How to Leverage an fTLD Domain:

TECHNICAL GUIDE TO SECURITY REQUIREMENTS

fTLD
Registry Services, LLC

TABLE OF CONTENTS

TECHNICAL GUIDE TO SECURITY REQUIREMENTS

INTRODUCTION

The fTLD Registry Services' ("fTLD") Technical Guide to Security Requirements (the "Requirements") is designed for technical staff and third-party providers of financial organizations and provides detailed guidance on implementing the Requirements for a .BANK or .INSURANCE domain name (collectively "fTLD Domain"). This guide describes the Requirements that must be deployed by registrants (organizations that register domain names) and registrars[1] (entities authorized by fTLD to offer fTLD Domains) in order to comply with the Requirements and fully benefit from their fTLD Domains.

The following are related Guides to this document and available at `ftld.com/guide`:

- Executive Guide to Security Requirements is designed for Chief Information, Technology and Information Security Officers (i.e., CIOs, CTOs, CISOs) and provides an overview of the Requirements and the benefits of an fTLD Domain.

- Planning and Communications Guide is designed for officers, marketing managers and others; it focuses on strategies for planning and communicating your organization's move to an fTLD Domain.

The Requirements can be accessed at `ftld.com/enhanced-security`.

fTLD has compiled a list of publically available, free resources that can be helpful in understanding whether the implementation of an fTLD Domain addresses the defined Requirements, and it's available at `ftld.com/domaincheck`. Although these resources are useful, they are not the basis for fTLD's Requirements monitoring service and are not exact checks against the Requirements.

Why the Additional Requirements?

An fTLD Domain is a trusted, verified, more secure and easily identifiable location on the internet for the global banking and insurance communities and the customers and stakeholders they serve. fTLD Domains have been designed and built by the international banking and insurance communities specifically for banks, insurers (inclusive of providers and distributors) and others in the financial services sector. The goal of these internet domains is assuring greater online security for financial services organizations and their customers. Owned, operated and governed by the financial community, an fTLD Domain is where organizations can enhance their online presence, create new marketing and branding opportunities, differentiate themselves in a competitive market, secure shorter and more relevant and memorable domain names and communicate more securely with customers, third-party providers and other stakeholders.

To achieve and sustain the overall goal of a trusted and more secure place online, registering and operating an fTLD Domain name requires a greater commitment to security than in most other internet web extensions or Top-Level Domains (TLDs). fTLD requires its registrars and registrants to deploy robust security technologies and practices. We also ensure that these measures are routinely monitored for compliance with the Requirements. When fTLD's monitoring service detects

[1] See the list of Approved Registrars at `https://www.ftld.com/approved-registrars`

a compliance issue the respective registrar and/or registrant will be contacted about an appropriate remediation plan.

Broadly, the Requirements fall into a set of related specifications:

- **Digital Assertion, Verified Digital Identity and Multi-Factor Authentication**—following the eligibility verification process, eligible fTLD Domain participants will be issued multi-factor authentication credentials, which are required to change domain registration data via an online process with the respective registrar. An equivalent multi-factor authentication process will be used by registrars that manage this process in an offline setting.

- **Domain Name System Security Extensions (DNSSEC)** to ensure that internet users are landing on legitimate websites and not being misdirected to malicious ones.

- **Email Authentication** to ensure brand protection by mitigating spoofing, phishing and other malicious email-borne activities.

- **Strong Encryption (i.e., Transport Layer Security)** to ensure confidentiality and integrity of communications and transactions over the internet. TLS is an internet standard for providing encrypted communications over the internet. fTLD Domain participants are restricted from implementing known weak or vulnerable versions of TLS protocol and cipher suites.

- **DNS Resource Records Restrictions** require fTLD Domain registrants to comply with restrictions on authoritative name server Fully Qualified Domain Name (FQDN) and Domain name system (DNS) resource records.

    - The FQDN of authoritative name servers for an fTLD Domain must end in an fTLD Domain (e.g., `ns1.bankname.bank`, `ns1.insurername.insurance`).

    - DNS resource records (e.g., CNAME, DNAME, MX) may be used to alias to out-of-zone domains (i.e., non-fTLD Domains) and are subject to compliance the relevant Requirements.

Requirements may need to change to meet the evolving landscape of risks and threats on the internet. fTLD intends that its Requirements—and likewise this guide—will change to stay ahead of the threats it is designed to mitigate.

DIGITAL ASSERTION AND VERIFIED DIGITAL IDENTITY

What is Digital Assertion?

Digital assertion is a process by which a network service ("verifier") authenticates a user and generates an assertion about the result of the authentication to another network service that relies on that authentication to determine if services should be provided. Digital assertion allows for an individual or organization to assert its identity, have a means to prove that identity, involve a trusted organization to authenticate that identity and then assert that identity to gain access to an authenticated session. Often the assertion is a time-based token that acts as a credential for access to network systems.

What is the fTLD Domain Security Requirement for Digital Assertion?

Requirement #22:

> **Registration Authorities must establish digital assertion, or an equivalent process, during the registration process.**

In particular, Registration Authorities (i.e., registry operator and registrars) must provide an adequate description of their requirement for digital assertion, or equivalent process using best current practices, and how it will be applied to registrars and registrants.

How Does Digital Assertion and Verified Digital Identity Work?

While the internet was originally designed with a model of anonymity for its transactions, this is not appropriate in an fTLD Domain environment. Identity management involves establishing who is using a service and digital assertion and verified digital identity are techniques to enhance the level of trust in knowing who is using it.

Digital Identity Management

The fTLD Domain requirement mandates a strong approach to authentication. This is crucial in an fTLD Domain environment as a recent study highlighted that nearly 82 percent of cybercrime incidents targeted user credentials.

It's critical to understand that the digital identity management requirements affect registry operator-registrar and registrar-registrant interactions. Interactions with customers are not affected by this requirement.

One approach to improving on the common, but not robust, username and password combinations that are so familiar, is to introduce a second factor into the authentication process. Common approaches include a registration and verification component, the issuing of credentials and then options for a second-factor to be introduced. Thus, the user would possess the credentials provided after registration and, for the purpose of authentication a second-factor would be required. Examples of second-factors are:

- Random number generation tokens
- Data from a text message or email
- Supporting mobile app for a smartphone or tablet
- Other soft or hard tokens
- Biometrics

In this approach, the two factors are often referred to as "something you have and something you know."

A typical example is an authentication token that displays a time-limited random number. The combination of the personal information and the number displayed on the token constitute the two factors. Including two elements makes it more difficult for an unauthorized person to access the account because they would be required to know the personal information and have the access token as well.

According to research, multi-factor authentication[2] can drastically reduce the incidence of online theft and other network-based fraud because access or theft of a password alone is not enough to gain access to online services or information.

Digital Identity and Digital Assertion Deployment for Registry Operator and Registrants

The fTLD Domain requirement does not specify a particular technology or methodology for meeting the digital assertion condition, but does provide that it must meet National Institute of Standards and Technology (NIST) level three or better. To meet the requirement, the registration authorities must provide an adequate description of their policy for digital assertion or an equivalent process. The policy should use best current practices and describe how it will be applied to registrars and registrants.

[2] See https://en.wikipedia.org/wiki/Multi-factor_authentication

Such a policy must contain a description of:

- The registration system used to allow a user of the system to apply for credentials;
- The verification system used to make an assessment of whether or not credentials should be granted based on the information provided by the registration system;
- The registration reporting system that reports and audits the activity in the combined registration/verification system;
- A credential granting service for providing single-factor credentials to a user;
- An authentication system that combines the single-factor, long-term credentials with a second-authentication factor which is a single use and/or limited time authenticator; and
- An identity assertion service that allows assertions about digital identity to be passed between security domains within the registry/registrar/registrant environment.

While some in the registry/registrar environment have already developed a digital identity approach, it may be new to others. Multi-factor authentication is becoming easier to implement. In addition, multi-factor authentication (especially the second-authentication factor) is becoming easier to outsource. Network equipment vendors, telecommunication providers and specialist digital identity management companies have emerged in this product space and offer solutions that can be integrated with existing registration/credential granting systems.

DNSSEC

What is DNSSEC?

The original design of the DNS was almost completely focused on data availability and did not include a security component. Fundamentally, DNSSEC solves two crucial problems related to the DNS:

1. Adds a data integrity check to the DNS; and

2. Adds the capability to authenticate the trust chain of data provided by the DNS.

DNSSEC also provides a platform for future security technologies.

DNSSEC is a backward-compatible technology that extends the DNS by providing origin authentication of DNS data, data integrity and, where needed, authenticated denial of existence of domain name and resource records.

It is important to understand that DNSSEC does not solve every security problem related to the DNS. DNSSEC does not provide confidentiality of DNS responses or communications between DNS clients and servers. It also does not prevent attacks on DNS servers using other parts of the network stack—for instance, implementation of DNSSEC does not protect against Distributed Denial of Service (DDoS) attacks. As any organization may be targeted by malicious actors using a DDoS attack, it is advisable that their security plan, and that of their DNS provider, include a DDoS mitigation plan. Despite some of the drawbacks, DNSSEC ensures users are not hijacked en route to their intended destination and thereby builds trust and confidence to users conducting online activities.

What is the fTLD Domain Security Requirement for DNSSEC?

Requirement #23:

> **DNSSEC must be deployed at each zone and subsequent sub-zones for domains that resolve in the DNS.**

Registrars must communicate the DNSSEC requirement to their Registrants in their Registration Agreements and store Registrant's DNSSEC records. Registrars must support DNSSEC and registrants must deploy DNSSEC for each domain and subdomain name that resolves in the DNS.

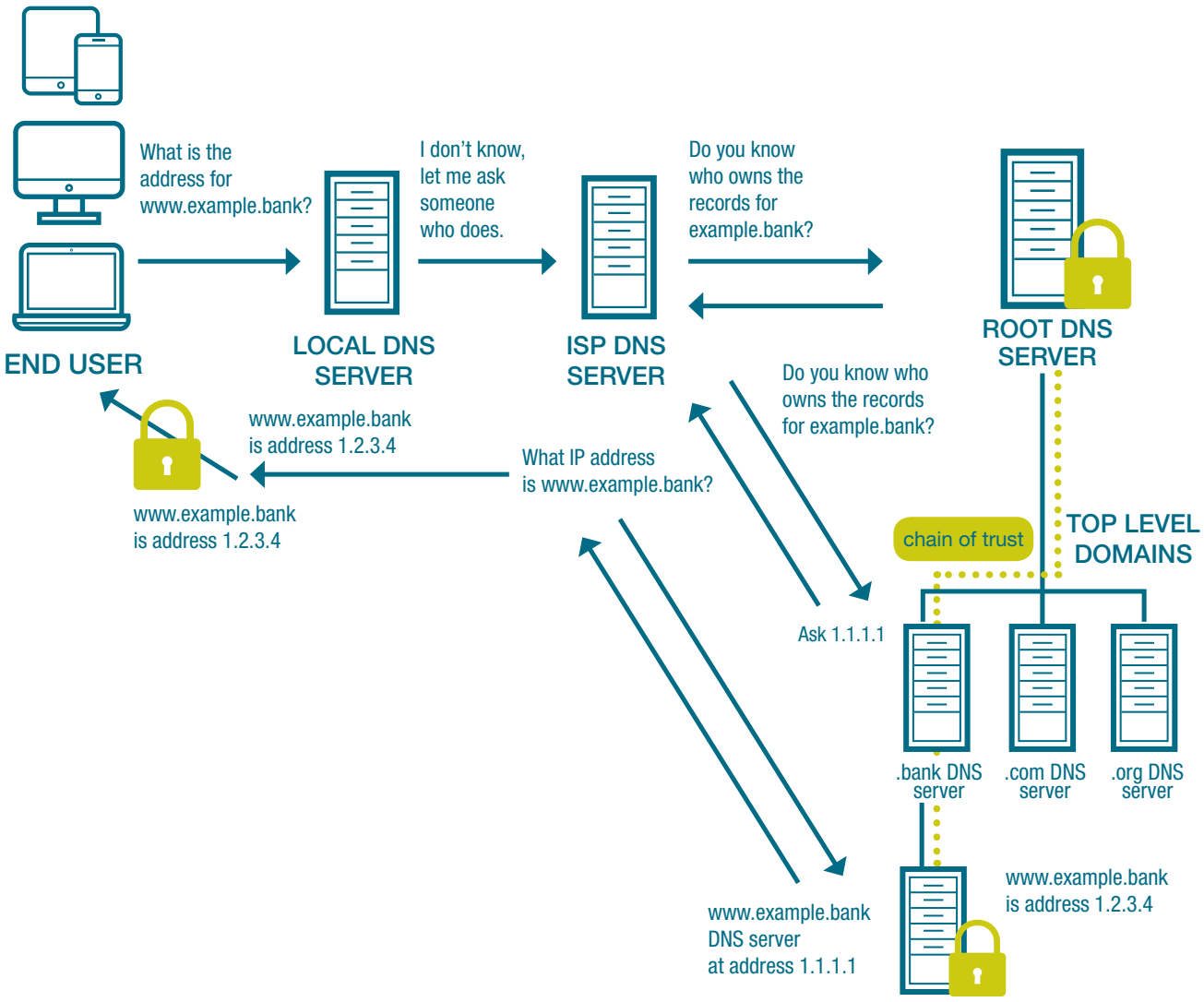
How Does DNSSEC Work?

DNSSEC uses public key cryptography to authenticate domain names and verify the integrity of traffic to and from those names. The internet relies on the ability of people and computers interacting at a distance. A customer may not have ever seen or met the principals of an internet business, but they still have to be able to exchange information with them. How does that customer know that the information sent from the business is legitimate and not an imposter?

In the real world there are many physical clues and cues that guide the application of trust to transactions, however most of these are simply not available on the internet.

One of the keys to building private and reliable communications between people, businesses and organizations on the internet is to use cryptography. Many types of cryptography are in use in the public internet, but DNSSEC uses a very specific type: public key cryptography.

DNSSEC allows a user to make a traditional query about an address and by using a combination of typical DNS queries and the addition of verifiable trust, ensure that the server answering the query is authoritative and the response to the query has not been tampered with during transmission. This is illustrated below for `www.example.bank`.



Public Key Cryptography

Public key cryptography is a secure, easy-to-deploy method for ensuring that messages and transactions can be changed into a form that can only be read by the intended recipient.

Each person, business or organization gets two "keys." One, the "public key" is given to all possible recipients and made public. The other, the "private key," is kept secret and in the hands of the sender. Messages encrypted with the "public" key can only be decrypted with the matching private key and messages encrypted with the "private" key can only be decrypted with the "public" key. When sending a message it is encrypted using the public key of the recipient. Given the only person who can decrypt the message is the person who has the matching private key, it doesn't matter if another party intercepts the message. Even if someone could intercept or get a copy of the encrypted message, they could not decode as they would not have the matching private key.

Public key cryptography has wide use in technologies such as Secure Sockets Layer (SSL), Transport Layer Security (TLS), Pretty Good Privacy (PGP) and Virtual Private Networks (VPNs) as a strategy for protecting messages on the internet.

Digital Signatures

One of the features of public key cryptography that is of special importance is digital signatures. Simply put, if a private key is used to generate a signature of a document or encrypt a message, the recipient can verify that it is legitimate by verifying the signature against the public key of the sender. As with protecting the confidentiality of messages, the digital signature relies on the essential property of the key pair; messages signed with the private key can only be verified using the matching public key and vice versa.

The signature that is created using the digital signing algorithm verifies the authenticity of that message. The digital signature guarantees that the sender of the message is who they say they are. Digital signatures have three important properties:

- **AUTHENTICATION**—Digital signatures are regularly used to authenticate the true source of messages. When a private key is used to sign a message, a valid signature indicates that the message was sent by the holder of the private key, and this ensures sender authenticity.

- **INTEGRITY**—In addition to being confident that the message came from who purported to send it, it is also important to ensure that the message has not been altered in any way during transmission. This is often referred to as "data integrity." A digital signature can ensure that the message was received just as it was sent: if a message is digitally signed, any change in the message after it was signed would invalidate the signature.

- **NON-REPUDIATION**—If a sender has signed some information, they cannot, at a later time, deny having signed it. Similarly, that someone has access to a public key of another does enable them to use that public key to forge a valid signature.

DNSSEC makes essential use of the first two of these properties of digital signatures. The fundamental feature that DNSSEC provides is to guarantee that the answer received as a result of a DNS query is exactly the answer that corresponds to the authoritative state of the DNS records related to that query.

DNSSEC Resource Records

DNSSEC is an extension to the DNS and it adds records to the DNS that can be used by DNS clients to validate the authenticity and integrity of an answer to a DNS query. Where a DNS server must indicate that it has no records with which to respond to a query, DNSSEC provides a way for this "negative answer" to be authenticated as well. To accomplish this DNSSEC adds a set of new DNS resource records:

- DNSKEY
- RRSIG
- NSEC
- DS

DNSKEY

Every zone file that is secured by DNSSEC has a public and private key pair. The administrator of the zone is responsible for keeping the private key secret. However, in order to check the signature on the zone data, DNS clients must be able to get a copy of the matching public key. As a result, the public key is published in the DNS using DNSSEC's DNSKEY resource record. Here is an example of a DNSKEY record:

```
example.bank. 86400 IN DNSKEY 256 3 8 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
Cbl+BBZH4b/0PY1kxkmvHjcZc8no
kfzj31GajIQKY+5CptLr3buXA10h
WqTkF7H6RfoRqXQeogmMHfpftf6z
Mv1LyBUgia7za6ZEzOJBOztyvhjL
742iU/TpPSEDhm2SNKLijfUppn1U
aNvv4w== )
```

The Time-to-Live (TTL) value is one day (86,400 seconds). TTL is a value that controls how long any name server may cache the information that it may acquire in a query. The Flags value is 256, indicating that this is a Zone Key. The protocol value is a constant number: 3. The field that follows is the identifier for the public key algorithm, and the value 8 indicates RSA/SHA-256. The RR value is simply the encoding of the public key that the DNS client will use to validate the signed zone.

RRSIG

A Resource Record Set is a collection of resource records in a DNS Zone file that have a common name, class and type. In practice this means there would be a resource record set associated with each part of a DNS query response. Another way to say this is that the RRSIG is the Zone Administrator's private key applied to the part of the zone requested by the client: a digital signature for the Resource Record Set.

In a typical zone with SOA (Start of Authority), NS (Name Server), A, MX (Mail Exchanger) and DNSKEY (Domain Name System KEY) resource records there would be five separate Resource Record Sets, each with a separate RRSIG record. Here is an example of a RRSIG record:

```
host.example.bank. 86400 IN RRSIG A 8 3 86400 20030322173103
(20030220173103 2642 example.bank.
oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t
GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrL
J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

The zone name is followed by the TTL value and the Class field. RRSIG indicates the resource record type. These three fields are followed by a "Type Covered" field which indicates that this is a signing of the A resource records in "`host.example.bank`." The next field indicates the name of the algorithm used to sign the resource record set. The number "3" indicates that there are three labels in the original zone name (in this case, "host," "example," and "bank."). The next field (86400) indicates the original TTL value for the covered A resource record set. Following that are date stamps indicating when the resource record set was signed and when the signature expires. The key tag is 2642 and the signer's name is "`example.bank.`" The rest of the RR value is the digital signature of the resource record set.

NSEC

The DNSKEY and the RRSIG can be used to check the authenticity and the integrity of a DNS response, but only when there is a response. There needs to be a way to check the authenticity and integrity of a response when the DNS server is unable to find a result for the DNS query. The DNS doesn't have a built-in resource record for this, so one was invented called the NSEC record.

The NSEC record was created so that something is returned in the event the resource requested does not exist. When a client makes a DNS query and either the name does not exist, or if the resource record type requested does not exist, the NSEC record is returned as a negative answer: a digitally signed indication that the name or resource record was not found.

Effectively the NSEC record is a way for the DNS server to bridge the gap between names in a secured zone. The strings in a zone file are sorted and then NSEC records are put in place to cover the gaps between the strings in the zone. If the zone contained the names "here" and "there" then there would be a NSEC record for "here" and its value would be "there" indicating that there are no defined names in the list of possible names between "here" and "there" Here is an example of a NSEC record:

```
alfa.example.bank. 86400 IN NSEC host.example.bank.
( A MX RRSIG NSEC TYPE1234 )
```

As before, the first four fields specify the name, TTL, Class and resource record type (this time, NSEC). This record then indicates that the next name, in sorted order, after "`alfa.example.bank`" in the zone is "`host.example.bank.`" The codes "A" "MX" "RRSIG" "NSEC" and "TYPE1234" indicate that there are "A" "MX" "RRSIG" "NSEC" and "TYPE1234" resource records associated with the name `alfa.example.bank.` One implication of this is that, by collecting the information from the NSEC records, the entire contents of the DNS zone can be enumerated.

DS

To prove a zone file's DNSKEY is truly authentic, the client first verifies that the parent's copy is authentic. It does this by calculating the signature of the key inside the DS record using the parent's public key. If it matches the RRSIG associated with the DS record, then the parent copy is authentic. Next, the parent copy is compared against the child's copy—if they are the same, the parent has authenticated the child's DNSKEY.

While there may be times when the parent zone's public key is suspected of being compromised, the chain of trust means that the parent zone's public key can be checked against the copy of the parent's zone held by the parent of the parent. The same process can be used until the DNSSEC client encounters a "trusted" DNSKEY. The ideal "trusted" DNSKEY is that of the root zone—and the good news is that the root zone has been signed since June 2010.

Here is an example of a DS record for the zone `example.bank`:

```
dskey.example.bank. 86400 IN DS 60485 8 1 ( 2BB183AF5F22588179A53B0A
98631FAD1A292118 )
```

As always, the first four fields are the name, TTL, Class, and resource record type (this time, the DS record). 60485 is the key tag for the corresponding "`dnskey.example.bank`" DNSKEY resource record. The number "8" indicates the algorithm used by "`dnskey.example.bank`" to construct the DNSKEY resource record. The value "1" indicates the algorithm used to construct the digest and the remainder of the record is the digest of the DNSKEY in hexadecimal format.

In essence, DNSSEC simply adds additional data to the responses that allow the DNS client to authenticate the resource records returned by the server.

If the client requests the security information:

- If the string and resource record type being queried exist, the authoritative name server simply adds the RRSIG to the response;
- If the string or the resource record being queried is not in the zone, the authoritative name server instead returns the NSEC record and its accompanying RRSIG record to indicate the negative response.

The DNS client can always take the returned RRSIG data and calculate its own copy of the signature to see if they match. The DNSKEY is needed to do this and if the client has not validated the DNSKEY within some set time period, the client needs to also validate the DNSKEY using the DS record stored at the parent zone. Each parent zone public key must also be validated in turn. Once the chain of trust is built and the signatures match, the DNS response can be considered to be validated and authentic.

DNSSEC and an fTLD Domain

To meet requirement #23 the zone, its parent zone, that zone's parent zone, the fTLD Domain and the root of the DNS all must be signed. To tell if a zone is signed, all you need to do is see if it has a DNSKEY record. If it does, the good news is that it supports DNSSEC.

Fortunately, the root and fTLD Domain are already signed and have DNSKEY records. Suppose we want to sign `example.bank`—what would we need to do next?

Every zone gets two pairs of keys: the Zone Signing Key (ZSK) and the Key Signing Key (ZSK). It is common to generate those keys first. Then, the essential steps are:

- Sign the zone with your ZSK
- Sign the ZSK with your KSK
- Publish the fingerprint of the KSK in the DS record published in the parent zone (for example, when signing the zone for `example.bank`, the DS record would be published in the .bank zone)

Every time the zone changes (perhaps a new domain is added to the `example.bank` zone, or the key needs to be updated) the zone must be resigned and the publishing task must be completed again. The process used for replacing one key in a zone with another is called a key rollover.

Keys in the DNS have a limited lifetime for security reasons. However, for fTLD Domain customers it is important that the existing chain of trust is not broken when the key rollover takes place.

A key rollover takes place when the records generated with the newest private key are first introduced into the zone. The key pair is often generated well in advance and the public key may also have been made public well in advance. Both key-signing keys and zone-signing keys are routinely (usually on a scheduled basis) rolled over.

DNSSEC Deployment for Registrars and Registrants

The process of generating keys, signing zones, generating, uploading and publishing DS records and managing key rollover is complex.

Registrars must support DNSSEC and provide tools for registrants to help publish DS records. Registrants must also support DNSSEC and have the tools for generating keys, signing zones, generating, uploading and publishing DS records and managing key rollover.

In cases where the registrant outsources the management of their fTLD Domain zone, they must ensure that the third party is able to successfully generate keys, sign their zones, generate and upload the DS records and manage key rollovers. For organizations that choose to outsource the DNSSEC management activity, there are a variety of vendors with service offerings that will comply with the fTLD Domain DNSSEC requirement. Webhosting companies, registrars and DNS management services may have service offerings that can assist a registrant in complying with the DNSSEC requirement.

In cases where a registrant has decided to implement DNSSEC on their own servers, it is useful to have tools that will help key management, signing, publishing and rollover tasks. In addition to commercial tools that are easily found using simple searches on the internet, there are two primary open source approaches available:

- OpenDNSSEC project at `http://www.opendnssec.org`
- DNSSEC-Tools project at `http://www.dnssec-tools.org`

TLS/ENCRYPTION

What is TLS?

TLS stands for Transport Layer Security. TLS is an open standard for encryption that makes it possible for people and applications to communicate in private over the internet. When a client and server communicate, TLS is used to ensure the messages passed between them cannot be read or changed during the transmission.

TLS is the successor to a technology called Secure Sockets Layer (SSL) and it provides three essential services to help ensure security on the internet:

- Message Confidentiality—TLS provides a mechanism for encrypting the messages between application clients and servers with the aim of ensuring that, even if someone captures the messages as they were in transit between the sender and destination, the messages could not be read.
- Authentication—TLS provides a mechanism to validate the identity of the endpoints of the communications. Servers authenticate themselves to clients and clients, optionally, authenticate themselves to servers.
- Message Integrity—TLS provides a mechanism to ensure that messages are not changed as they move between client and server.

TLS is often associated with protecting the contents of messages, but the protocol is also very useful in protecting against threats such as masquerade attacks, man-in-the-middle or bucket brigade attacks, rollback attacks and replay attacks.

What are the fTLD Domain Security Requirements for TLS?

There are three requirements that reference TLS/encryption:

Requirement #24:

> **Registrar and Registrant access to registration systems must be mutually authenticated via Transport Layer Security and secured with multi-factor authentication, NIST Level 3 or better.**

Requirement #25:

> **Registration Authorities and Registrants are required to use encryption practices defined by NIST Special Publication 800-57, or its successor, for all electronic communication between parties, including but not limited to web access, mail exchange, and file transfer, avoiding the use of unencrypted protocols to prevent tampering with messages.**

Requirement #29:

> **Transport Layer Security (the successor to SSL) must be implemented as detailed in the Notes column of the requirement.**

Requirement #24 ensures that message confidentiality and integrity is a part of the authentication process for access to registration systems. Requirement #25 ensures the benefits of TLS are present in all interactions between registries-registrars, registrars-registrants, and registrants and their customers including email and Web interactions. Requirement #29 specifies the TLS protocol version and related cipher prohibitions.

An fTLD Domain is an HTTPS-only community to support confidentiality and integrity of web and other services by default. Public key certificates must be deployed to meet the TLS/encryption requirement.

To ensure a positive, uninterrupted user experience of an fTLD Domain, an unencrypted fTLD Domain may exist for the sole purpose of redirecting to an encrypted fTLD Domain. For example, `http://companyname.insurance` responds with the minimal web code required to redirect to `https://companyname.insurance`. A common method to redirect is to use the HTTP '301 Moved Permanently' response status code.

How Does TLS Work?

TLS is built from a pair of protocols: the "record" protocol and the "handshake" protocol. Fundamentally, TLS exchanges records—the content and formatting of those records being agreed at the start through a "negotiation" between client and server. The content of those records may be control messages for TLS, the actual encrypted content flowing between both parties and a variety of other information.

The TLS Record Protocol provides connection security that has two basic properties:

- **The connection is private**. Every message between two endpoints is secured by encryption. A shared secret is used for the message encryption. The shared secret is generated uniquely for each connection between the two endpoints and is based on the results of another protocol (often the TLS Handshake Protocol).

- **The connection is reliable**. Every message has a checksum applied so that the receiving end can always check to see if the message that was received is exactly the same as the message that was sent.

The shared secret is important because symmetric cryptography is significantly faster than public key cryptography. However, the shared secret is a problem because both endpoints need to know the same shared secret to exchange messages; this is where the TLS Handshake Protocol plays a role.

The TLS Handshake Protocol provides connection security between two endpoints on the internet, and has the following three properties:

- **Each end of the communications channel can be authenticated** (that is, they can be forced to prove that they are who they say they are). This authentication takes place using public key cryptography. In traditional web applications only the server is authenticated.

- **The negotiation of a shared secret is secure**. The secret that is generated and shared (to be used by the Record Protocol) is impossible for an eavesdropper to obtain. The shared secret cannot be discovered, even by attackers who place themselves in the middle of the session.

- **The negotiation is reliable**. If an attacker attempts to insert themselves into the negotiation process, they can be discovered by the peers during the negotiation.

The combination of the negotiated secret provided by the TLS Handshake Protocol and the symmetric cryptography and checksums provide a secure communications channel for any application layer protocol on the internet (e.g., World Wide Web, email, VoIP).

The handshake is essential to TLS, but also fairly complex. The following overview describes the handshake when both ends of the connection must authenticate themselves to one another.

1. The client begins the handshake by sending an initial message to the server indicating the highest version of the TLS protocol that it supports, a list of cryptographic suites it supports, a list of message compression methods it supports and a random number.

2. The server responds:

    a. First, with a message with the highest version of the TLS protocol that the two sides have in common, the best of the cipher suites that the two have in common, the most efficient compression methodology that both support and a random number.

    b. Second, it sends the server's Digital Certificate.

    c. Third, it sends the public key for the server.

    d. Fourth, it requests that the client send a copy of the client's Digital Certificate.

    e. Fifth, it sends a short message to indicate that it is done with this part of the negotiation.

3. The client then responds:

    a. First, with a copy of its Digital Certificate.

    b. Second with a message containing the public key of the client.

    c. Third, the client uses its private key to sign the previous handshake messages.

    This signature can be verified as authentic by checking it using the client's certificate's public key. This proves that the client has access to the private key of the certificate and thus owns the certificate.

4. The server and client then send messages using the random numbers that have been exchanged to generate a common secret. For the remainder of this connection, all cryptographic material is generated from this common secret.

5. Once the common secret has been generated the client sends a message to the server telling the server that all future communications sent to the server will be authenticated and encrypted.

6. The client sends a last message indicating that its part of the handshake is complete.

7. The server decrypts the last message of the client (if the decryption fails, the handshake is considered to have failed).

8. The server sends a message to the client telling the client that all future communications sent to the client will be authenticated and encrypted.

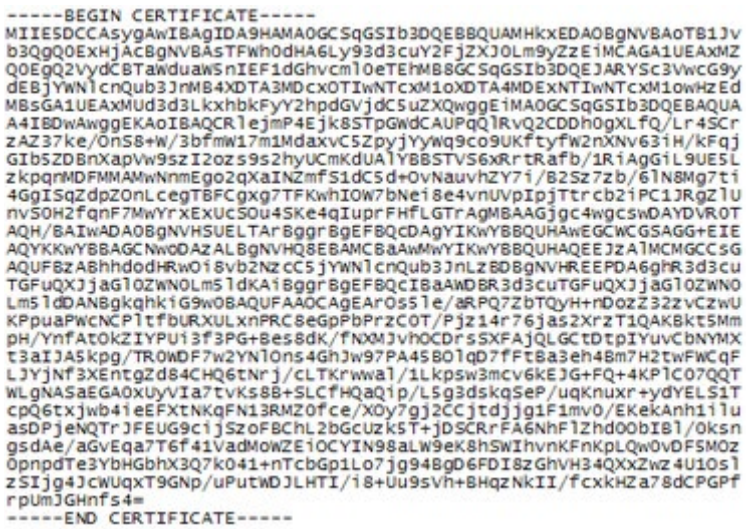
9. The server sends a last message indicating that its part of the handshake is complete.

10. The client decrypts and verifies the last message from the server.

11. The handshake is successful and the application continues with all messages encrypted in exactly the same way as their “last messages” in the handshake.

TLS is a powerful tool for ensuring secure channels of communications. However, there are known attacks on the protocol and every security professional working with an fTLD Domain should be aware of the known vulnerabilities of TLS. Details are beyond the scope of this document, but a summary is available here: `https://tools.ietf.org/html/rfc7457`.

Crucial to the handshake is the availability of a digital certificate. Digital certificates are files with a specific, standardized format which attests to the identity of an entity. That entity could be nearly anything such as a person, a named user, a server, a smartphone or a program on a client. Inside the certificate, along with the attestation of identity, is the public key. Here’s what a sample certificate looks like:



```
-----BEGIN CERTIFICATE-----
MIIESDCCAsygAwIBAgIDA9HAMA0GCSqGSIb3DQEBBQUAMHkxEDAOBgNVBAoTB1Jv
b3QgQ0ExHjAcBgNVBAsTFWh0dHA6Ly93d3cuY2FjZXJ0Lm9yZzEiMCAGA1UEAxMZ
Q0EgQ2VydCBTaWduaW5nIEF1dGhvcml0eTEhMB8GCSqGSIb3DQEJARYSc3VwcG9y
dEBjYWNlcnQub3JnMB4XDTA3MDcxOTIwNTcxM1oXDTA4MDExNTIwNTcxM1owHzEd
MBsGA1UEAxMUd3d3Lkxhbkfyy2hpdGVjdC5uZXQwggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCRlejmP4Ejk8STpGWdCAUPqQlRvQ2CDDhOgXLfQ/Lr4SCr
zAZ37ke/OnS8+W/3bfmW17m1MdaxvC5ZpyjYyWq9co9UKftyfW2nXNv63iH/kFqj
GIb5ZDBnXapVw9szI2ozs9s2hyUCmKdUAlYBBSTVS6xRrtRafb/1RiAgGiL9UE5L
zkpqrMDFMMAMwNnmEgo2qXaINZmfS1dC5d+OvNauvhZY7i/B2Sz7zb/6lN8Mg7ti
4GgISqZdpZOnLcegTBFCgxg7TFKwhIOW7bNei8e4vnUVpIpjTtrcb2iPC1JRgZlU
nvSOH2fqnF7MwYrxEXUcSOu4SKe4qIuprFHfLGTrAgMBAAGjgc4wgcswDAYDVR0T
AQH/BAIwADA0BgNVHSUELTArBggrBgEFBQcDAgYIKwYBBQUHAwEGCWCGSAGG+EIE
AQYKKwYBBAGCNwoDAzALBgNVHQ8EBAMC8aAwMwYIKwYBBQUHAQEEJzAlMCMGCCsG
AQUFBzABhhdodHRwOi8vb2NzcC5jYWNlcnQub3JnLzBDBgNVHREEPDA6ghR3d3cu
TGFuQXJjaGlOZWNOLm5ldKAiBggrBgEFBQcIBaAWDBR3d3cuTGFuQXJjaGlOZWNO
Lm5ldDANBgkqhkiG9w0BAQUFAAOCAgEArOss1e/aRPQ7ZbTQyH+nDozZ32zvCzwU
KPpuaPWcNCPltfbURXULxnPRC8eGpPbPrzCOT/Pjz14r76jas2XrzT1QAKBkt5Mm
pH/YnfAtOkZIYPUi3f3PG+8es8dK/fNXMJvhOCDrsSXFAjQLGCtDtpIYuvCbNYMX
t3aIJA5kpg/TROWDF7w2YNlOns4GhJw97PA45BOlqD7fFtBa3eh4Bm7H2twFWCqF
LJYjNf3XEntgZd84CHQ6tNrj/cLTKrwwal/1Lkpsw3mcv6kEJG+FQ+4KPlCO7QQT
WLgNASaEGAOxUyVIa7tvKs8B+SLCfHQaQip/L5g3dskqSeP/uqKnuxr+ydYELS1T
cpQ6txjwb4ieEFXtNKqFN13RMZOfce/xOy7gj2CCjtdjjg1F1mv0/EKekAnh1ilu
asDPjeNQTrJFEUG9cijSzoFBChL2bGcUzk5T+jDSCRrFA6NhFlZhdOObI8l/Oksn
gsdAe/aGvEqa7T6f41VadMoWZEiOCYIN98aLW9eK8hSWIhvnKFnKpLQw0vDF5MOz
OpnpdTe3YbHGbhX3Q7kO41+nTcbGp1Lo7jg94BgD6FDI8zGhVH34QXxZwz4U1Osl
zSIjg4JcWUqxT9GNp/uPutWDJLHTI/i8+Uu9sVh+BHqzNkII/fcxkHZa78dCPGPf
rpUmJGHnfs4=
-----END CERTIFICATE-----
```

When using TLS, each side of the connection can use the digital certificate to try to validate its contents and the claim of identity for the entity. One of the fields that is often used for this is called the “Common Name.” Each side of the conversation can check the Common Name against what they expect. They can also check the expiration date of the certificate. Especially useful in these circumstances, both sides can check who issued the certificate.

Digital certificates are issued by Certificate Authorities (CA). There are companies that offer a service to do validation of the identity information in the digital certificate and then issue certificates that they can confirm. These companies are often called Trusted Certificate Authorities. The real value of certificates appears when they come from a trusted source. Individual users can generate their own certificates, called self-signed certificates, but these have limited value.

TLS/Encryption for an fTLD Domain

The primary use for TLS in an fTLD Domain environment is in securing activity on the internet. Requirement #24 specifies that registrar and registrant access to registration systems must be mutually authenticated via TLS and secured with multi-factor authentication, NIST Level 3 or better.

When the registration system is an application running in cooperation with a web server, this implies that the clients and servers must have digital certificates, must support at least the TLS specified in the Requirements, and must be able to fully negotiate the handshake that sets up the secured session.

Requirement #25 specifies that all electronic communication between parties, including, but not limited to, web access, mail exchange, and file transfer, use encrypted protocols to prevent tampering with messages.

Requirement #29 specifies that TLS must be implemented using trusted protocol versions because some early implementations of TLS and some cipher suites supported by the protocol are known to be insecure. A public key certificate must be used to meet this requirement.

fTLD Domains must support the TLS specified in the Requirements. Changes to TLS protocols are ongoing and registrants should confirm the current requirement. Registrants and registrars will be notified of changes to any Requirements on a timely basis.

In addition, a series of cipher suites which are currently known to be insecure are not permitted for use with an fTLD Domain and are identified in requirement #29.

The steps required to implement TLS on a web server is discussed in the next section of this document. If the services are built to be used via a web browser, access through the https:// schema makes TLS implementation straightforward. Services built to be used through other tools, for instance through an iPhone, Android or Windows Phone app, need to use the secure transport APIs for the particular platform being targeted.

All component parts of web pages on websites from an fTLD Domain must be sent using TLS. It is possible to build “mixed-content” web pages, which are sent from server to client over TLS, but include resources (e.g., images, CSS or JavaScript files) that are not protected by TLS. However, such pages are not secure and are prohibited because of requirement #29. An active man-in-the-middle (MITM) attacker can piggyback on a single unprotected CSS file, as an example, and hijack the entire user session.

However, this does not prohibit an fTLD Domain from using content from other services. As an example, if an fTLD Domain incorporates information from a social media feed, the content from the social media may not originate or be hosted from an fTLD Domain service. It is acceptable to incorporate content from other sources in an fTLD Domain.

In addition to implementing TLS, registrars and registrants may choose to implement HTTP Strict Transport Security (HSTS). HSTS is effectively a backup for TLS. It was engineered to ensure that security remains intact even in the case of configuration problems and implementation errors. HSTS protection is simple to implement: you set a single response header in your websites. After that, browsers that support HSTS[3] will enforce it. While the use of HSTS is not currently a requirement, it is strongly encouraged by fTLD and may become a requirement in the future.

[3] See owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet#Browser_Support

Once implemented, HSTS does not allow any insecure communication with the website that uses it. It achieves this goal by automatically converting all plaintext links to encrypted ones. It also makes it harder for a user to click-through certificate errors and warnings which may be a leading indicator of a more significant problem such as malware or a man-in-the-middle attack (Certificate errors are a potential indicator of an active MITM attack. Studies have shown that almost all users click through these warnings without heeding them.).

TLS/Encryption Deployment for Registrars and Registrants

For an fTLD Domain, implementing TLS on web servers requires six primary steps:

1. **Generating a server certificate request file.** A certificate request is essentially certificate data that has not been signed by a Certificate Authority (CA). The CA turns the request into a certificate by signing it. Building the certificate request file is a task that is specific to each operating system and web server. The end of the process is simply a file that will be transmitted to a Certificate Authority.

2. **Generating a server certificate.** To submit your TLS server certificate request file to a Certificate Authority you must follow the enrollment instructions the CA provides. Most CAs publish their enrollment instructions on their website. Enrollment instructions vary among commercial CAs and for different server certificate types. The end of the process is a digital certificate signed by the CA and returned to your organization.

3. **Installing a server certificate on the web server.** Most web servers and operating systems have Digital Certificate management and administration tools that allow you to install the new Digital Certificate on your server. Once the Digital Certificate has been installed the web server must be configured to use it as part of the TLS Handshake.

4. **Configuring TLS on a web server.** Each web server hosting registration system will have to be configured to support the current version of the requirement. Details of configuration are specific to each web server and operating system, but often it consists of adding support via an included module or selection of a configuration option. Some examples of this are Microsoft IIS[4] and Apache[5]. fTLD Domain servers should also consider configuration of HSTS as a default response header.

5. **Generating, acquiring, and installing client certificates.** Requirement #24 of the Requirements means mutual authentication via TLS is required. Client certificates are required for those computers on which browsers are going to use the registration system. The registrar must deploy a system for generation, acquiring and installing client certificates.

6. **Ensuring that TLS clients trust the CA certificate.** This generally means making sure that the CA's certificate is stored in the client's trusted root certificate store.

In step 2, above, the Certificate Authority generates the Certificate used for TLS. The certificate is tied to a specific FQDN. fTLD suggests that registrars and registrants ensure that their certificates cover all the names to be used with a site. If the main domain name is going to be `www.myexample.bank`, you may also wish to have certificates configured for `myexample.bank` and other names which users might

[4] See Microsoft IIS at `microsoft.com/web/platform/server.aspx`

[5] See `http://httpd.apache.org`

use (or, which your financial institution uses for brand recognition purposes); a wildcard certificate (e.g., *.myexample.bank). is one solution to cover every DNS name. A secure fTLD Domain web server should have a certificate that is valid for every DNS name configured to point to it. The goal is to avoid invalid certificate warnings at the browser, which will potentially confuse users and may weaken their trust.

Once installed and configured, the server should be able to respond properly to client requests for TLS sessions. It is possible to test the configuration of the server and determine if the combination of the certificate installation and the server configuration has been successful. Several organizations provide free services to make this check such as:

DigiCert® Certificate Inspector – https://www.digicert.com/cert-inspector.htm

CheckTLS.com – https://www.checktls.com

Wormly – https://www.wormly.com/tools

Symantec CryptoReport and Toolbox – https://ssltools.websecurity.symantec.com/checker

Encryption of Web, Mail and Other Service Ports

As an example, in the case of electronic mail, one approach for meeting this requirement is to implement STARTTLS. STARTTLS is an open protocol that improves upon plaintext protocols such as email and file transfer. Essentially, STARTTLS provides a way to upgrade older plaintext protocols to an encrypted, TLS-based connection. The advantage of STARTTLS is that it is an open standard and extensions are provided for email (in RFC 2595 and RFC 3207), instant messaging and presence (in RFC 6120), and directory services (in RFC 2830).

Previous to STARTTLS, the internet provided TLS (and SSL) ports for well-known services. This is very much an analog to HTTP and HTTPS. For instance,

- SMTP had a variant on port 465 for SMTPS
- POP3 had a variant on port 995 for POP3S
- IMAP had a variant on port 993 for IMAPS
- NNTP had a variant on port 563 for NNTPS

Today most mainstream providers of services—especially email and file transfer—employ STARTTLS as the alternative to these legacy approaches. For instance, Google has a real-time report on email encryption available at:

google.com/transparencyreport/saferemail/

Commercial software supporting these internet services widely supports STARTTLS. As an example, Microsoft Exchange uses a feature called Opportunistic TLS that enables TLS by default for the server. This enables any sending system to encrypt the inbound SMTP session to Exchange. By default, Exchange 2013 also attempts TLS for all remote connections.

Traffic should be encrypted as the default option. Services in an fTLD Domain should always attempt to have transmission encrypted. The only case where defaulting back to plaintext for transmission is in extraordinary cases: for instance, sending emails to legacy domains. STARTTLS should always be attempted, but some connections will not support TLS. However, since all modern browsers support strong encryption, there should not be a fallback for HTTP connections.

EMAIL AUTHENTICATION

What are DMARC, SPF and DKIM?

Much of the interpersonal communication on the internet still depends on one of the oldest internet services: electronic mail. Social networking and other mobile services have not changed this. More than ever, those wishing to do harm on the internet have a tremendous financial incentive to compromise user accounts, enabling theft of passwords, bank accounts, credit cards and other security violations. The original electronic mail protocols never envisioned the risks to consumers in simple email spoofing. In some cases, the simple insertion of a logo of a famous brand is enough to give email legitimacy with many users.

Even sophisticated users can be fooled by fake messages. And large providers of mail services have to make very difficult choices about which messages to deliver and which ones they should not. DMARC is a technical specification, using the DNS and mail servers, to attempt to deal with some of these issues. The goal is to help email senders and receivers work to reduce the number of phishing attacks and other malicious email-borne activities.

There are three standards that make up the email authentication technology:

- **DMARC – Domain-based Message Authentication, Reporting and Conformance** is a technology that uses the resources of DNS and email servers to help avoid email abuse—specifically, phishing. It is layered over two technologies (SPF and DKIM) that allow for the specification of policies for incoming email. Publishing a DMARC record is one of the Requirements;

- **SPF – Sender Policy Framework** is a technology that allows an administrator to publish information about legitimate sending hosts in a specially formatted DNS resource record; and

- **DKIM – DomainKeys Identified Mail** is a technology that allows a mail receiver to check that incoming mail from a domain is authorized by that domain's administrators and that the email has not been changed as it has gone through the network.

fTLD recommends that both SPF and DKIM be used with DMARC. However, the requirement allows an administrator to use only one. In practice, email authentication is more effective in the presence of both SPF and DKIM records as email delivery rates will be increased.

What is the fTLD Domain Security Requirement for Email Authentication?

Requirement #26:

> **Registrants must publish a valid Domain-based Message Authentication, Reporting and Conformance (DMARC) record with a requested mail receiver policy of either quarantine or reject for domains that resolve in the DNS.**

For domains intended to send email, Registrants must publish at least one of the following email authentication DNS resource records:

- **Sender Policy Framework (SPF)**
- **DomainKeys Identified Mail (DKIM)**

When used to protect non-email sending domains, Registrants are required to publish a DMARC reject requested mail receiver policy.

The goal of this requirement is to provide an increased level of trust for email from an fTLD Domain. This is done by using an internet-standard approach to email authentication. This is a crucial requirement for ensuring that the electronic mail channel for an fTLD Domain is protected against the delivery of invalid or spoofed email purporting to be from an fTLD Domain.

What is Email Authentication and Why is it Important?

Email authentication allows the receiver of an email to make judgements about the sender of that email. In addition, the sender can develop and publish policy for email authentication. If a receiver can establish authentication of the source of email, then one of the most difficult problems for email can be avoided—spoofing also known as phishing. Spoofing is the ability of someone to pretend to be another organization and attempt to lure an email receiver into using a rogue website instead of an authentic website or into providing confidential information in order to compromise sensitive user information.

For organizations using an fTLD Domain, email authentication is an essential part of establishing trust in email coming from their organization. Email authentication assists in the prevention of phishing attacks and also plays a role in emerging reputation and accreditation systems that are another part of establishing trust in email as a vehicle for communications between financial services organizations and their customers.

There are two basic types of email authentication: internet Protocol-based approaches and cryptographic approaches. The internet Protocol approach, SPF, requires publishing information about legitimate email servers in the DNS for each domain being used. The other approach, cryptographic, attaches a signature to each message in a way that is difficult to forge, proving that the message came from the indicated sending domain. The industry standard for the cryptographic authentication is DKIM.

Today, companies are moving toward adoption of SPF, DKIM, or both. Major email providers such as Microsoft, Yahoo! and Gmail are using both SPF and DKIM to authenticate email senders. DMARC is a tool that uses SPF or DKIM to make decisions about incoming mail based on a published policy.

Understanding Email Authentication Protocols

The email authentication protocols are complex. While this section provides a basic overview and some examples, there are resources available to learn more about SPF, DKIM and DMARC.

This section provides links to some of the available sources of information on the three protocols.

- `http://www.openspf.org/SPF_Record_Syntax` – Provides a full SPF record syntax explanation;
- `http://tools.ietf.org/html/rfc7372` – The IETF RFC for SPF;
- `http://www.dmarc.org` – A group that provides tutorial information and support for DMARC;
- `https://datatracker.ietf.org/doc/rfc7489/` —The IETF RFC for DMARC;
- `http://www.dkim.org` – A group that provides tutorial information and support for DKIM;
- `https://datatracker.ietf.org/doc/rfc5863/` – The IETF RFC for DKIM; and,
- `http://www.bits.org/publications/security/BITSEmailAuthenticationFeb2013.pdf` – BITS paper on email authentication

Example: Building a SPF Policy

Registrants are reminded that the development of DMARC, SPF and DKIM has been very dynamic. The online resources in the previous section should be considered authoritative. However, in this section, fTLD provides an example of what is needed to build a SPF policy record to meet the requirement. Registrants in an fTLD Domain zone are required to have a SPF or a DKIM record. Similar to the DNSSEC resource records noted earlier in this guide, a SPF record is a resource record that identifies which mail servers are permitted to send email on behalf of your domain.

The idea behind a SPF record is to prevent spammers from sending messages with forged "From" addresses at your domain. This is especially important in the case of an fTLD Domain where the goal is to establish high levels of trust between institutions and their customers. Email service providers (e.g., Google, Microsoft, Yahoo!), on behalf of their customers, refer to the SPF record to determine whether a message purporting to be from your domain comes from an authorized mail server and then deliver it or not.

For example, suppose that your domain `example.bank` uses your internal mail server. You create an SPF record that identifies your internal mail server (suppose it is `mail.example.bank`) as the authorized mail servers for your domain. When a recipient's mail server receives a message from `customerservice@example.bank`, it can check the SPF record for `example.bank` to determine whether it is a valid message. If the message comes from a server other than your internal mail servers listed in the SPF record, the recipient's mail server can reject it as spam. It's perfectly possible to have an SPF record that authorizes more than one domain as the legitimate server for your outbound email.

If your domain does not have an SPF record, some recipient domains may reject messages from your users because they cannot validate that the messages come from an authorized mail server.

The diagram on page 27 shows how SPF is used to provide the "Envelope From" information as part

of the DMARC standard.

Suppose a mail is sent with the following address information:

```
Mail-from: customerservice@example.bank

From: customerservice@example.bank

To: steve@zelda.example.org
```

In SMTP, when one mail server connects to another the sending server introduces itself. In our example, let's say the sending server says HELO `mail.boston.example.bank`. Now that we know the hostname the sending server claims to be, we can check that. We do that by looking up the SPF record for `mail.boston.example.bank`, which might look like this:

```
mail.boston.example.bank.     TXT   "v=spf1 a -all"
```

This record tells us that the only host that can announce itself as `mail.boston.example.bank` is `mail.boston.example.bank` (indicated by the "a"). Note if there was no SPF record for `mail.boston.example.bank`, the result would be None, rather than Pass or Fail.

If the IP address of the sending server matches the IP address of `mail.boston.example.bank`, we have a Pass result for SPF.

If the IP address of the sending server does not match the IP address of `mail.boston.example.bank`, we proceed to the next part of the SPF record, -all, which yields a Fail result.

Since the mail-from shows an `@example.bank address` we look up the following SPF record:

```
example.bank. TXT "v=spf1 a:mail.boston.example.bank -all"
```

This record indicates that there is only one server that is allowed to send mail using the `example.bank` domain, and that is `mail.boston.example.bank`. Now that we know that, we look up the IP address of the `mail.boston.example.bank` host.

If the IP address we find for `mail.boston.example.bank` matches the IP address of the incoming connection, then we have a match and the SPF test yields a Pass result.

If the address does not match `mail.boston.example.bank`, then we go on to the next part of the SPF record, in this case "-all" which tells us that any other IP address yields a Fail result

Example: A DKIM Record

The requirement identifies that registrants may use DomainKeys Identified Mail (DKIM) as one of the two options for authenticating email messages in compliance with DMARC. DKIM is an email validation system that allows a mail server to check that a domain is authorized to send a particular piece of email and verify that the message has not be modified during its transport through the network. DKIM works differently from SPF in that DKIM uses a digital signature to sign the message. This signature is added to the content of the message and the recipient can then use it, along with a public key published in the DNS, to verify

that the message was sent by an authorized person and that the message has not been altered.

DKIM consists of a signing and verifying application usually built into a mail server.

DKIM requires that there be a public key available for the validation process. It works like this:

```
myselector._domainkey.example.bank TXT "k=rsa; p=AIGf ... AQAB"
```

Notice that there is a selector attached to the special domainkey subdomain. Here is a list of the possible options for this text record and their meanings:

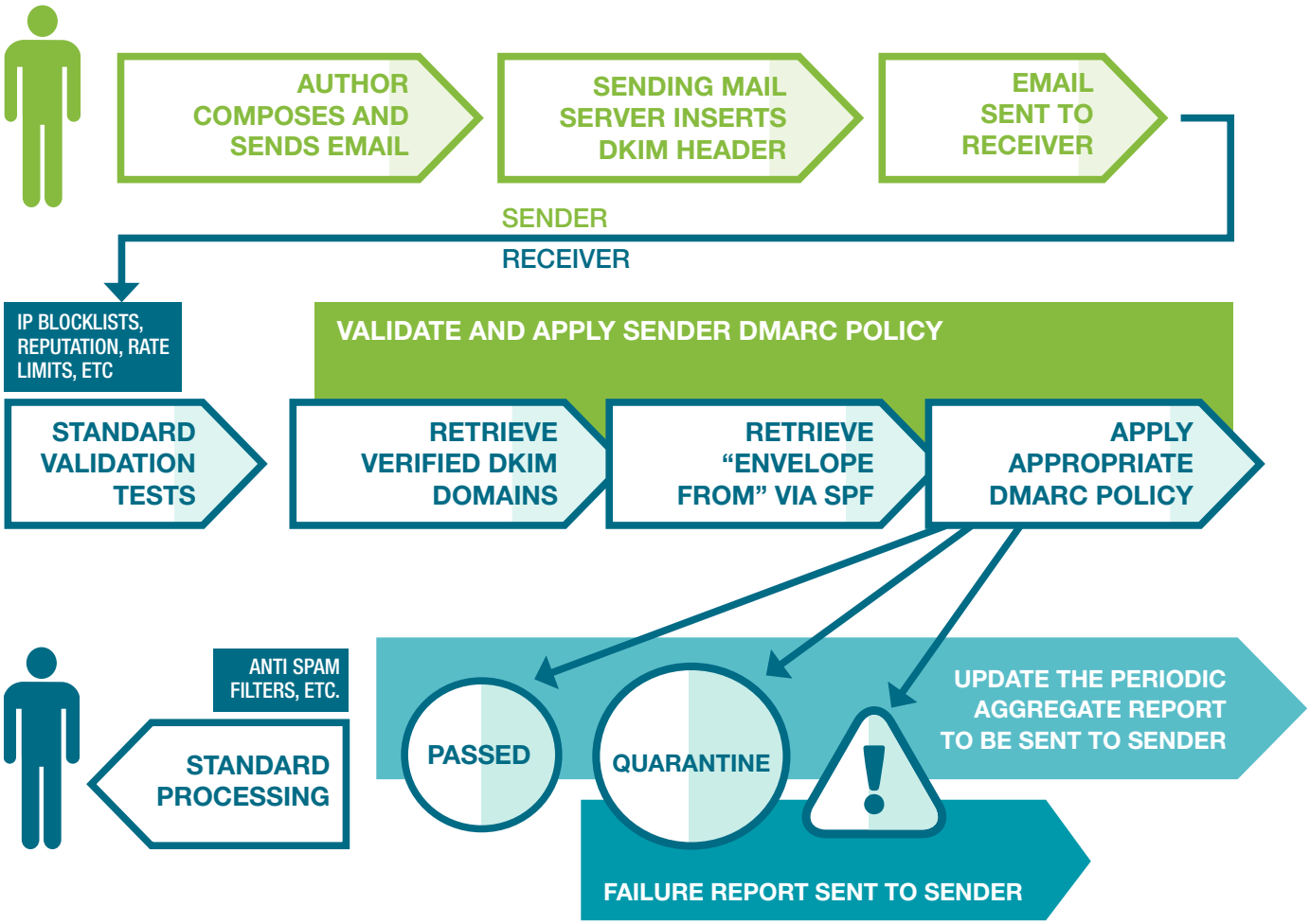
- k = key type (rsa is the default). All Signers and verifiers support the "rsa" key type.
- n = notes that may be of interest to a human. No interpretation is made by any program. This tag is optional.
- p = the public-key data, encoded as a Base64 string. An empty value means that this public-key has been revoked. This tag MUST be present.
- t = testing mode ("y" means that this domain is testing DomainKeys and unverified email MUST NOT be treated differently from verified email. Recipient systems may wish to track testing mode results to assist the sender.) This tag is optional.

With these two special TXT records in the DNS we can now see how DKIM works in practice. The sending server inserts a mail header in the message called the "DomainKey Signature." Effectively, this is a digital signature over the entire message contents using the private key of the server. A sample might look like this:

```
DKIM-Signature: a=rsa-sha1;
s=myselector;
d=example.bank;
i=customerservice@mail.example.bank;
c=simple; q=dns;
b=dydVyOfAKCdLXdJOc8G2q8LoXSlEniSbav+yuU4zGffruD00lszZVoG4ZHRNiYzR;
```

When this recipient gets the message, the server can look up the public key for `example.bank` and then calculate whether or not the message contents are the same as what was sent. It also allows the recipient to find out the signing organization's message signing policy.

We've seen how SPF and DKIM can add message authenticity to the sending process. The following diagram shows how DMARC fits into the process.



Example: DMARC in Practice

A DMARC policy uses the DNS to allow a sender to indicate that their emails are protected by SPF and/or DKIM. DMARC also suggests what the recipient should do if neither of those authentication methods passes—for instance, delete, none, quarantine or reject the message.

For an fTLD Domain, DMARC removes guesswork from the receiver's handling of these failed messages. It also provides a user the ability to eliminate fraudulent or harmful messages falsely purporting to come from an fTLD Domain. DMARC also provides an optional mechanism for receivers to report back to the sender about messages they receive with DMARC evaluations.

Here's an example DMARC record:

```
_dmarc.example.bank. IN TXT "v=DMARC1; p=reject; rua=mailto:postmaster@example.
```

```
bank, mailto:dmarc@example.bank"
```

Here is a list of the possible options for this text record and their meanings:

- v = the protocol Version for DMARC—this is a required parameter
- p = policy for the domain—this is another required parameter and the values can be none (which takes no action), quarantine (which marks messages as spam), and reject (which rejects the message entirely). The fTLD Domain guidelines suggest using p=none when implementing the Requirement for email.
- pct = the Percentage of the email message that is subjected to filtering
- rua = an email Address(es) for the Reports
- sp = a policy for the subdomains of the domain affected by this DMARC record
    - aspf = the Alignment Mode of SPF
    - adkim = the Alignment Mode for DKIM
    - ri = the Reporting Interval

The alignment modes for SPF enables an administrator to manage which parts of the message are authenticated by either SPF. Alignment, for DMARC DKIM, has to do with whether the signing domain used for the signature must strictly match the RFC5322 "From" domain, or if the signing domain and RFC5322 domains must merely be domains within the same organization domain, e.g., proper subdomains of the org domain, or one the organizational domain and the other a subdomain. Since the alignment can be different in diverse implementations, DMARC allows both "strict" and "relaxed" versions of the alignment with the goal of checking against the MailFrom portion of the incoming message.

The requirement suggests, but does not require, that the identifier alignment for both SPF and DKIM be set to "strict." Also, the optional pct tag is helpful during implementation. It can be used to stage and sample your DMARC deployment. Since 100 percent is the default, passing "pct=20" in your DMARC TXT record results in one-fifth of all messages affected by the policy actually receiving the disposition instead of all of them.

DMARC Deployment for Registrars and Registrants

DMARC deployment and mail authentication can be challenging. However, there is a large body of information available for implementers.

For those organizations running their own mail and DNS servers, there are both commercial and open source tools to add to existing mail servers. For organizations who want to offload the mail services to a separate appliance several message gateways have built-in DMARC support. Several tools have been released that support enterprise-class DMARC support in large scale email tools (e.g., Hexamail Guard for Microsoft Exchange).

For organizations that outsource their email services, there are a variety of third-party providers of hosted email that support SPF, DKIM and DMARC. In addition, it is possible to have a third-party sender relay your mail through a server you provide. Similar to DNSSEC, it is necessary to work with the organization that is hosting your email in order to ensure that when they send email on your behalf it is DMARC compliant.

DNS ALIASING RESTRICTIONS

What are Aliasing Resource Records?

The DNS has many resource records. When a DNS client requests information from a DNS server, the server returns information contained in resource records. For example, when the administrator on `example.bank` delegates authority for the `www.example.bank` subdomain to `admindc1.example.bank`, the following line would be added to the zones `example.bank` and `www.example.bank`:

```
www.example.bank. IN NS admindc1.example.bank.
```

This is an example of a resource record called a Name Server (NS) record. When DNS resolvers request the corresponding internet Protocol (IP) address for a FQDN, they request the Address (A) record. For example, the following A resource record, located in the zone `example.bank`, maps the FQDN of the server to its IP address:

```
www.example.bank. IN A 172.16.48.1
```

A zone file has many DNS resource records that provide a wide variety of data to be looked up by resolvers. Many computers have a domain name, which points to information about the host. This information (stored in those resource records) may include IP addresses, information about mail routing, etc. Computers may also have one or more domain name aliases, which are simply pointers from one domain name (the alias) to another (the official or "canonical" domain name).

The DNS provides a set of tools that makes it possible to have "aliases" for domain names. This can be convenient when running multiple services from a single IP address. You can, for example, point `ftp.example.bank` and `www.example.bank` to the A record for `example.bank`, which in turn points to the IP-address. Then, if you ever need to change the IP-address, you only have to change it in one place (A record). It's also convenient when you want to point one domain to another, for instance pointing `ourlongname.bank` to `example.bank`.

What is the fTLD Domain Security Requirement for Aliasing Resource Records?

Requirement #27:

> **DNS resource records (e.g., CNAME, DNAME, MX) may be used to alias to out-of-zone domains (i.e., non-fTLD Domains).**

DNS Resource Records may be used to alias from an fTLD Domain to a non-fTLD Domain, subject to compliance with the relevant Requirements (i.e., DNSSEC, TLS). When you alias, users may think they are on a trusted website providing security when they may not be because their true URL destination is masked. Application of the relevant Requirements when aliasing is to ensure the greatest protections for your customers and others accessing information at your fTLD Domain.

How do the Aliasing Resource Records Work?

We have seen previously how resource records work in the DNS. A client in the DNS makes a request for information from a DNS server for information. We have also seen that the DNS may have a substantial variety of information about a particular domain name. One of the elements that a zone file may contain is an alias for a domain name.

CNAME Records

Suppose that a client asks for the A record for `customerservice.example.bank.` The administrators of the zone that is authoritative for `example.bank` might insert a record that looks like this:

```
customerservice.example.bank IN CNAME www.example.bank.
```

In this case, `customerservice.example.bank` is the "alias" and `www.example.bank` is the "canonical name."

Name servers are designed to handle CNAME resource records in a completely different way than other records. When a name server looks up a name and finds a CNAME record for it in the zone file, it replaces the name with the canonical name and looks up the new name. For example, when the name server looks up `customerservice.example.bank`, it finds a CNAME record pointing to `www.example.bank`. It then looks up `www.example.bank` and returns the records stored for that name. There is a crucial feature about aliases such as `customerservice.example.bank`. You should always use the canonical name (e.g., `www.example.bank`) in the data portion of the resource record.

One of the uses of CNAME is to "redirect" a user from one domain to another. For instance, a user may have a domain name in more than one TLD. Here is an example of what such a CNAME record might look like:

```
www.example.net. IN CNAME www.example.com.
```

This CNAME record handles the case when somebody has both the domain name example.net and example.com. In this case the administrator of the DNS zone has "redirected" users who type in `www.example.net` to `www.example.com`. The user types in one domain name, but uses the DNS records associated with another domain name. In this case, the domain names are in different TLDs.

The ability to alias within an fTLD Domain is an important capability for some organizations that work with third-party providers for content distribution and Distributed Denial of Service mitigation services.

Administrators interested in other information about the configuration options related to CNAME should investigate the IETF RFC 2219 at:

```
https://tools.ietf.org/html/rfc2219
```

DNAME Records

The DNS makes it possible to go beyond simply providing an alias for a single domain name. It is also possible to create a record in a zone file that provides an alias for an entire subtree of a domain. In addition to the CNAME resource record type, there is also a Delegation Name record (or, simply the

DNAME record). The DNAME record works like the CNAME record, but for an entire domain name subtree. Here's an example:

```
customer.example.bank. IN DNAME servers.example.bank.
```

In this case, if a user tries to look up `www.customer.example.bank` the DNS server does the lookup instead for `www.servers.example.bank`. The server changes the domain name based on the mapping in the DNAME record and the reissues the query. The records provided from the query for `www.servers.example.bank` are then returned to the client. The server is asked to "synthesize" a new query based on the original `www.customer.example.bank` and turn it into `www.servers.example.bank`.

MX Records

One of the most commonly used resource records is the mail exchanger (MX record) used to specify the mail servers responsible for accepting emails on behalf of a receiving domain using the Simple Mail Transfer Protocol (SMTP).

Valid MX records are made up of two defining attributes. First, MX records must have a hostname that maps directly to one or more A or AAAA records in the DNS and cannot point to CNAME records. Second, MX records must be associated with a numerical value indicating preference and this determines the order in which sending servers contact the receiving server.

Domains may have one or more authorized mail servers and the MX record with the *lowest* numbered record is the most preferred. MX records of different preference values are useful to ensure backup mail servers can be located as an alternative should the primary mail server(s) go offline such as a network outage. The same preference number can be used for multiple mail servers to distribute the load across an array of mail servers.

An example of an MX resource record might look like this:

```
IN MX       10    mail1.example.bank
IN MX       10    mail2.example.bank
IN MX       20    mail3.example.bank
mail1       IN A  192.168.0.10
mail2       IN A  192.168.0.11
mail3       IN A  192.168.100.10
```

This example directs sending mail servers to attempt delivery of `example.bank` email to two servers, 192.168.0.10 and 192.168.0.11, with an equal preference of 10. If neither primary server is available, delivery to the third mail server at 192.168.100.10 will be attempted. This backup mail server is typically configured as a "store and forward" mail server; queueing incoming emails and immediately forwarding the mail to the primary (higher preference) mail servers when possible. The backup server is often located on a geographically different network to ensure availability should the primary servers go offline.

A thorough review of the MX record is beyond the scope of this document, but it should be recognized that domains that do not have one or more MX records are incapable of *receiving* email, but are capable of *sending* email (including being exploited to send spoofed email). fTLD requires the implementation of the email authentication protocol, DMARC, on all fTLD Domains and subdomains regardless of whether email is intended to be sent or received to protect them from spoofed email purporting to originate from the fTLD Domains and subdomains. For more information on email authentication see requirement #26.

NAME SERVER RESTRICTIONS

What is a DNS Name Server?

A DNS name server translates human-memorable domain names and host names into corresponding Internet Protocol (IP) addresses to locate computer systems and resources on the internet. DNS allows for two types of name servers:

- **Primary masters;** where the name server reads the zone data from a file on the server; and,
- **Secondary masters or slave;** where the name server gets the zone data from another name server authoritative for the zone, called its master server.

When a secondary master starts, it contacts its master name server and pulls the zone data over—this is called a zone transfer. Once the zone transfer is completed, both the primary master and the slave name servers are authoritative for that zone. There are good operational reasons for using additional DNS servers for zone replication:

- Added DNS servers provide zone redundancy, which makes it possible for DNS names in the zone to be resolved for clients if a primary server for the zone stops responding.
- Added DNS servers can be placed to reduce DNS network traffic. For example, adding a DNS server to the opposing side of a low-speed, wide area network (WAN) link can be useful in managing and reducing network traffic.
- Additional secondary servers can be used to reduce loads on a primary server for a zone.

What is the fTLD Domain Security Requirement for Name Server Host Names?

Requirement #28:

> **Name server host names must be in the parent zone.**

This requirement is intended to ensure that authoritative name servers are trusted and reliable.

Host Name Deployment for Registrars and Registrants

This is one of the easier Requirements to manage. A Name Server (NS) resource record indicates that a name server is authoritative for a zone. For instance, consider the records below:

```
example.bank.  IN NS dns1.example.bank. example.bank

               IN NS dns2.example.bank. example.bank

               IN NS backupns.example.bank
```

These records would state that there are three name servers for `example.bank.` To help ensure that these three name servers are trusted, the NS records are published in the parent zone.

For those organizations configuring and running their own DNS, control over the zone files is simple to arrange.

For organizations that outsource DNS and zone management, they will need to work with their third-party provider to ensure that this requirement is met. Operationally, even in situations where a third party is assisting with the configuration and management of the DNS, complying with this requirement is very straightforward.

URL REDIRECTION

What is URL Redirection?

URL redirection (e.g., URL forwarding, domain redirection) is a technique for making web content available at more than one website address. URL redirection is most often used to automatically forward a user from one URL location to another (source: `https://shorturl.example.bank` -> destination: `https://example.bank/longurl`), however for the purposes of the Requirements, fTLD does not distinguish between automatic redirection and manually initiated redirection (such as when a user clicks a link).

What is the fTLD Domain Security Requirement for URL Redirection?

Requirement #30:

> **URL Redirection to an out-of-zone domain must be made from the HTTPS version of an in-zone domain.**

URL Redirection is permissible in the following scenarios:

1. An unencrypted fTLD Domain website may exist for the sole purpose of redirecting to an encrypted fTLD Domain website. This practice ensures a positive, uninterrupted user experience of an fTLD Domain website.

   Using an HTTP 301 Moved Permanently status code on the source URL is recommended and, as a byproduct, will improve search engine optimization results.

   An fTLD Domain is an "HTTPS only" community. Registrants are encouraged to implement HTTP Strict Transport Security (HSTS) to declare that customer web browsers should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol.

2. The target domain's FQDN resides within an fTLD Domain (e.g., `www.bankname1.bank` and `www.bankname2.bank` redirect to `https://www.bankname3.bank`).

3. From an fTLD Domain to a non-fTLD Domain, provided the fTLD Domain is secured with HTTPS.

Registrants that redirect from domains in the secure zone to those outside are strongly encouraged to inform visitors of this action via an explicit message (e.g., a speed bump) to avoid confusion and to ensure that visitors understand they are leaving the fTLD Domain. Typically speed bumps are implemented with server side scripting language such as ASP, PERL, PHP, Python, or Ruby.

EMERGENCY SITUATIONS

What is the fTLD Domain Security Requirement for Emergency Situations?

Requirement #31:

Registrants are exempt from requirements 23, 25, 26, 27, 28, 29 and 30 in Emergency Situations

The Requirements are an important basis for enhancing the default security posture of registrants and to create a safer ecosystem to maintain trust and facilitate commerce. However, fTLD recognizes that despite best efforts, cyber attackers can from time-to-time bypass or render security controls ineffective, potentially causing significant harm to registrants and their customers.

These Emergency Situations are defined as present or imminent events such as:

- Incidents that threaten systematic security, stability and resiliency of registrant infrastructure;
- Unauthorized access to or disclosure, alteration, or destruction of registrant data or that of its customers; and
- An occurrence with the potential to cause a failure of registrant infrastructure.

In the course of responding to an Emergency Situation, registrants may decide their existing security posture is untenable and require a new approach to defeat or lessen the impact of a particular threat. Registrants are encouraged to take any action they deem appropriate to protect themselves, their infrastructure and their customers and the Requirements shall not be interpreted in a way to impede self-defense.

However, should defensive actions conflict or contravene the technical requirements, registrants must apply for, and, in the meantime, may operate under the presumption of being granted, a temporary exemption to one or all of the technical requirements.

Whether or not an exemption to the Requirements is necessary and appropriate, any Emergency Situation lasting longer than three business days requires written notification of the incident by the registrant to fTLD by the end of the third business day.

Requirement exemptions and/or notice of incidents lasting longer than three business days must be submitted through the Emergency Situation Notification Form found here: `ftld.com/incident-form/`.

OTHER CONSIDERATIONS

Network Resources

Although some organizations have already implemented measures such as DNSSEC or TLS, many of the Requirements may necessitate changes to the way you deliver services to your customers and the public. Public-facing services such as Web sites and electronic mail may require you to implement new controls as a result of the Requirements. In order to meet the Requirements, additional tasks and features will likely be required in the network.

Staffing Resources

The staffing resources needed to implement an fTLD Domain will vary according to the type and size of your organization. Larger organizations will likely have the expertise and resources internally to effect the implementation, and will therefore want to concentrate on communication and coordination.

Smaller organizations, or those that have outsourced related IT functions to third-party providers, will likely need to work closely with those providers to understand and communicate their evolving needs and ensure that they are met in a cost-efficient and timely way. Smaller organizations may not have the depth of in-house technical expertise needed to effect a full implementation or may need those resources to support current operations. These organizations may need to work with providers of existing functions such as email, Web services, API between services and banking records, and payment services to implement the Requirements.

Whatever the size or type of your organization, the first step will be to identify and work with both internal and external partners, including your domain name registrar and any third-party Web, communications and security providers.

The project to implement an fTLD Domain should also include externally-facing activities to communicate the benefits to your customers and business partners and help them adjust to the change associated with your planned use of an fTLD Domain, which is covered in the Planning and Communications Guide.

Using a project management methodology may be advisable to plan and implement the various Requirements; for example, publishing a basic DMARC record should be relatively straightforward and adopting TLS, if it is new to your organization, requires implementation across a range of Internet technologies and may be challenging.

Following the implementation of an fTLD Domain, you may require additional staff resources for on-going management and maintenance of the Requirements. An organization implementing DNSSEC for the first time, for example, will need to plan for the additional workload to maintain it. You may also need to dedicate resources from Web design, content management and networking.

Working with Third-Party Providers

Using a third-party provider does not obviate or transfer your responsibility for compliance with the Requirements. Any arrangement with a third-party provider supporting or hosting an fTLD Domain must be compliant with the relevant Requirements, and this may mean that the registrant needs to contractually bind its third-party providers to implement them. For example, if you use a vendor to manage marketing mail-outs to your customers then you likely need to modify your DMARC, DKIM and SPF records to include your authorizedthird-party email senders. It is your responsibility to ensure your third-party providers comply with the Requirements.

Ongoing expenditures will be necessary to maintain, monitor and demonstrate compliance with the Requirements. To illustrate, in a DNSSEC deployment, if you add a new name or level to your second-level domain. For example, if you create a new product or service and a new name to go along with it—`www.newservice.yourbank.bank`—you will need to re-sign the zone and re-publish it. If you change mail servers or use third-party email senders you will need to immediately update the DMARC record to reflect this.

You may wish to investigate specialized DNS or email authentication providers that can meet the Requirements, and find new partners to facilitate your implementation. New third-party providers have emerged whose business is built around helping clients meet these kinds of Requirements. fTLD has approved a number of third-party providers that can assist you in complying with the Requirements and they are accessible at `ftld.com/third-party-provider-program`.

Implementation Sequencing

Implementation and use of your fTLD Domain must be in compliance with all relevant Requirements. The timing of the activities required to implement the Requirements will be different for every organization. A large organization may be in full control of its own IT staff and can prioritize and schedule according to its own objectives. A smaller organization may not directly control the necessary third-party resources or be able to set a fully directed implementation process. In all cases, you must meet each requirement before finalizing the launch and use of your fTLD Domain.

KEEPING UP TO DATE WITH THE SECURITY REQUIREMENTS

fTLD will periodically review and update the Requirements. As these updates occur, you will be provided reasonable notice to implement the updated, relevant Requirements to your fTLD Domain,

The fTLD security framework will continue to evolve to stay ahead of the threats it is designed to mitigate. As new technologies emerge that improve the trust between customers and providers of financial services, the Requirements and best practices will also change.

Our promise to you is that fTLD will continue work collaboratively with industry and security experts to ensure that the Requirements of an fTLD Domain represent proven measures to ensure your security and your customers' well-placed trust.

fTLD
Registry Services, LLC