considered equivalent to clicking on an unlabeled hypertext link.

4.6. User Intentionality

Since QR codes are scanned in many different contexts, the mere act of scanning a QR code MUST NOT be interpreted as constituting an affirmative acceptance of terms or conditions or as creating an electronic signature.If such semantics are required in the context of an application, these MUST be established by secondary user actions made subsequent to the scanning of the QR code.

There is a risk that use of QR codes to automate processes such as payment will lead to abusive practices such as presentation of fraudulent invoices for goods not ordered or delivered. It is therefore important to ensure that such requests are subject to adequate accountability controls.

For example, a payment protocol allows a user to pay money to a person by scanning a QR code. Mallet creates a QR code that causes him to be paid $500 every time it is clicked, he visits local restaurants and pastes his labels over the QR codes used to download the menu.

4.6.1. Malware Vector

QR codes can be used as a means of distributing malware. It is therefore essential that all applications processing content obtained by means of an EARL perform all the usual precautions against maliciously constructed content including checking for buffer overrun conditions, integer overflow and underflow, semantic substitution attacks, etc.

For example, Mallet knows that Alice is using an outdated browser with a vulnerability that allows a maliciously constructed image file to cause a stack buffer overflow. He presents his contact card to Alice with a link to maliciously constructed content exploiting this vulnerability to gain control over Alice’s device.

5. IANA Considerations

Registrations are requested in the following registries:

* well-known URI registry

* Uniform Resource Identifier (URI) Schemes

In addition, the creation of the following registry is requested: Uniform Data Fingerprint Type Identifier Registry.

5.1. Well Known

The following registration is requested in the well-known URI registry in accordance with [RFC5785]

URI suffix earl

Change controller Phillip Hallam-Baker, phill@hallambaker.com

Specification document(s): [This document]

Related information

5.2. URI Registration

The following registration is requested in the Uniform Resource Identifier (URI) Schemes registry in accordance with [RFC7595]

Scheme name: earl

Status: Provisional

Applications/protocols that use this scheme name: TBS

Contact: Phillip Hallam-Baker mailto:phill@hallambaker.com

Change controller: Phillip Hallam-Baker

References: [This document]

5.3. URI Registration

The following registration is requested in the Uniform Resource Identifier (URI) Schemes registry in accordance with [RFC7595]

Scheme name: jscontact

Status: Provisional

Applications/protocols that use this scheme name: TBS

Contact: Phillip Hallam-Baker mailto:phill@hallambaker.com

Change controller: Phillip Hallam-Baker

References: [This document]

5.4. Uniform Data Fingerprint Type Identifier Registry

This document describes a new extensible data format employing fixed length version identifiers for UDF types.

5.4.1. The name of the registry

Uniform Data Fingerprint Type Identifier Registry

5.4.2. Required information for registrations

Registrants must specify the Type identifier code(s) requested, description and RFC number for the corresponding standards action document.

The standards document must specify the means of generating and interpreting the UDF Data Sequence Value and the purpose(s) for which it is proposed.

Since the initial letter of the Base32 presentation provides a mnemonic function in UDFs, the standards document must explain why the proposed Type Identifier and associated initial letter are appropriate. In cases where a new initial letter is to be created, there must be an explanation of why this is appropriate. If an existing initial letter is to be created, there must be an explanation of why this is appropriate and/or acceptable.

5.4.3. Applicable registration policy

Due to the intended field of use (human data entry), the code space is severely constrained. Accordingly, it is intended that code point registrations be as infrequent as possible.

Registration of new digest algorithms is strongly discouraged and should not occur unless, (1) there is a known security vulnerability in one of the two schemes specified in the original assignment and (2) the proposed algorithm has been subjected to rigorous peer review, preferably in the form of an open, international competition and (3) the proposed algorithm has been adopted as a preferred algorithm for use in IETF protocols.

Accordingly, the applicable registration policy is Standards Action.

5.4.4. Size, format, and syntax of registry entries

Each registry entry consists of an integer code.

5.4.5. Initial assignments and reservations

The following entries should be added to the registry as initial assignments:

Code	Description	Reference
34	EARL-AES-SHA3	[This document]
104	Nonce	[This document]

Table 2

6. Acknowledgements

7. Appendix A: Base32-D

Base32 in lower case with dashes between blocks of 4 characters, no padding.

The output is always an integer multiple of 20 bits.

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	a	9	j	18	s	27	3
1	b	10	k	19	t	28	4
2	c	11	l	20	u	29	5
3	d	12	m	21	v	30	6
4	e	13	n	22	w	31	7
5	f	14	o	23	x		
6	g	15	p	24	y		
7	h	16	q	25	z		
8	i	17	r	26	2		

8. Appendix B: UDF Type Identifier Encoding

A Uniform Data Fingerprint is a Base32-D encoded binary string in which the initial octets specify the

9. Normative References

[draft-hallambaker-dare]
Hallam-Baker, P., "Data At Rest Envelope (DARE)", Work in Progress, Internet-Draft, draft-hallambaker-dare-00, 6 October 2025, <https://datatracker.ietf.org/doc/html/draft-hallambaker-dare-00>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/rfc/rfc5288>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[SHA-3] Dworkin, M. J., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", August 2015.

10. Informative References

[draft-hallambaker-mesh-udf]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-19, 14 October 2024, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-udf-19>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/rfc/rfc5785>.

[RFC7595] Thaler, D., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

Author’s Address

Phillip Hallam-Baker
ThresholdSecrets.com
Email: phill@hallambaker.com