KeyForge:

Mitigating Email Breaches with Forward Forgeable Signatures

Michael A. Specter
specter@mit.edu // mspecter@
Sunoo Park
Matthew Green

This talk, I'll:

- Introduce ***Forward Forgeable Signatures*** (FFS)
  - Signatures that become **unverifiable** after a set wall-clock time limit
- Introduce (informally) two *constructions* of FFS's:
  - **KeyForge & TimeForge**

Motivation

Motivation:

How can we **disincentivize** email theft?

Email's Value:

1. Email is near ubiquitous
2. Email has metadata
   a. Location (originating IP), activity, email client (including OS)
3. High attack surface; many ways of getting into an account
4. Email is *undeniable*

For example:

The New York Times

Private Security Group Sa… Was Behind John Podesta… Hack

Hillary Clinton's campaign chairman, John Podesta, readin… the Clinton campaign plane last month. Doug Mills/The New …

BBC | Home | News | Sport | Reel | Worklife | Travel | More | Search

NEWS

Home | Coronavirus | Video | World | US & Canada | UK | Business | Tech | Science | Stories | More

Entertainment & Arts

US & Canada

18 revelations from Wikileaks' hacked Clinton emails

27 October 2016

US election 2016

WIRED | BACKCHANNEL | BUSINESS | MORE | SUBSCRIBE

APRIL GLASER | SECURITY | 07.27.2016 09:30 AM

Here's What We Know About Russia and the DNC Hack

Russia was very likely responsible for the hack that has upended the DNC.

How did Wikileaks know that these messages were **real**?

Email isn’t **deniable**.

Cryptographic Verification via

WikiLeaks - The Podesta Email

https://wikileaks.org/podesta-emails/emailid/5205

WikiLeaks Leaks News About Partners Search Sho

Return to search

View email | View source

This email has also been verified by Goog 2048-bit RSA key

Re: From time to time I get the questions advance

From:jpalmieri@hillaryclinton.com
To: donna@brazileassociates.com, balcantara@hillaryclinton.com
CC: john.podesta@gmail.com, Minyon.Moore@deweysquare.com
Date: 2016-03-12 19:41
Subject: Re: From time to time I get the questions in advance

WikiLeaks Leaks News About Partners Search Shop Donate Submit

04 November 2016

Domain Keys Identified Mail, or DKIM, is a highly regarded email security system that can be used to independently authenticate the contents and sender of an email that uses it.

DKIM was developed and is widely deployed as an email server anti-spam mechanism, including on Gmail.com and HillaryClinton.com. DKIM-enabled mail servers cryptographically sign the emails they relay so that the recipients' mail servers can authenticate them. DKIM has the beneficial side-effect of causing messages to become "cryptographically non-repudiable"; that is, after the email has been sent, the sender cannot credibly repudiate the message and say that it is a forgery. A DKIM mail server creates a cryptographically strong proof attesting to the authenticity of the email, which it adds to each of the headers of each email it sends. This cryptographic proof can then be tested by anyone who obtains a copy of the email.

In the Podesta email archive, many of the politically significant emails use DKIM authentication, including several contentious emails which some politicians have attempted to repudiate. These mails are, in fact, signed by HillaryClinton.com's email provider, Google. This authentication is on top of the journalistic validations of the email archive already carried out by WikiLeaks.

For example, an email that DNC Chair Donna Brazile falsely claimed to be "doctored by Russian sources" is in fact validated. Similarly validated is the email referencing a future appointment of Tim Kaine as Vice-President of the United States, which Mr Kaine publicly attempted to allege was fake. Both these emails have been secondarily validated by Google as being sent, with the content exactly as published by WikiLeaks.

You can see on our pages a notice when an email has additional validation through DKIM. What does this mean? It means that the content of the email has been independently verified to be authentic in its entirety and this verification process can be performed by anyone. Most DKIM- authenticated emails are essentially indisputable.

You can see the DKIM signatures on emails that have them by clicking on the "view source" tab and looking at the email's headers for "DKIM-Signature:", for example:

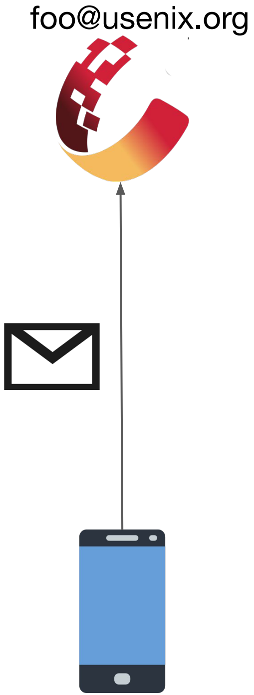
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=gmail.com; s=20120113; h=mime-version:in-reply-to:references:date:message-id:subject:from:to:cc:content-type; bh=LMXa7c2eNKxvY4PrcbVDYCrY8kl1NpfrYq0D1CP9cM0=; b=cGVf2qJhuzMfD3qsH8q9pABcHFE3lI1t/sw8jT3fNJ.....==

Technical note:

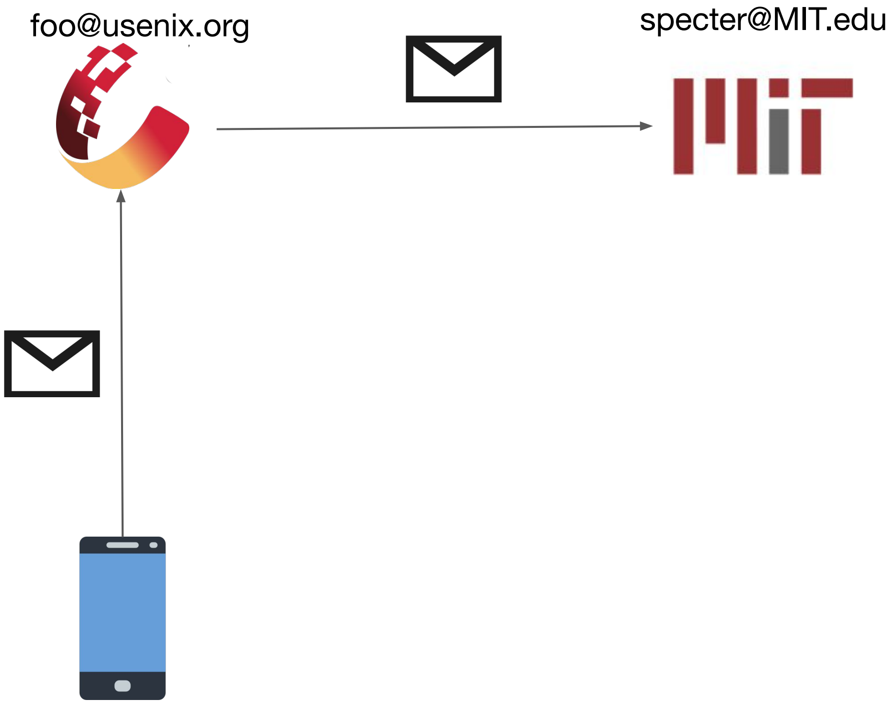
Due to the complexities of modern email systems, and the fragility of cryptographic signatures, any formatting or character change to a

DomainKeys Identified Mail (DKIM)

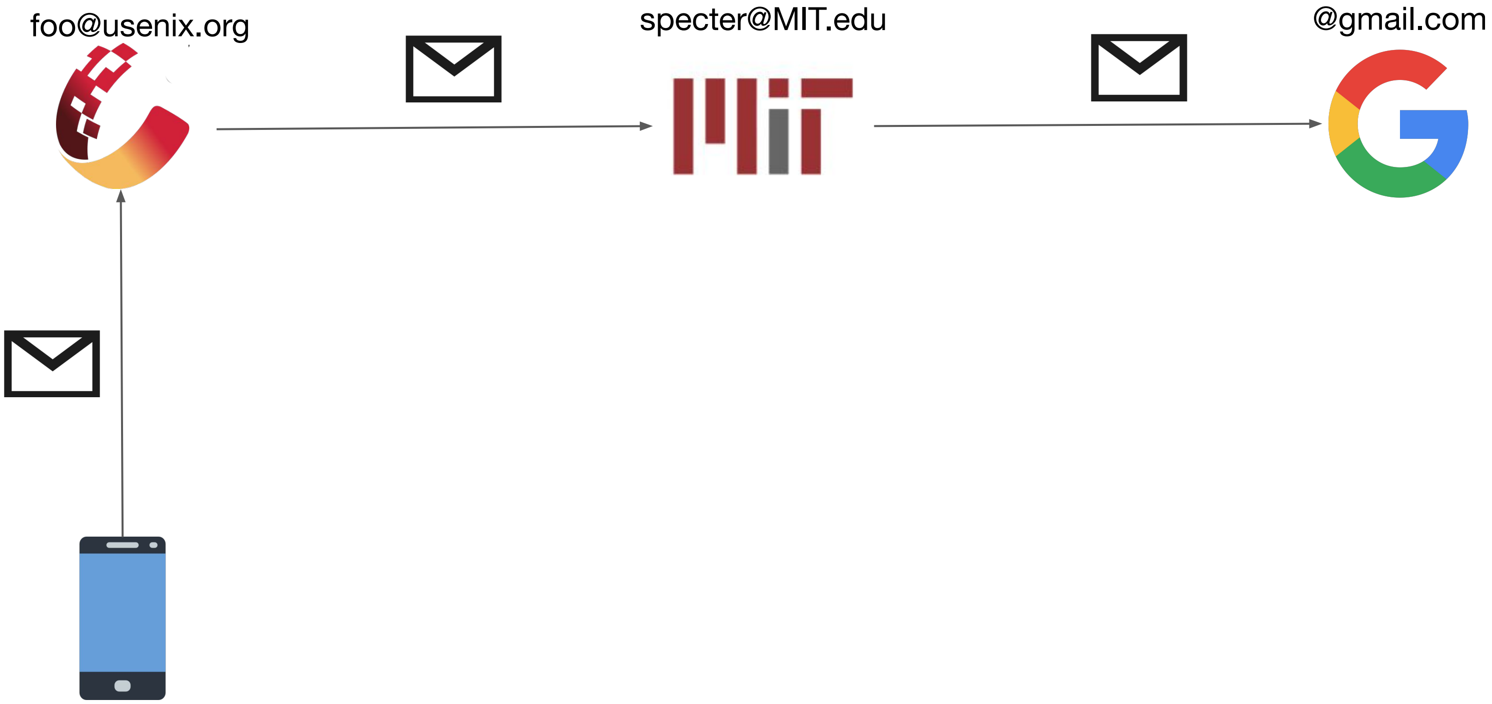
DKIM’s Goal is *Just* to Stop Spam



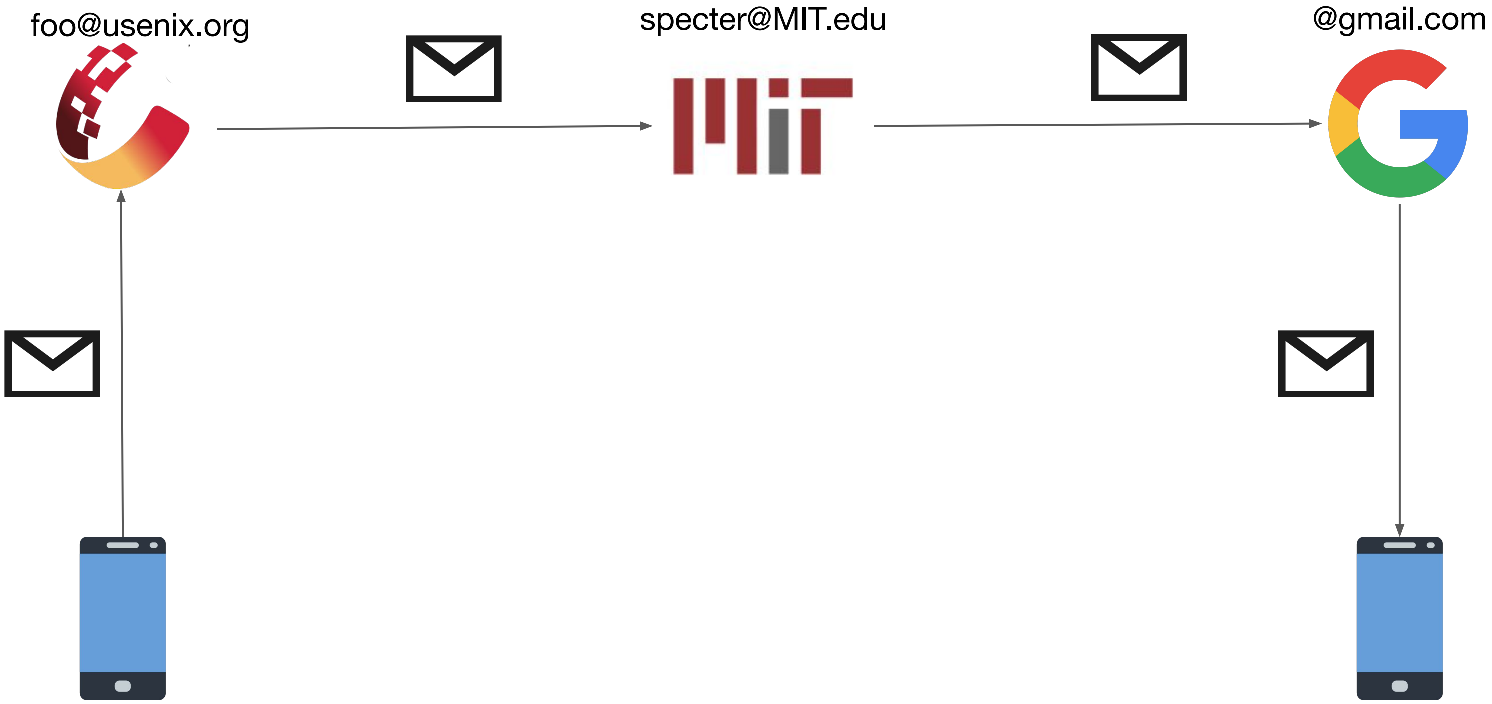
DKIM's Goal is *Just* to Stop Spam



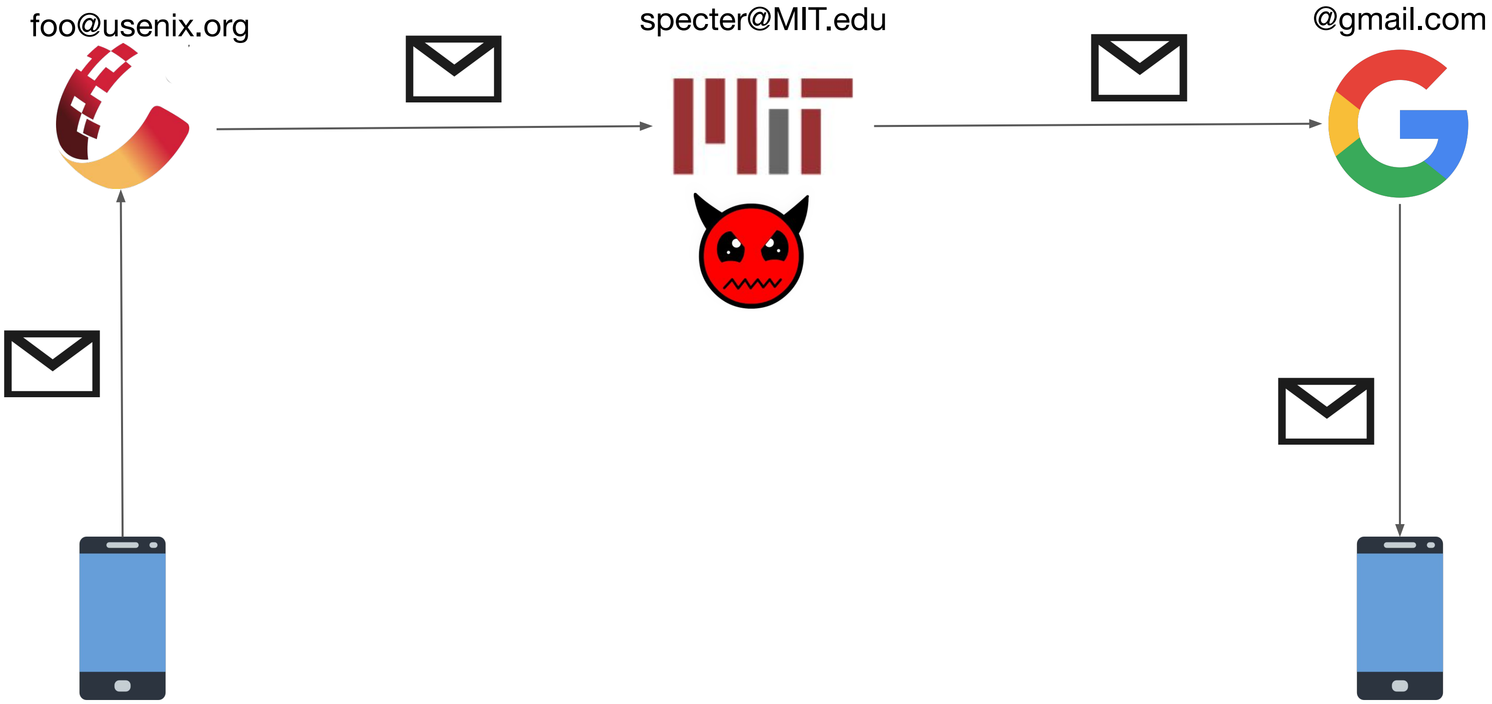
DKIM's Goal is *Just* to Stop Spam



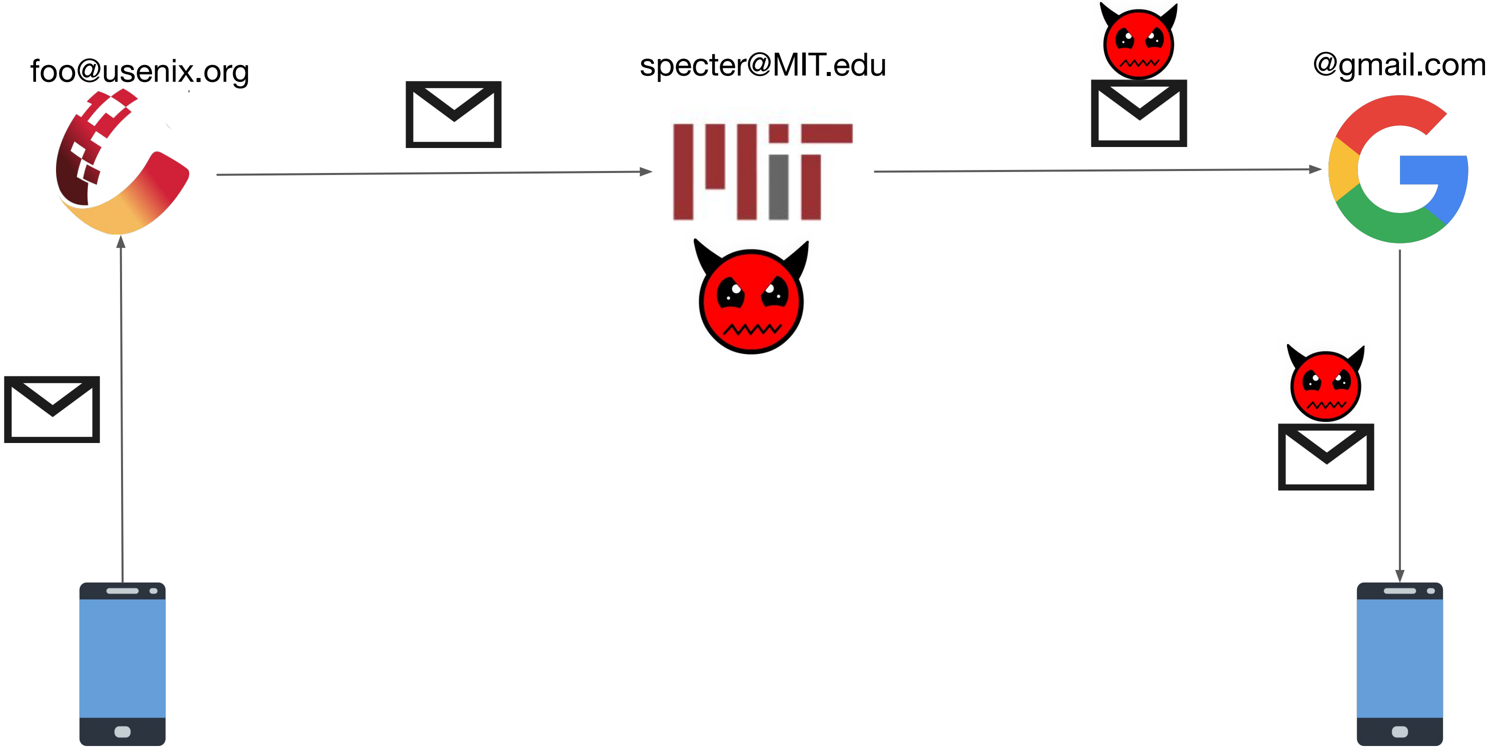
DKIM's Goal is *Just* to Stop Spam



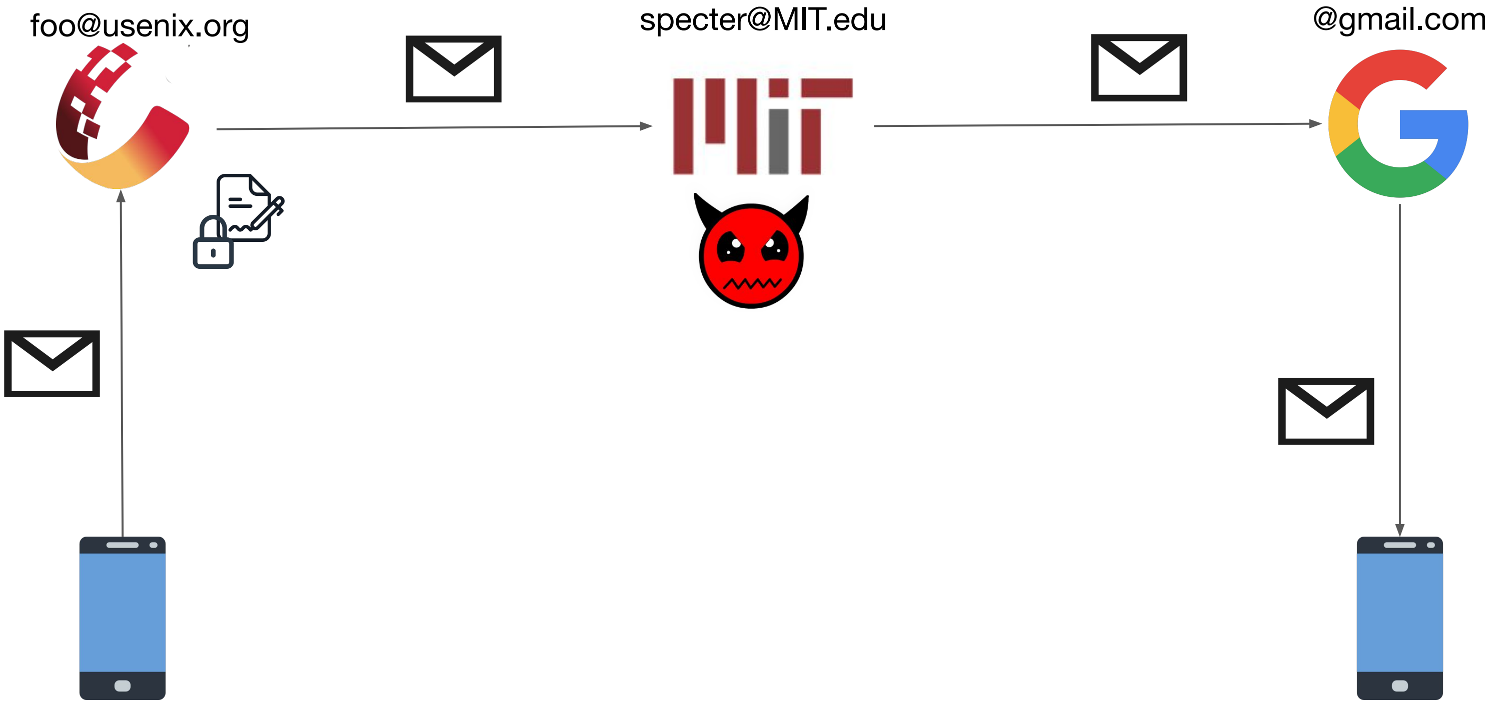
DKIM’s Goal is *Just* to Stop Spam



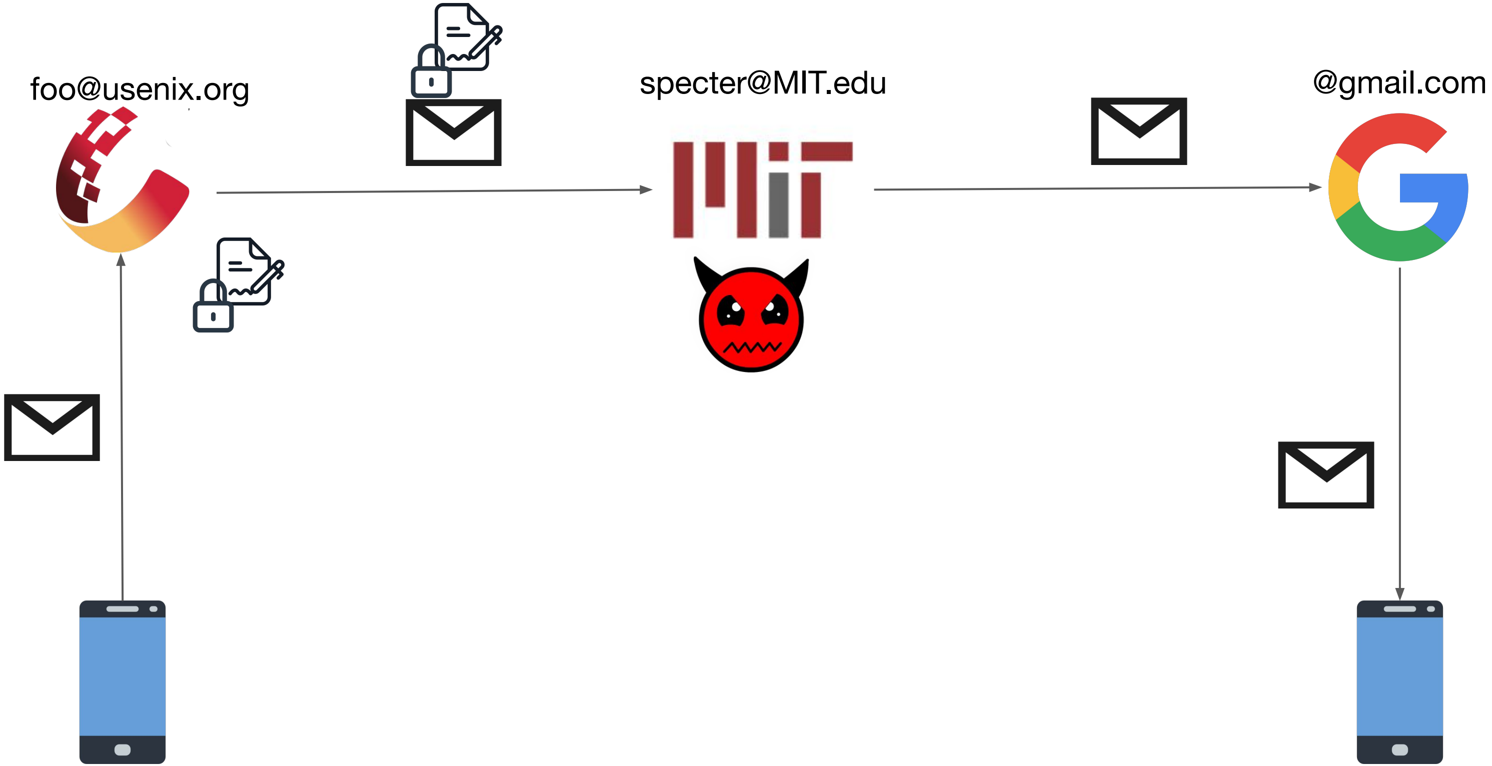
DKIM's Goal is *Just* to Stop Spam



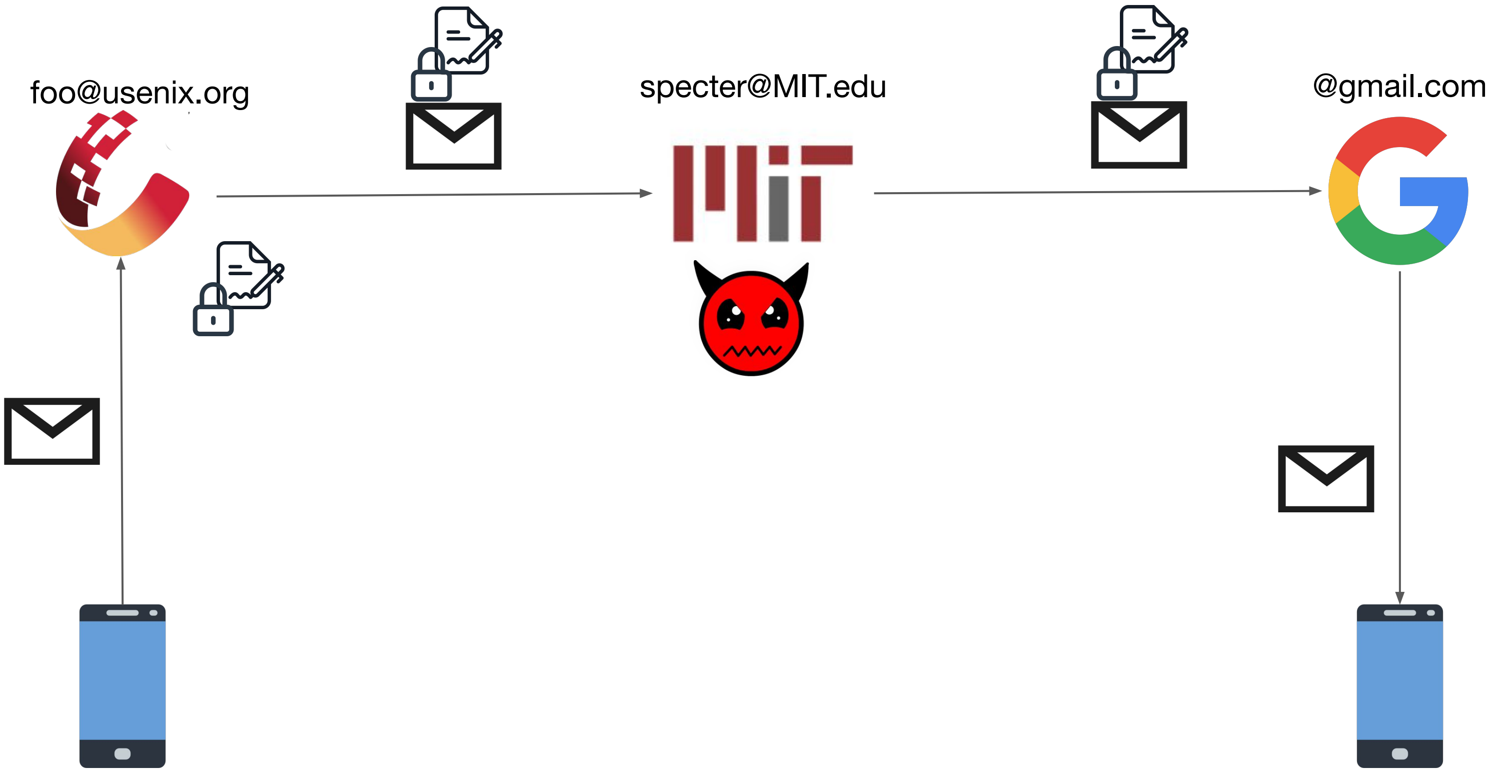
DKIM's Goal is *Just* to Stop Spam



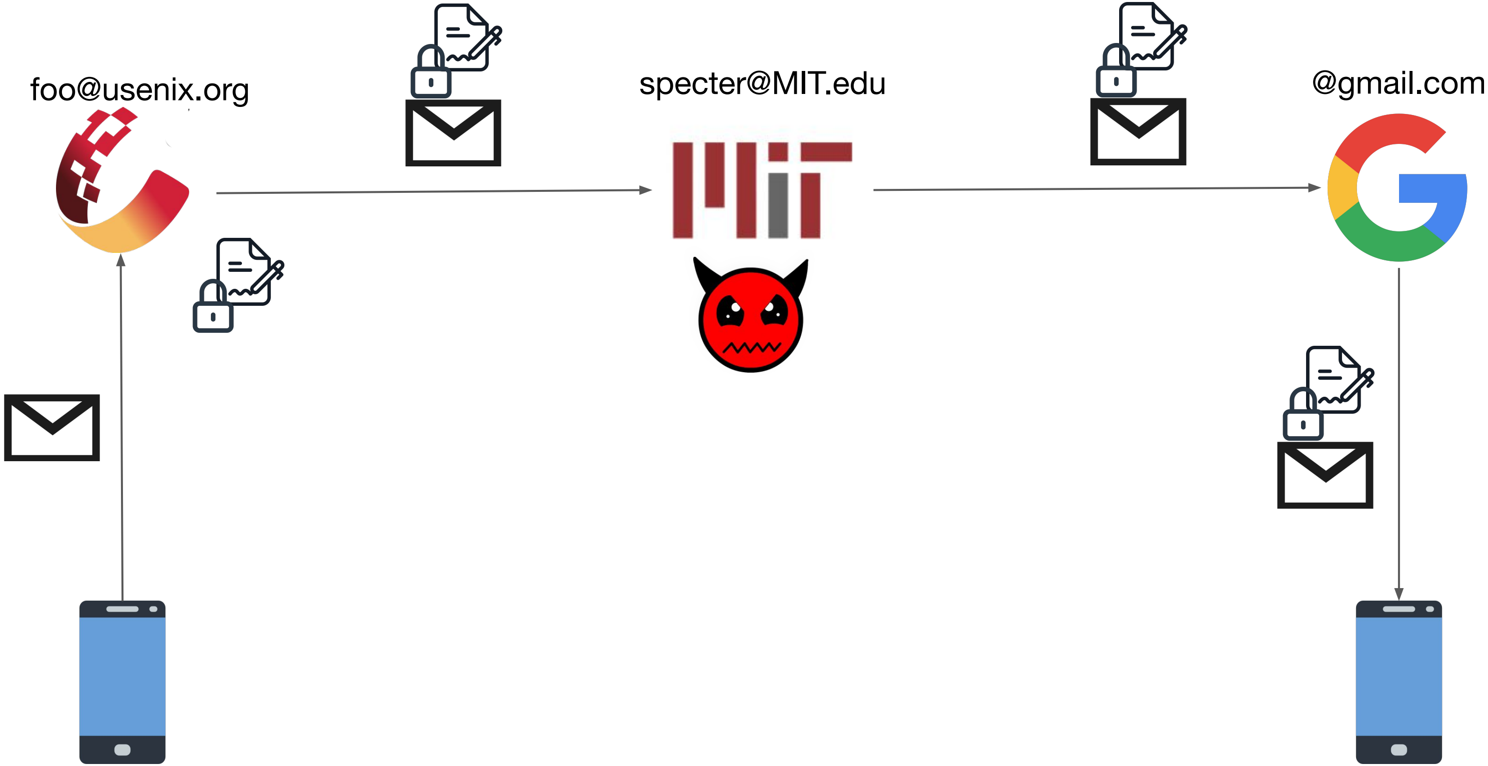
DKIM's Goal is *Just* to Stop Spam



DKIM's Goal is *Just* to Stop Spam



DKIM's Goal is *Just* to Stop Spam



As an **unintended side effect**, DKIM makes email **non-repudiable**.

Is it possible to ensure that **email is deniable**
While keeping DKIM's spam-resistance?

Why is this *hard*?

- Mostly Synchronous
- Sender knows the destination
- Use a Deniable Authenticated Key Exchange (DAKE)!

- Asynchronous & **non-interactive**
- Sender can’t know the destination server
- Inherently breaks DAKEs!

Known open problem since the original DAKE paper!

Off the Record (Borisov et al. 2004)

Long-lived public keys

- DKIM keys are stored in DNS.
  - One cannot update DNS *that* regularly
- Rotation is *hard*
- Google’s keys have been the same since 2016:

```
5. specter@Michaels-MacBook-Pro: ~ (zsh)
------------------------------------------------------------
~ » dig 20161025._domainkey.gmail.com txt                                specter@Michaels-MacBook-Pro

; <<>> DiG 9.10.6 <<>> 20161025._domainkey.gmail.com txt
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61767
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;20161025._domainkey.gmail.com. IN      TXT

;; ANSWER SECTION:
20161025._domainkey.gmail.com. 299 IN   TXT     "k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAviPGBk4ZB64UfSqW
yAicdR7lodhytae+EYRQVtKDhM+1mXjEqRtP/pDT3sBhazkmA48n2k5NJUyMEoO8nc2r6sUA+/Dom5jRBZp6qDKJOwjJ5R/OpHamlRG+YRJQqR" "tqEgS
iJWG7h7efGYWmh4URhFM9k9+rmG/CwCgwx7Et+c8OMlngaLl04/bPmfpjdEyLWyNimk761CX6KymzYiRDNz1MOJOJ7OzFaS4PFbVLn0m5mf0HVNtBpPwWu
CNvaFVflUYxEyblbB6h/oWOPGbzoSgtRA47SHV53SwZjIsVpbq4LxUW9IxAEwYzGcSgZ4n5Q8X8TndowsDUzoccPFGhdwIDAQAB"
```

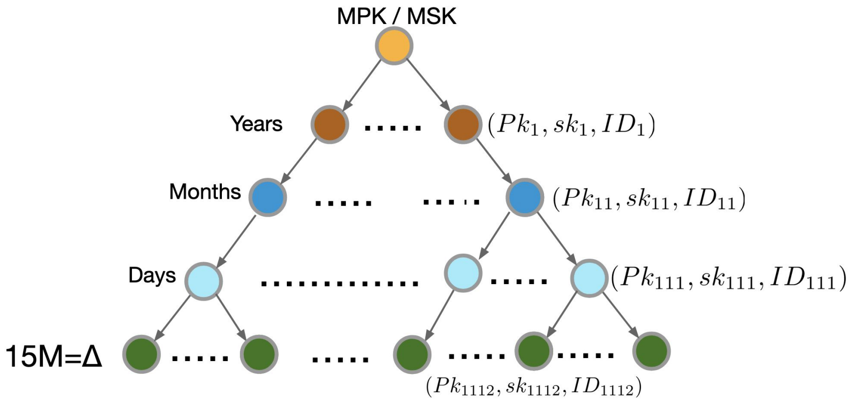
Our Solution:
Forward Forgeable Signatures!

Key Idea: Forward Forgeable Signatures!

- DKIM signatures are only really useful for the first ~15 minutes
- Signatures “expire” -- become forgeable -- after a delay Δ.

In the paper, we present two constructions:

KeyForge:

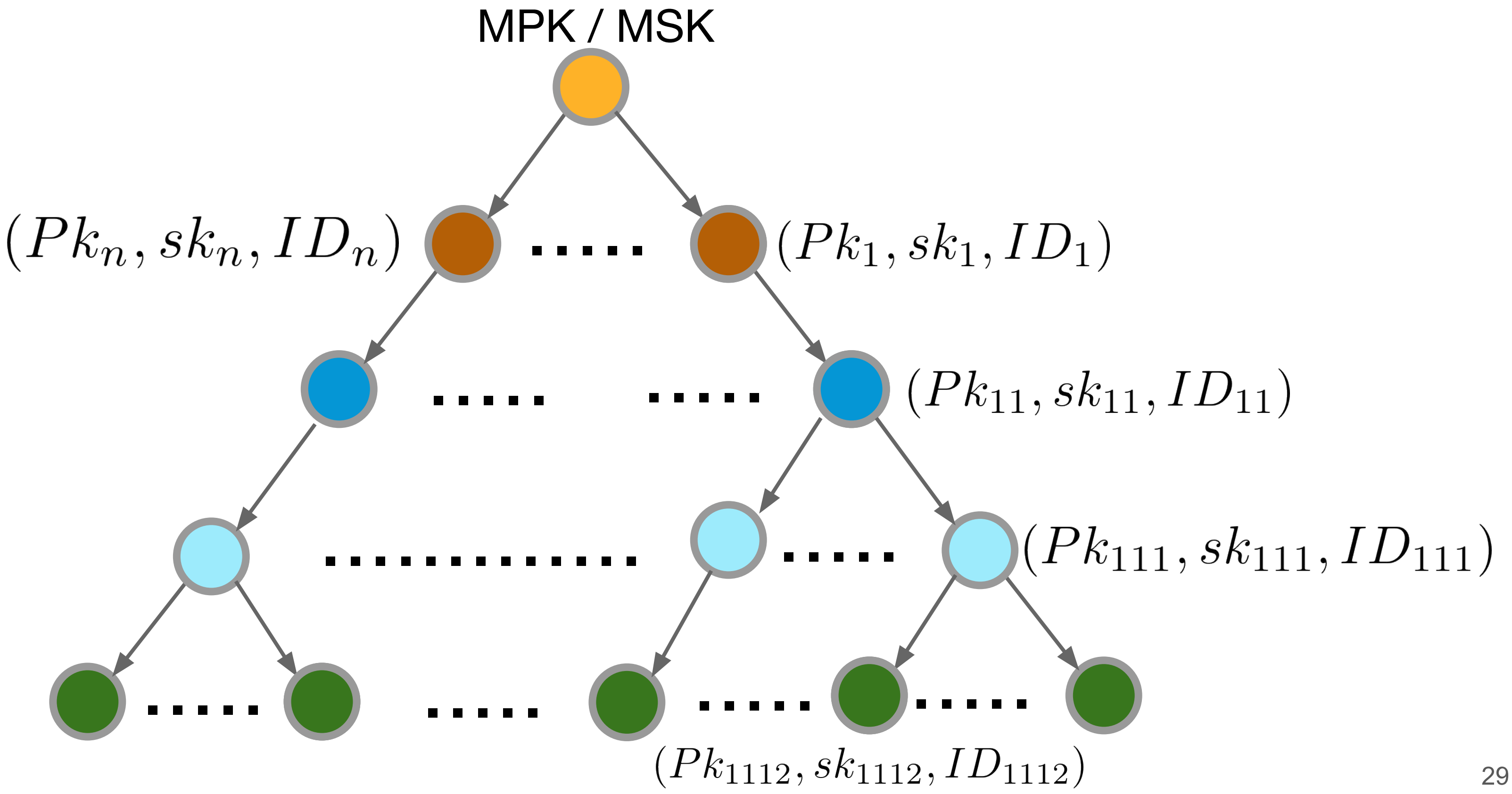


TimeForge:

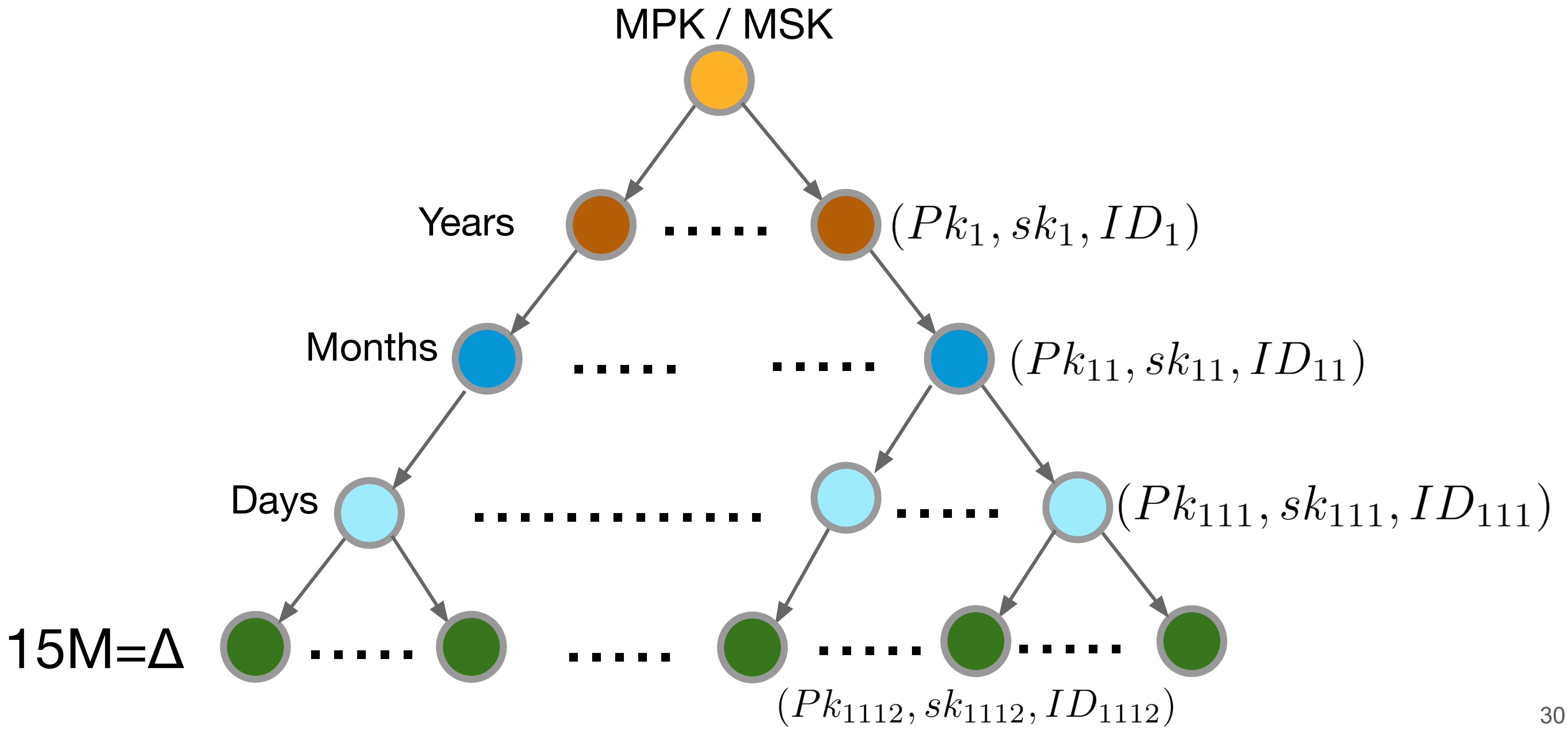
KeyForge: Intuition

- Sign, just like you would with DKIM
- ...But we'll make it easy to derive infinite keys
  - With only one public key
- Publicly release private keys when time elapses (Δ)
- Use a *Hierarchy of Keys* (HIBS)
  - $Secret_{child} = Hash(ID_{child} \,||\, Secret_{parent})$

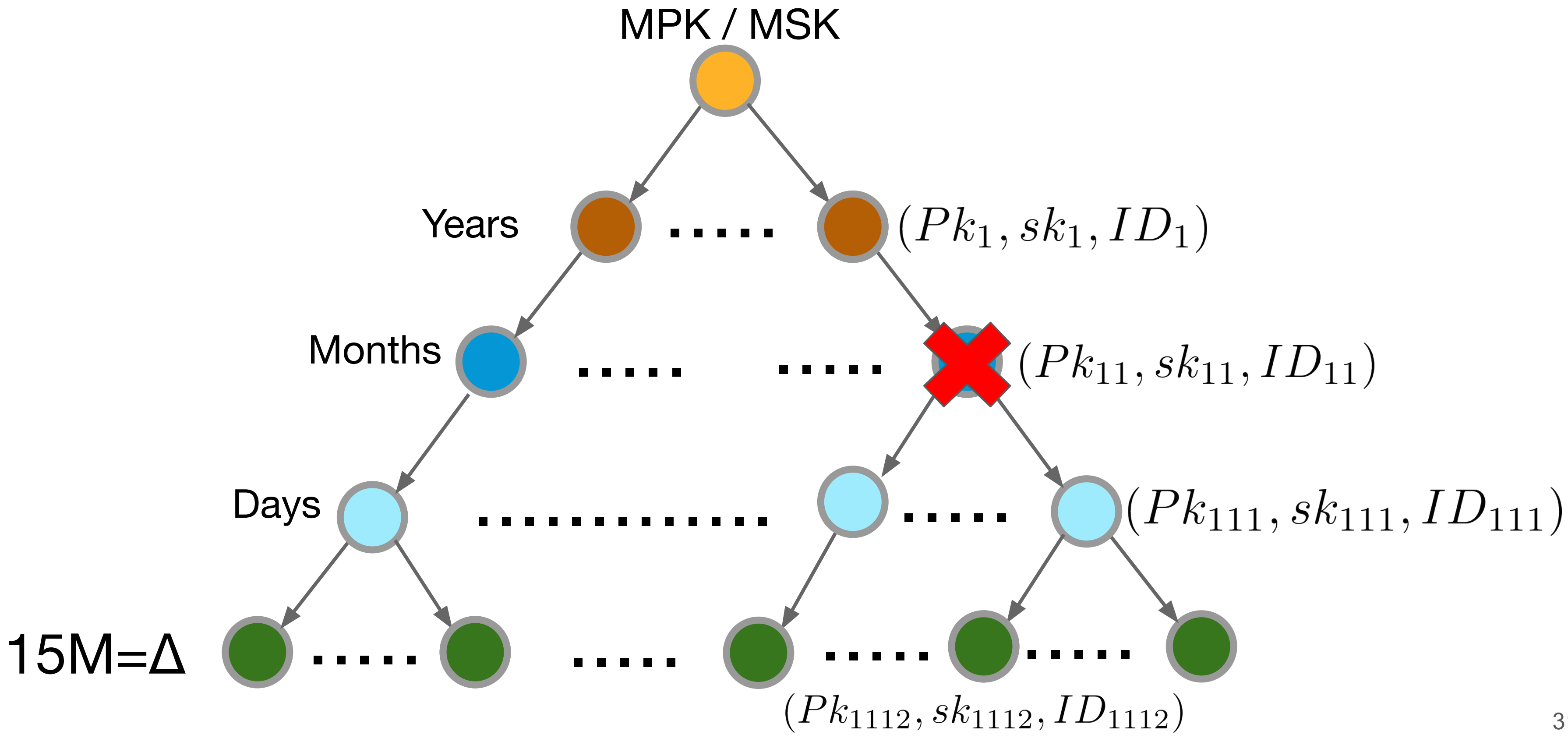
KeyForge: Intuition



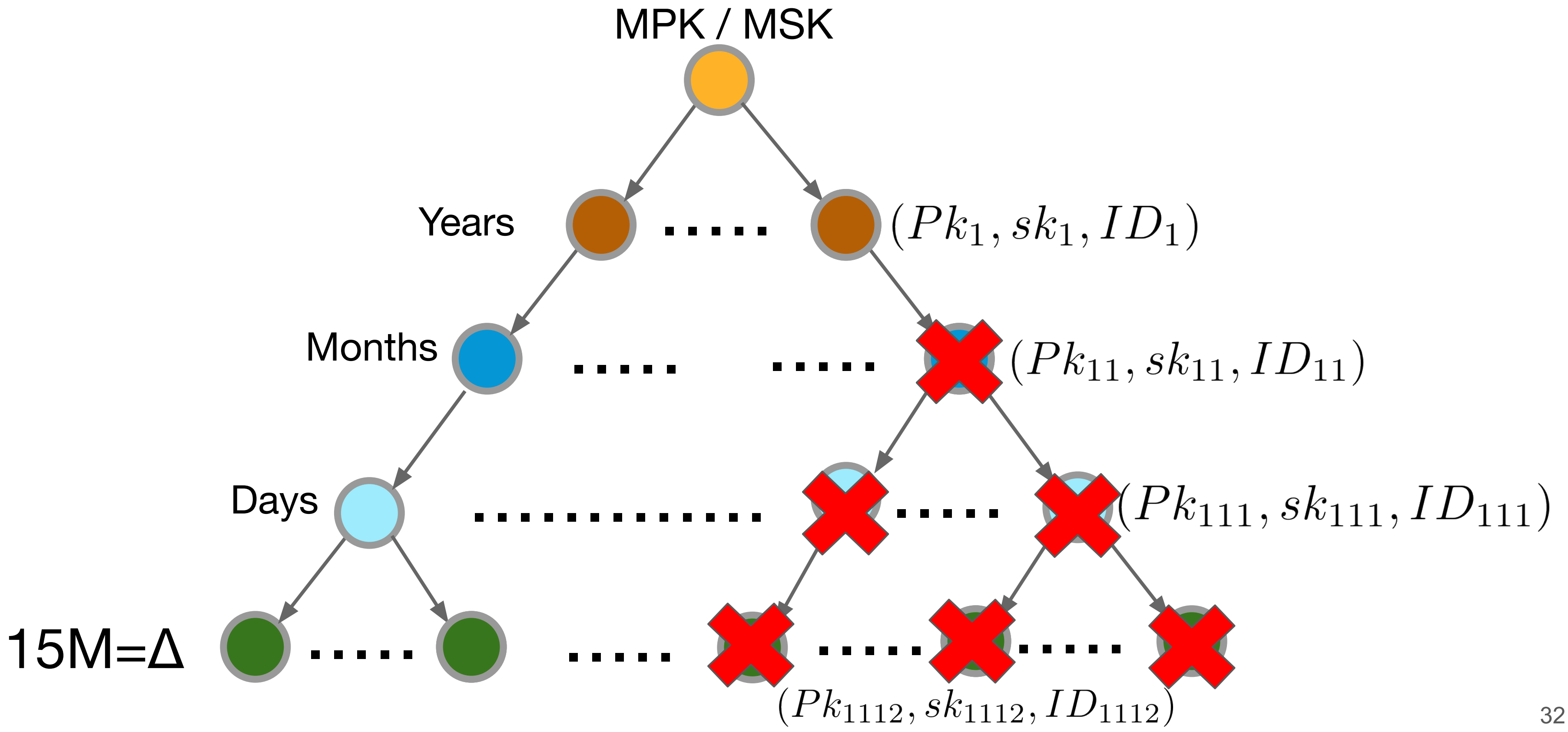
KeyForge: Intuition



KeyForge: Intuition



KeyForge: Intuition



TimeForge

Can we minimize expiry keys?

TimeForge: Intuition

Create a proof, given a message **m**:

1. The sending server has signed **m**

OR

2. The time has expired.

TimeForge: Intuition

Create a proof, given a message **m**:

Signature(m) 1. The sending server has signed **m**

OR

2. The time has expired.

TimeForge: Intuition

Create a proof, given a message **m**:

Signature(m)

1. The sending server has signed **m**

WIPoK

OR

2. The time has expired.

TimeForge: Intuition

Create a proof, given a message **m**:

Signature(m)

1. The sending server has signed **m**

WIPoK

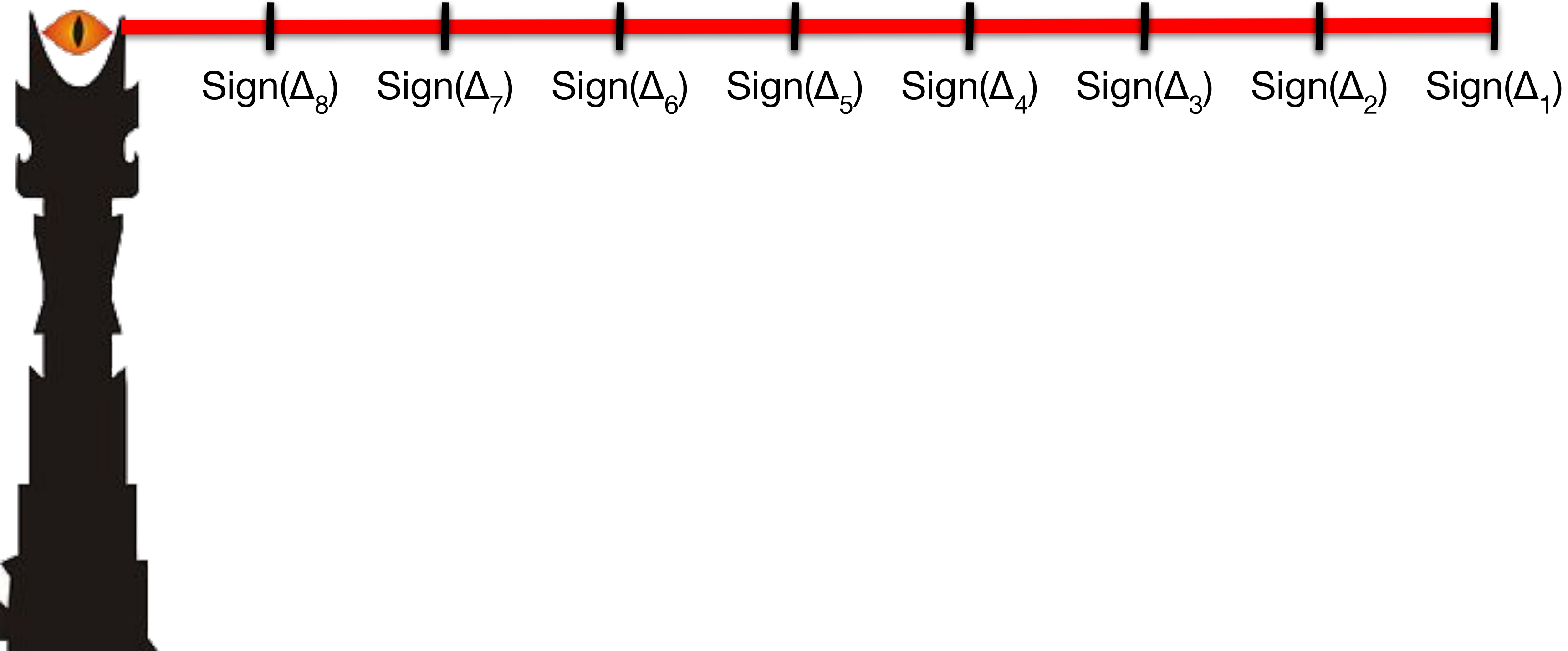
OR

Aliens?

2. The time has expired.

TimeForge: Publicly Verifiable Time Keeper

A beacon signs and publishes a monotonically increasing timestamp:



TimeForge: Publicly Verifiable Time Keeper

A beacon signs and publishes a monotonically increasing timestamp:

Sign(Δ_8) Sign(Δ_7) Sign(Δ_6) Sign(Δ_5) Sign(Δ_4) Sign(Δ_3) Sign(Δ_2) Sign(Δ_1)

Signature(m) — 1. The sending server has signed **m**

WIPoK — OR

A signature from a PVTK on t>Δ — 2. The time has expired.

Evaluation

- We implemented both protocols
  - ~3k lines of Go and C
- KeyForge appears to be practical!
  - Relatively small time increase in signing and verification.
  - Signatures are actually *smaller* than DKIM's RSA
- TimeForge is a promising prototype
- See paper for details!

KeyForge:

Mitigating Email Breaches with Forward Forgeable Signatures

Michael A. Specter
Sunoo Park
Matthew Green

specter@mit.edu // mspecter@