

YURLs

YURLs are URLs that obey the "y-property".

This page is a cut-and-paste of the content of the following list of links, some of which have had to be retrieved from the Wayback Machine. In due course, it will be rewritten in my own words, to get around the notional copyright in the original material.

- [HTTPS - Leave the Certificate Authority Behind](#)
- [What Does the 'y' Refer to?](#)
- [Decentralised identification](#)
- [Securing the Unikernel](#)
- [HTTPS URL scheme](#)
- [Norman Hardy on the YURL](#)
- [Not One Click for Security \(HP Labs\)](#)

The key thing is the HTTPS URL, or YURL. This has the format:

```
httpsy://algorithm:fingerprint@domain:port/path1/!redactedPath2/...
```

Note that this is incompatible with Gemini as a result of the ban on the use of the authority section in URLs following agitation on the Gemini mailing list.

Leaving the CA behind

HTTPS: a proposal for a protocol that eliminates the need for certificate authorities in many cases and enables placing sensitive information in a bookmarkable link. Also enables the creation of secure bookmarkable OAuth bearer tokens.

Discussion notes, key understandings, outstanding questions, observations, and, if appropriate to this discussion: action items, next steps: Draft format of the new httpsy protocol proposal:

```
httpsy://algorithm:fingerprint@domain:port/path1/!redactedPath2/...
```

the protocol is httpsy. The algorithm is used to interpret the fingerprint, for example, "sha-256". The fingerprint of the public key is used to challenge the server to prove that he is the holder of the public key, to foil DNS cache poisoning and similar attacks. Any part of the path prefixed with a bang "!" is redacted when the url is displayed in the window by the browser, in the referrer header, and in server logs.

Controversy over whether this improves the user's situation with respect to phishing or makes it worse: on the one hand, the domain that people look at to see where they are is buried in a long string of gibberish, on the other

hand, it can be claimed that the use of the domain to determine your location, in a world with millions of sites, necessarily not humanly distinguishable, is the source of the problem, not the solution.

System eliminates need for certificate authorities in many circumstances, the self-signed cert is adequate to prove that, if someone you trust gives you a link, you are guaranteed when you click the link to arrive at the place the trusted party intended for you to go.

Concern raised about untrustworthy parties sending you to untrustworthy places, but they can do that today anyway.

Often requires a “trust on first use” pattern similar to what you do with ssh.

Does not solve the problem with reliably going to a place that you saw on a billboard, since the billboard must be completely memorable.

The redacted parts prefaced with a bang can hold credentials, turning these links into unguessable self-authorizing links, suitable for use both as bookmarkable webkeys and as oauth bearer tokens.

Alan Karp and Marc Stiegler are leading a group that meets on Friday mornings, with people from HP, Google, PayPal, and others, to develop an RFC spec for httpsy.

What does the ‘y’ refer to?

Abstract

A YURL is a URL that enables communication with a site determined solely by the creator of the URL. [URL]

Overview

The most fundamental primitive of distributed messaging is the sending of a message to a remote site. The message target is determined by an identifier. One site is introduced to another by receiving such an identifier. Fig. 1 demonstrates this concept.

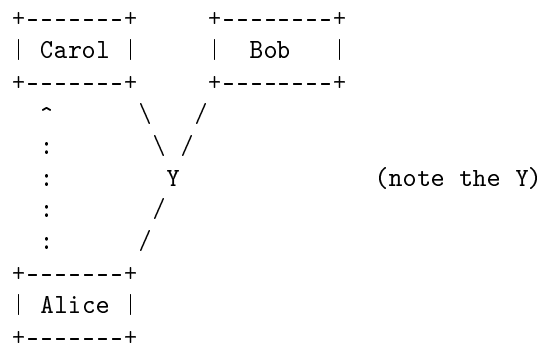


Fig. 1: A Granovetter Introduction

In Fig. 1, Alice introduces Carol to Bob. The arrows represent identifiers and the circles represent sites. The short fat arrow represents a message. The sociologist Mark Granovetter originally developed diagrams of this type to illustrate how the topology of interpersonal relationships changes over time, as people introduce acquaintances to one another. [Granovetter]

Granovetter diagrams are a powerful tool for understanding how distributed messaging networks evolve. [Ode] For example, Fig. 1 is also an accurate depiction of the use of OIDs in CORBA, the use of channels in the pi-calculus, and the use of URLs in the WWW. The Granovetter diagram is applicable to any messaging network in which sites lack omniscience.

The Granovetter diagram shows that in the absence of omniscience, a site's connection to a target site is determined by the site that performed the introduction. For example, in Fig. 1, Bob becomes connected to Carol because Alice introduced Carol to Bob. If Alice had instead introduced Dave to Bob, Bob would be connected to Dave instead of Carol. Since Alice decides which identifier to send to Bob, Alice determines what identifier Bob will use when sending messages. Alice determines the target of messages sent by Bob.

This fundamental property of introduction is highlighted in Fig. 1 by the two bold arrows that form a 'y' shape. A YURL is an identifier that allows Bob to rely on this y-property of introduction not being violated. Bob can then be confident that his messages are only delivered to the target that Alice specified.

As a concrete example of the importance of the y-property, consider the case where Alice is a bank, Bob is a customer and Carol is Bob's bank account.

Description

Enforcing the y-property depends on the features of the identifier for the introduced site. In Fig. 1, this identifier is the short bold arrow, pointing to Carol. To enforce the y-property, the identifier MUST provide enough information to: locate the target site; authenticate the target site; and, if required, establish a private communication channel with the target site. A URL that meets these requirements is a YURL.

The y-property

Briefly stated, the y-property is: "The introducer determines the message target."

The y-property means that only the introducer has the privilege of determining the recipient of a message sent to the introduced site and the processor of the sent message. The introducer is the site authorized to write to a communication channel read by a client site. The introducer uses the communication

channel to provide an identifier to the client site. The identifier identifies the introduced site. The introduced site is a site selected by the introducer. The client site is the site that receives the identifier and uses it to send a message to the introduced site. Receiving a message means having access to the plaintext of the message. Processing a message means producing a response message which the client site will accept as an authentic response to the sent message.

The y-property is the result of applying the principle of least privilege to the fact that the introducer decides which site to introduce.

A YURL is a URL

The term YURL refers to a subset of URL. Like a URL, a YURL **MUST** provide the information required to open a communication channel to the target site. The YURL may locate the target site by directly giving its network location or by specifying a locating service which can ascertain the target site's current network location. The use of a locating service is encouraged.

Site authentication

A YURL **MUST** provide all the information required to authenticate the target site. Authentication of the target site **MUST ONLY** rely on information contained in the YURL. If any outside information were used for authentication, the creator of that information would have power to determine the target of sent messages, violating the y-property. In particular, any URL scheme that depends on the PKI for authentication, such as https, is not a YURL.

Private communication

The creator of a YURL may wish to be the sole recipient of a message, or may wish that the message be available to unspecified others. The message target may be public to enable caching. If the message target is private, the YURL **MUST** provide a means to establish a private communication channel with the target site. If a private message is not sent over a private communication channel, an eavesdropper could receive the message, violating the y-property.

The choice of whether a message target is public or private **MUST ONLY** be made by the creator of the YURL. A decision to not use a private communication channel **MUST ONLY** be made based on information contained in the YURL, or obtained from the authenticated target site.

Threat model

- Adversary motivation: The adversary is trying to violate the y-property.
- Adversary capabilities: All sites are located on a public network. The adversary can intercept and modify any packet after it has left a site.

Decentralised Identification

The WWW is already tantalizingly close to providing a decentralized infrastructure for representing trust. Through the use of prose and hyperlinks, humans can express highly refined trust relationships between entities on the WWW. For example, "I've completed several satisfactory transactions through e-gold." Accurately expressing these relationships is one of the primary aims of human language. Software agents can similarly express trust relationships using their equivalent of prose, the application protocol.

Reliance upon the DNS is the stumbling block that prevents these trust relationship languages from achieving the "weblike, decentralized" qualifier. This point too was noted by Tim Berners-Lee in "Weaving the Web":

"For all its decentralized growth, the Web currently has one centralized Achilles' heel by which it can all be brought down or controlled."

When you dereference a URL, the keepers of the DNS and PKI determine which web server responds to your request. To make the WWW a "weblike, decentralized infrastructure" for representing trust, we need a way for WWW entities to directly link to each other. YURLs, such as httpsy URLs, enable linking without indirection through centralized authorities like the DNS or PKI.

Since referring to other entities is such a common task, its mechanisms are often overlooked. Many also assume that the vagaries and failings of existing identification mechanisms are intrinsic to the task and therefore inevitable. These assumptions are often wrong. For example, the "phishing" attacks that currently plague the WWW are not intrinsic to online communication; they are a symptom of a poorly conceived identification mechanism (see Trust Management for Humans). The documents presented here explore what referral is, and how the WWW can be adapted to do it well.

White papers

- [Naming vs Pointing](#)
- [An introduction to introduction](#)
- [Things that are similar to YURLs \(i.e., PGP, SSH\)](#)
- [Why use YURLs?](#)

Naming vs Pointing

The problems with a name-centric designation model, like the PKI, are explained using an analogy.

The scenario

Alice has brought her son, Jon, to a party. Also at the party is Libby, a friend of Alice's. Jon and Libby have not met. Many other people are at the party.

The PKI analogy

In a PKI-like model, Alice introduces her son to Libby by telling Libby that her son is named "Jon". Libby then looks around for someone who claims to be named "Jon". Upon finding such a person, Libby asks the person for their driver's license, to confirm that the person's name is indeed "Jon".

The YURL analogy

In a YURL-like model, Alice introduces her son to Libby by pointing at him and telling Libby: "This is my son."

The introducer role

In both cases, Alice is the introducer, and controls which person Libby comes to know as Alice's son. In the PKI-like introduction, Alice can give any name she wants to Libby. If Alice says her son's name is "David", Libby comes to know David, not Jon, as Alice's son. Similarly, in the YURL-like introduction, Alice can point at whomever she chooses.

Problems with naming

Using names and licenses creates a number of potential problems in a PKI-like introduction that are not present in a YURL-like introduction.

Trust

Libby must trust the Department of Motor Vehicles (DMV) to be meticulous about putting the right name on an issued license.

Entry barrier

If Jon is underage, the DMV will not issue him a license.

Scalability

Jon may normally be resident in a foreign country. For Jon to have a license recognized by Libby, either the DMV must be able to issue a license to a foreigner, or Libby must be able to recognize a license issued by a foreign DMV. Both cases present difficult scalability problems. In the former case, the DMV must be able to accurately establish the name of any person, from

anywhere in the world. In the latter case, Libby must trust a license issued by any state in the world.

Name conflation

When Libby is looking for a person named "Jon", Jan might say: "Hi, I am Jon." When Libby checks Jan's license, she is expecting it to say "Jon" and does not notice that it actually says "Jan".

Fake identification

Mallory works at the DMV, and likes playing party tricks. Knowing that Alice is bringing her son to the party, Mallory makes himself a fake license. The fake license is indistinguishable from the real thing. When Libby comes asking for a person named "Jon", Mallory can claim to be "Jon" and can prove it with a license.

Identity revocation

The DMV may decide that another person is the rightful user of the name "Jon" and revoke Jon's license. Anyone looking for Jon will instead find the new licensee.

Time

A license is only valid for a limited duration of time. When a person's license expires, they must go to the DMV and get a new one issued. This process can be time consuming.

Money

Issuing a license is hard work, and the DMV will not do it for free. Everyone who gets a license must pay a fee to the DMV.

Identity loss

Due to the cost and difficulty involved in issuing a license, the DMV must issue licenses that are valid for years. Over such a long duration, people are bound to lose their license or have it stolen. As a result, a person using a license may not be the legitimate holder of the license.

Conclusion

A name-centric designation model is fraught with problems. Pointing is often an easier and safer solution than naming. A YURL enables pointing on the Internet.

An introduction to introduction

Abstract

On the WWW, introduction takes place through the use of URLs. A page "introduces" you to another site by providing a hyperlink containing a URL for the other site. An introduction happens every time you receive a URL.

When you visit a WWW site, your browser "authenticates" the site. Authentication only means verifying that the site responding to your request is the same site identified by the introduction URL.

Authentication in the WWW is currently implemented using https:// URLs. This primer explains the current implementation and introduces a better approach: the YURL.

The status quo: https

When you click on an https:// URL, your browser establishes a secure connection to the identified site before requesting the document. If the secure connection is successfully created, your browser displays a small lock icon in the lower corner of its window and loads the document. If your browser cannot authenticate the identified site, it will interrupt your browsing with a pop-up dialog notifying you of the failed authentication.

Fig. 1: A failed authentication alert. (omitted)

The certificate

To authenticate a site, your browser performs checks on a certificate that it receives from the site. A certificate is a signed document containing a public key and information about the holder of the corresponding private key.

Every secure site has a public/private key pair. Your browser uses a site's public key to encrypt a request to the site. The site uses its private key to decrypt the request. Anyone can encrypt a request using a public key, but only the holder of the private key can decrypt the request. A public/private key pair is like a locked mailbox: anyone can insert mail, but only the mailbox owner can remove mail.

A public/private key pair is a digital identity. https:// authentication uses a certificate to match a URL to a digital identity.

The three checks

Authenticating a target site means verifying that the provided public key belongs to the site identified in the URL. Your browser performs three main checks, listed in the dialog shown in Fig. 1, to complete the authentication.

A certificate names the owner of the provided public key. Your browser verifies that the owner name is the same as the domain name in the URL. For example, if you click on a link with URL `https://www.waterken.com/`, your browser will verify that the provided certificate names "www.waterken.com" as the owner of the provided public key. This is the third check listed in the dialog in Fig. 1.

A certificate is only valid for a specified period of time. Your browser verifies that the provided certificate has not expired. This is the second check listed in the dialog in Fig. 1.

Anyone can compose a document containing a public key and naming an alleged owner. A certificate is only useable if you trust its creator. A certificate creator is called a "Certificate Authority" or CA. A CA has its own public/private key pair used for signing certificates. The private key is used for signing and the public key is used for checking the signature. Your browser has a collection of CA public keys built into it. Your browser assumes that you trust all of these CAs. When you visit a site, your browser verifies that the provided certificate was signed using one of these built-in public keys. This is the first check listed in the dialog in Fig. 1.

If all three of these checks pass, the browser trusts the CA's claim that the provided public key belongs to the site identified in the URL. The identified site has been authenticated.

What does a CA do?

A CA checks the veracity of information in a certificate and signs it. When purchasing a certificate from a CA, you send the CA your domain name, your mailing address, and your public key. The CA verifies that you are the legitimate owner of the domain name. If so, the CA issues you a signed certificate.

When you purchased your domain name, you were given an entry in the Internet's WHOIS database. This entry lists your domain name and mailing address. Essentially, a CA lets you additionally associate your public key with this information. The checks performed by the CA only ensure that you are the same person for whom they created the WHOIS database entry. One may reasonably ask why the CA did not ask for the public key during the original registration process.

What does a CA not do?

Checking that you are the legitimate domain name owner is the only check performed by the CA. In particular, the CA does not vouch for your character or good standing. Nor does the CA vouch for the security of your site's software or setup. The CA in no way vouches for the content provided by a

site. A site may provide fraudulent content from a domain name that it legitimately owns. Successful authentication does not mean that the user cannot be deceived about the true author of a document.

When comparing authentication solutions, keeping in mind the exact nature of the provided service is crucial to an accurate evaluation. Authentication verifies the site you were actually introduced to, not the site you thought you were introduced to.

A better way

The combination of public key cryptography and CAs is called the Public Key Infrastructure, or PKI. This name gives the impression of a large and important bureaucracy, which the PKI is. Like other bureaucracies you may be familiar with, the PKI is expensive, unwieldy, and error-prone. Might there be a better way of using public key cryptography that does not entail a bureaucracy like the PKI? Yes, a widely used and trusted way already exists; it just has not been applied to the WWW yet.

Every public key has a unique fingerprint. Creating a new public/private key pair with the same fingerprint as an existing public key is mathematically impossible. Instead of using the PKI to match a public key to a URL, the public key fingerprint is included in the URL. The browser verifies that the fingerprint in the URL matches the public key provided by the visited site. Certificates and Certificate Authorities are unnecessary.

A URL that uses this authentication technique is called a YURL. For a more detailed understanding of YURLs, read "What Does the 'y' Refer to?". For a comparison of YURLs to other widely used and trusted software, read "Things that are Similar to YURLs". For a detailed examination of the technical advantages of YURLs over the PKI, read "Why Use YURLs?". To understand how YURLs integrate with typical WWW browsing, read "Trust Management for Humans". Try using YURLs with the Waterken™ Browser.

Things that are similar to YURLs

PGP

In a PGP introduction, you provide a future correspondent with your email address and your PGP fingerprint. Later, the correspondent will download your key from a keyserver and authenticate it by checking that the fingerprints match. An email is encrypted to the authenticated public key before being sent to the target email address. Since you are the only one holding the corresponding private key, this process ensures that you are the only one who can receive the plaintext of the email.

The PGP message sending procedure enforces the y-property. Authentication of the message target is done relying solely on information obtained in the

PGP introduction. The PGP fingerprint is the sole source of authentication information. PGP encrypts the email, providing a private communication channel. The target site is located using the email address.

The httpsy scheme uses the same implementation techniques found in a PGP introduction. The key-id fulfills the same function as the PGP fingerprint. The host fulfills the same function as the email address.

SSH

When first connecting to an SSH server, the user is presented with output like:

```
The authenticity of host 'mail.waterken.com (209.88.68.62)' can't be established.  
RSA key fingerprint is 8c:4c:3b:56:4a:26:58:48:af:e3:2c:6d:f9:5f:0b:77.  
Are you sure you want to continue connecting (yes/no)?
```

At this point, the user is supposed to verify that the RSA key fingerprint matches that of the installed SSH server. This information should be provided by the sysadmin who created the user's account on the server. If the fingerprints match, the RSA key is accepted and stored in the user's .known_hosts file. For all future connections, the SSH client software will verify that the server provided RSA key matches the previously stored key.

The SSH connection procedure enforces the y-property. Authentication of the server is done relying solely on information obtained in the introduction performed by the sysadmin. The RSA key fingerprint is the sole source of authentication information. Future connections leverage a cache, the .known_hosts file, containing only information derived from the introduction. SSH encrypts all traffic, providing a private communication channel. The target site is located using the server's domain name.

Both SSH and httpsy implement the y-property by completing a cryptographic handshake using the hash of the target site's public key instead of the PKI.

Why use YURLs?

Principle of least privilege

The principle of least privilege is the main constraint in the design of secure software. This constraint requires that a task be completed using the minimum distribution of privilege. As explained by the y-property, only the creator of a URL requires the privilege of determining the target of the URL. A YURL meets this constraint, the PKI does not. The PKI additionally gives the Certificate Authority (CA) the privilege of determining the target of the URL.

To understand the gravity of PKI's violation of the principle of least privilege, examine the list of CA certificates in your web browser. This list is commonly upwards of 50 long. All of these participants have the privilege to impersonate

any site. If the private key for any one of these certificates is not perfectly protected, an attacker can impersonate any site.

Certificate lifecycle control

Today, the unfortunate reality is that servers get hacked. When your server gets hacked, your private key is exposed and no longer useful for authentication. To return to business, you must revoke the old certificate and get a new certificate issued. When using the PKI, you must pay your CA for this service. When using YURLs, you can provide this service for yourself. When using YURLs, you are your own CA. You have complete control over the creation of certificates.

Due to the high cost of CA services, sysadmins infrequently change a server's certificate. Increasing the lifetime of a certificate increases your vulnerability to identity theft. The longer the certificate is valid, the longer the private key must be protected. When using YURLs, sysadmins can shorten the lifetime of a certificate, change keys more frequently, and thus reduce their site's vulnerability to identity theft. Keys could even be changed at a frequency that would enable the site to forgo certificate revocation and Certificate Revocation Lists (CRLs).

Name ownership

A PKI based URL, like https, identifies your site using a domain name. Your right to use a domain name is governed by a non-democratic central authority, the IANA. You do not own your domain name, and you may lose the right to use it. In this event, a new owner of the domain name can put his own resources at the URLs that formerly identified your resources. Your resources would no longer be accessible.

A YURL, like httpsy, identifies your site using your CA public key fingerprint. You own your CA fingerprint. Taking away ownership of your CA fingerprint requires determining the corresponding CA private key. Determining a private key based on knowledge of the public key is mathematically impossible. No organization controls the creation of public/private key pairs. You do not need anyone's permission to acquire an identity, and no organization can reclaim your identity.

Monopoly rents

Despite the large number of CA certificates in the web browser, one company has a monopoly on selling certificates to the public. This company charges nearly \$1000 for a certificate. When using YURLs, you issue your own certificates. No third party demands a fee.

Paying monopoly rents may be a tolerable situation when you only need limited services, but the situation becomes untenable when your needs expand.

A single certificate for your entire WWW site may be affordable, but purchasing certificates for smaller collections of resources, like each web service, will quickly become unaffordable.

These economics typically result in many web services being bundled under the same domain name and SSL certificate. This solution creates a difficult and expensive performance bottleneck, as requests to all hosted web services must be routed through the same domain name. Since the same SSL private key authenticates all bundled web services, the security of each web service is not independent. This security vulnerability grows linearly with the number of bundled web services.

When using YURLs, you can give each web service its own authentication identity. This solution provides simple and cheap load distribution, as each web service can be independently migrated between computers. The security of each web service can be made independent, reducing vulnerability and facilitating management and security auditing.

When to use the PKI

Storing authentication information in the URL makes it difficult to memorize. If your URLs must be human memorable, the PKI is an appropriate choice for authentication.

Rarely are humans required to memorize a URL. Software is almost always used to remember the URL for the user. Before deciding to use the PKI, be sure that your application has this requirement. If it does, reevaluate your user interface before committing to the PKI. Using a hyperlink is always better than a memorized URL.