Today in Cryptography (5830)

Digital signatures
RSA signatures and full domain hash
PKI

TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Check CERT using CA public verification key
Check σ

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

← CERT = (pk_s , signature over it)

Pick random x
$X = g^x$

← p , g , X , σ = Sign(sk_s, p || g || X)

Pick random y
$Y = g^y$

Y →

$PMS = g^{xy}$

$PMS = g^{xy}$

ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Check CERT using CA public verification key
Check σ

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

← CERT = (pk_s , signature over it)

Pick random x
$X = g^x$

← p , g , X , σ = Sign(sk_s, Nc||Ns||p||g||X)

Pick random y
$Y = g^y$

Y →

$PMS = g^{xy}$

$PMS = g^{xy}$

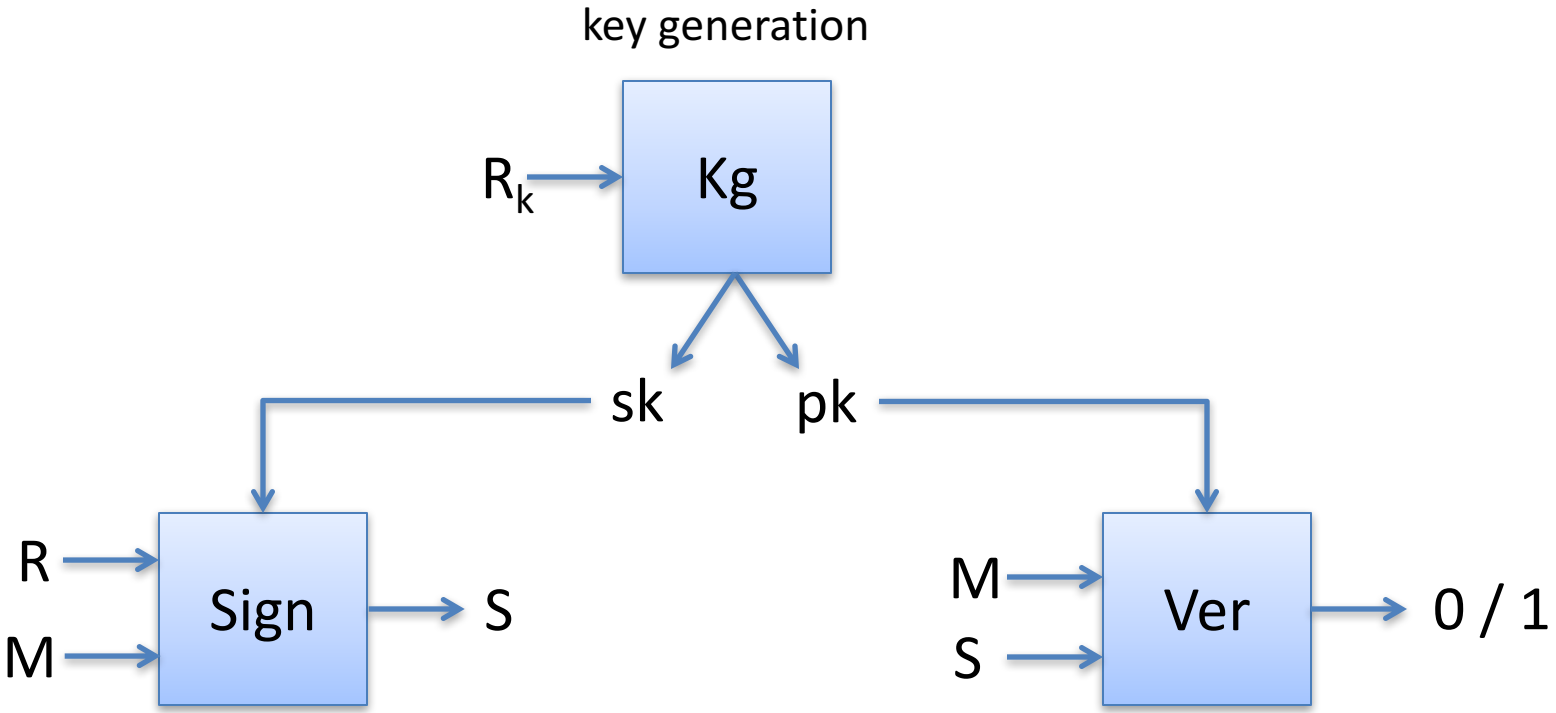
ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

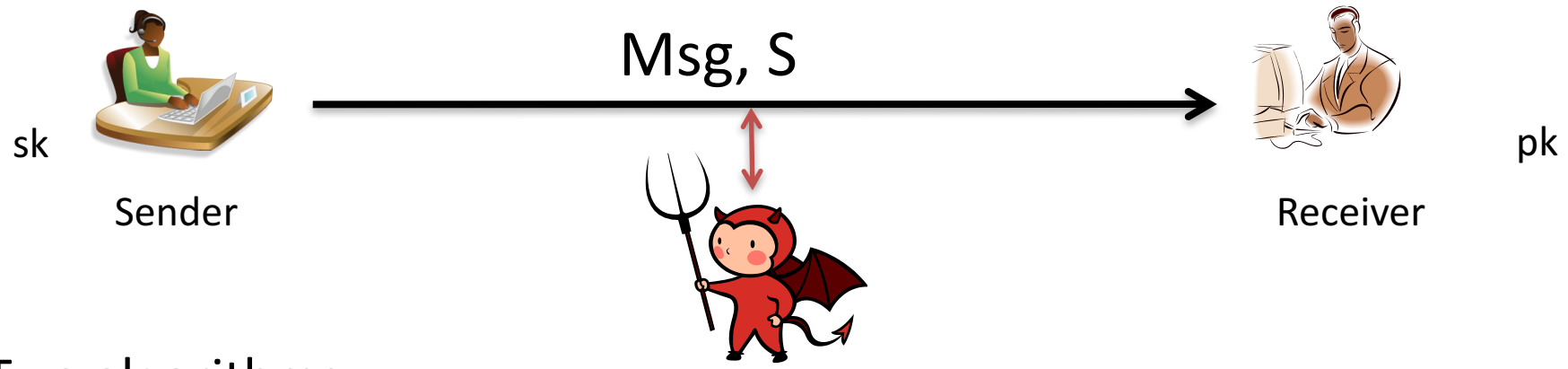
Digital signatures



Anyone with public key can verify a signature

Only holder of secret key should be able to generate a signature

Digital signatures



Two algorithms:

(1) Key generation outputs (pk,sk)

(2) Sign (sk,Msg) outputs a signature S (may be randomized)

(3) Ver(pk,Msg,S) outputs 0/1 (invalid / valid)

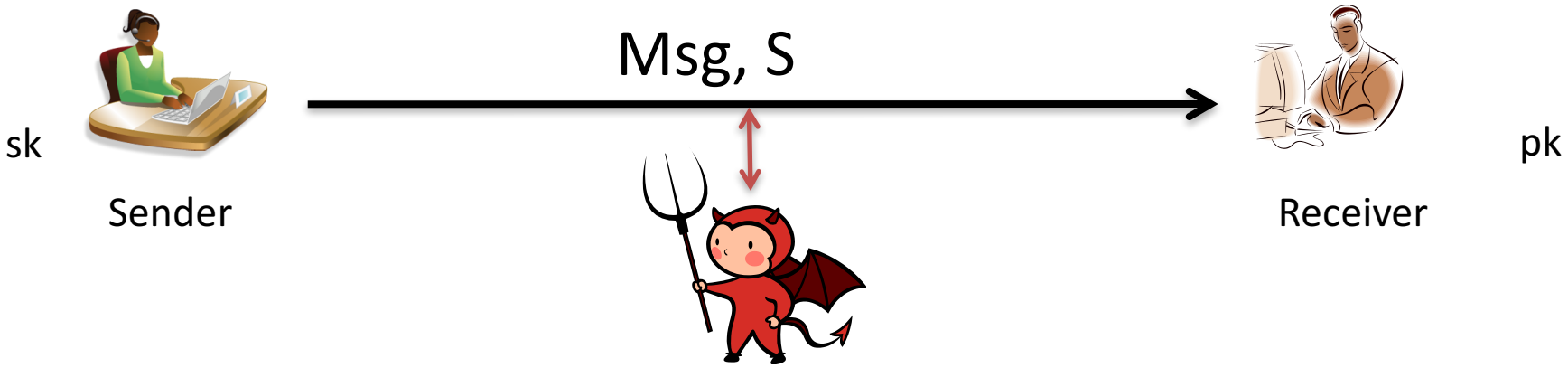
Correctness: Ver(pk,Msg,Sign(sk,Msg)) = 1 always

Security: No computationally efficient attacker can forge valid signature for a new message even when attacker gets

$(Msg_1 , S_1) , (Msg_2 , S_2), \dots , (Msg_q , S_q)$

for messages of his choosing and reasonably large q.

Digital signatures



"Raw" RSA as a signature scheme:

Key generation gives (N,e) , (N,d)
Sign((N,d),M) = M^d mod N
Verify((N,e),M,S) checks if S^e mod N = M

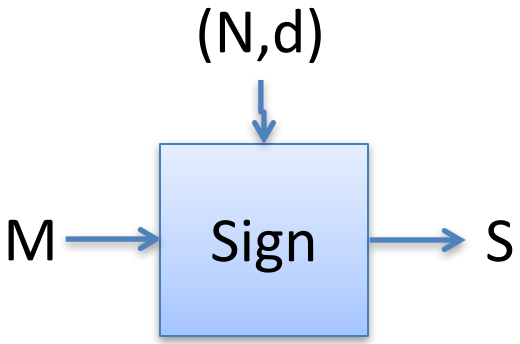
Secure? No!

PKCS #1 v1.5 RSA signing

Kg outputs (N,e),(N,d) where $|N|_8 = n$

Want to sign using hash with output length m bytes

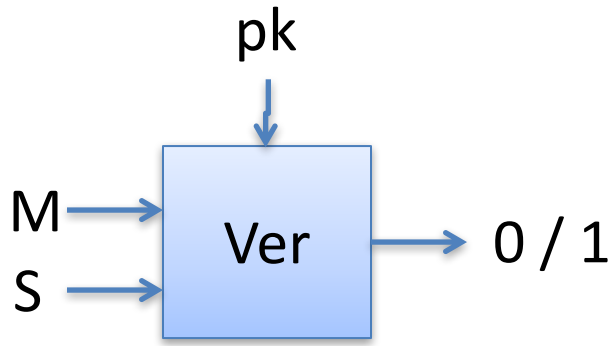
Let p = n – m - 3



Sign((N,d), M)

Y = 00 || 01 || FF^p || 00 || H(M)

Return Y^d mod N



Verify((N,e), M, S)

$Y = S^e$ mod N ; aa||bb||cc||dd||h = Y

If (aa ≠ 00) or (bb ≠ 01) or (cc ≠ FF^p)

or (dd ≠ 00)

Return error

Return H(M) = h

PKCS#1 v1.5 digital signature security

Padding oracle attacks that work against RSA

- PKCS#1 v1.5 decryption can be used to forge
- PKCS#1 v1.5 signating oracle can be used to forge

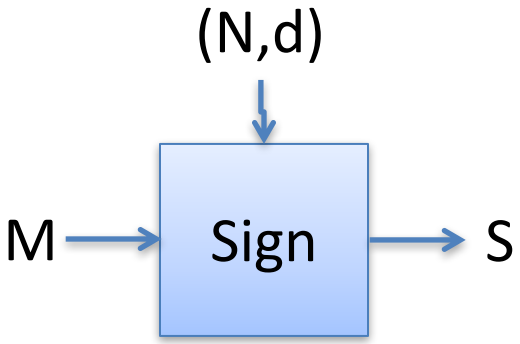
If one uses same RSA key pair for signing and encryption, then decryption and signing oracles offer similar forging / decryption opportunities

- Don’t use same keys for signing and encryption

Full Domain Hash RSA

Kg outputs pk = (N,e) , sk = (N,d) where $|N|_8 = n$

H is hash with m-byte output k = ceil((n – 1)/m)

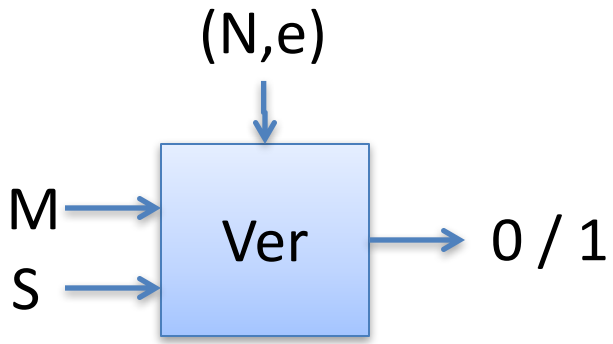


Sign((N,d), M)

X = 00 || H(1||M) || … || H(k||M)

$S = X^d \bmod N$

Return S



Ver((N,e), M, S)

$X = S^e \bmod N$

X' = 00 || H(1||M) || … || H(k||M)

If X = X' then

  Return 1

Return 0

Probabilistic Signature Scheme (PSS) provides stronger security bounds and also deployed now, see PKCS#1 v2

TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Check CERT using CA public verification key
Check σ

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

← CERT = (pk_s , signature over it)

Pick random x
$X = g^x$

← p , g , X , σ = Sign(sk_s,Nc||Ns||p||g||X)

Pick random y
$Y = g^y$

Y →

$PMS = g^{xy}$

$PMS = g^{xy}$

ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

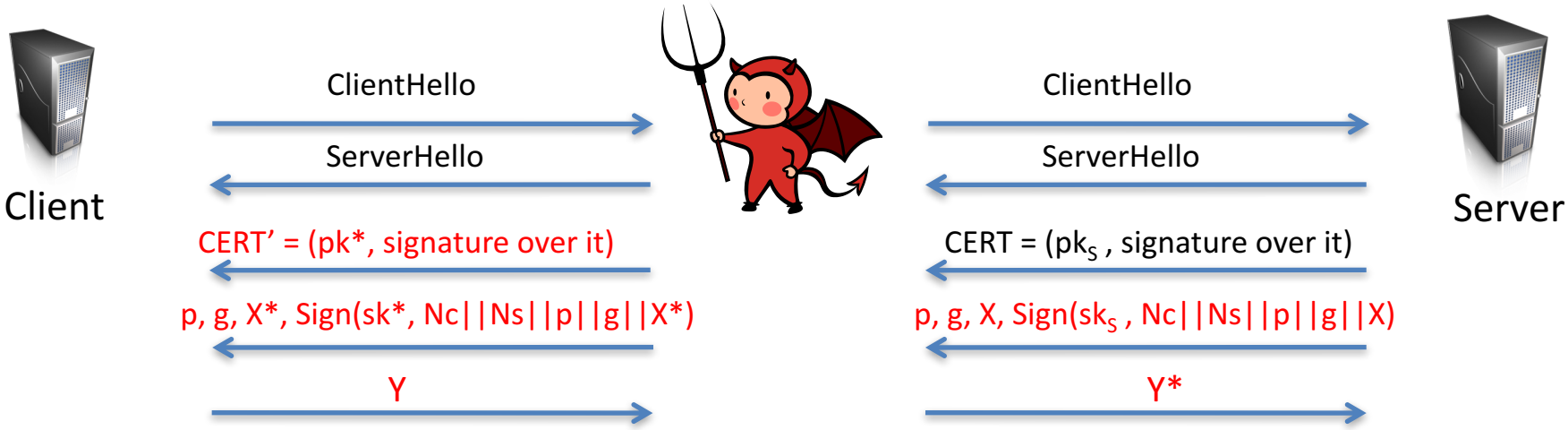
ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

Man-in-the-middle attacks

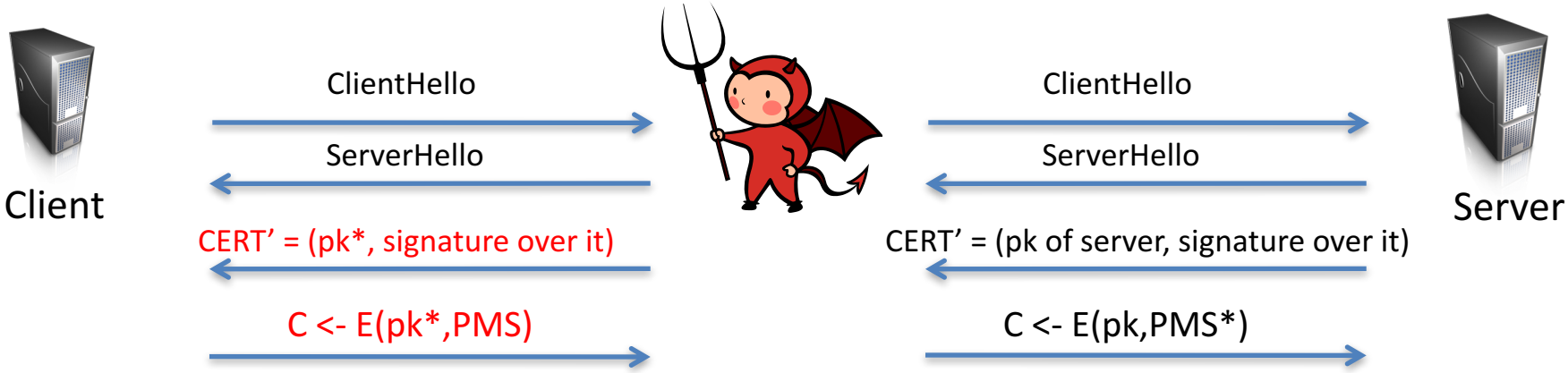
Suppose authentication vulnerability:
CERT can be forged, Client doesn't check CERT, etc.



Attacker can choose X*, Y*, so it knows discrete logs
Completes handshake on both sides
Client thinks its talking to Server
All communications decrypted by adversary, re-encrypted and forwarded to server

Man-in-the-middle attacks

Suppose authentication vulnerability:
CERT can be forged, Client doesn't check CERT, etc.



Attacker can choose pk*, thus knowing sk*
Completes handshake on both sides
Client thinks its talking to Server
All communications decrypted by adversary, re-encrypted and forwarded to server

Apple iOS <7.0.16 signature verification code

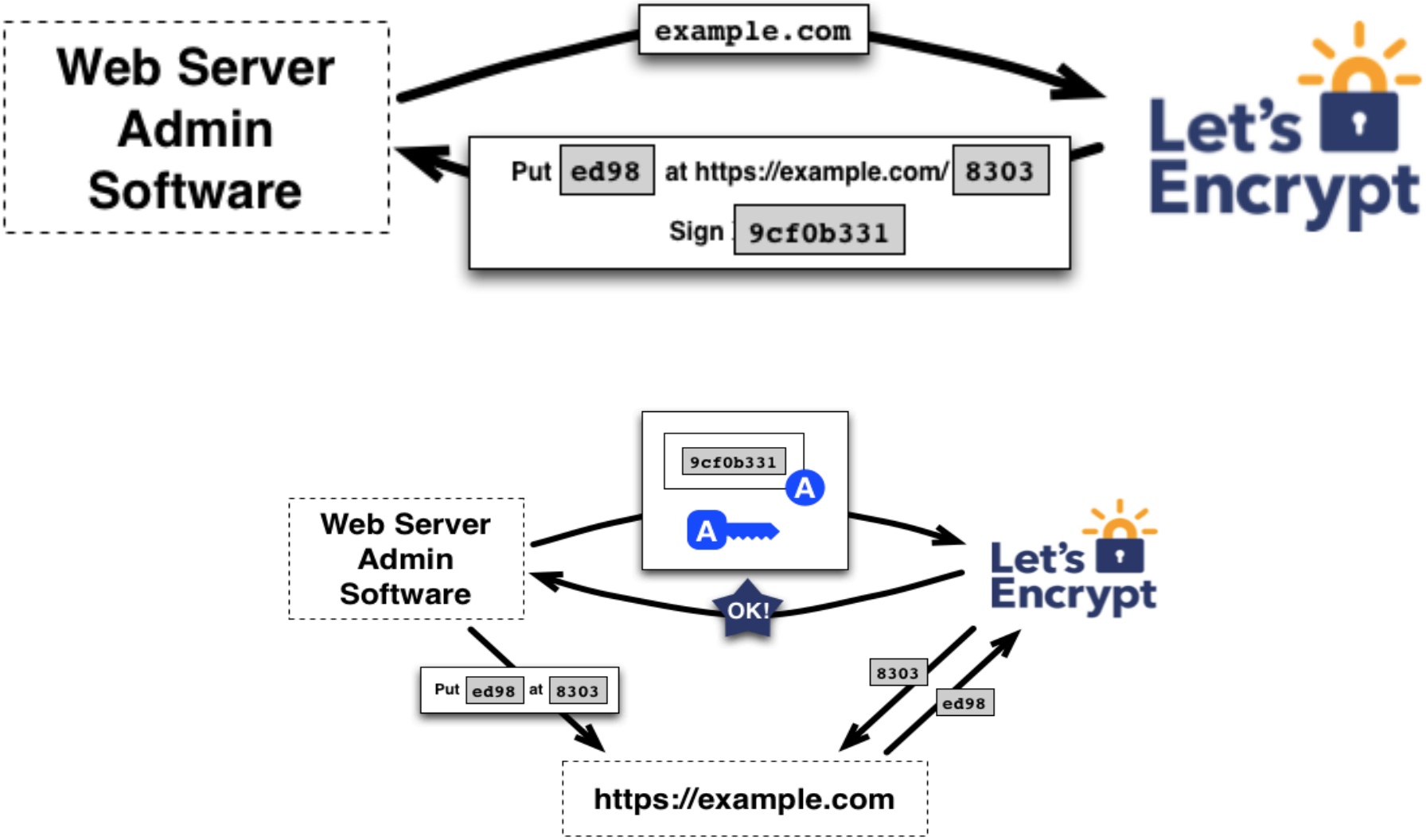
```
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

        err = sslRawVerify(ctx,
                           ctx->peerPubKey,
                           dataToSign,                          /* plaintext */
                           dataToSignLen,                  /* plaintext length */
                           signature,
                           signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
                goto fail;
        }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

The Web PKI Ecosystem

- http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf

- ~1800 CAs that can sign *any* domain controlled by 683 organizations

Free CAs


Web Server Admin Software
example.com
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt
9cf0b331
A
A
OK!
Web Server Admin Software
Let's Encrypt
Put ed98 at 8303
8303
ed98
https://example.com

Buggy Domain Validation Forces GoDaddy To Revoke SSL Certificates (threatpost.com)

33

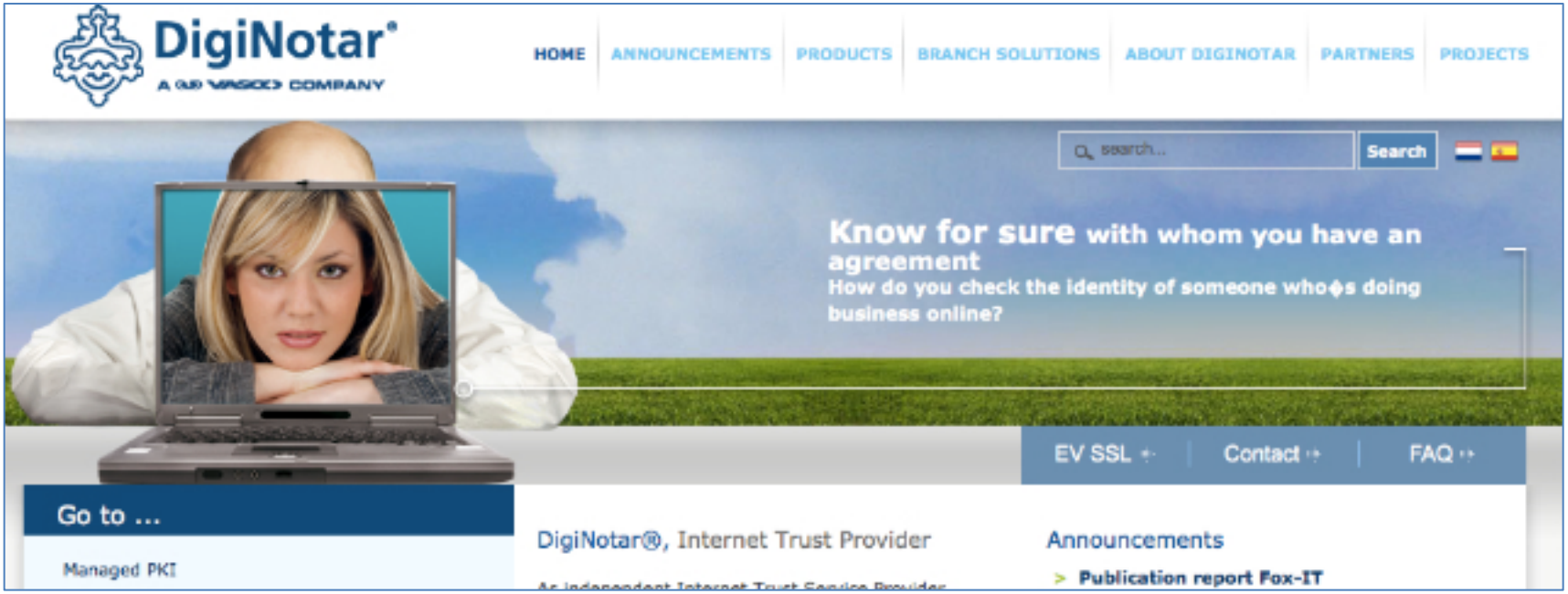
Posted by **BeauHD** on Wednesday January 11, 2017 @09:25PM from the time-to-take-action dept.

msm1267 quotes a report from Threatpost:

> GoDaddy has revoked, and begun the process of re-issuing, new SSL certificates for more than 6,000 customers after a bug was discovered in the registrar's domain validation process. The bug was introduced July 29 and impacted fewer than two percent of the certificates GoDaddy issued from that date through yesterday, said vice president and general manager of security products Wayne Thayer. "GoDaddy inadvertently introduced the bug during a routine code change intended to improve our certificate issuance process," Thayer said in a statement. "The bug caused the domain validation process to fail in certain circumstances." GoDaddy said it was not aware of any compromises related to the bug. The issue did expose sites running SSL certs from GoDaddy to spoofing where a hacker could gain access to certificates and pose as a legitimate site in order to spread malware or steal personal information such as banking credentials. GoDaddy has already submitted new certificate requests for affected customers. Customers will need to take action and log in to their accounts and initiate the certificate process in the SSL Panel, Thayer said.

Revocation

- Certificates must often be revoked
  - Short expirations
  - CRLs (Certificate revocation lists)
  - OCSP (online certificate status protocol)
    - Client queries CA to check on validity of cert
      - privacy concerns, performance / scalability issues
    - Stapling: server periodically gets fresh, time-stamped OCSP signature from CA. Sends to clients



Today, Microsoft issued a Security Advisory warning that fraudulent digital certificates were issued by the Comodo Certificate Authority. This could allow malicious spoofing of high profile websites, including Google, Yahoo! and Windows Live.

https://nakedsecurity.sophos.com/2011/03/24/fraudulent-certificates-issued-by-comodo-is-it-time-to-rethink-who-we-trust/

https://technet.microsoft.com/library/security/2524375

Certificate/public-key pinning

- Client knows what cert/pk to expect, rejects otherwise
  - Pre-install some keys
  - HPKP (HTTP Public Key Pinning)
    - HTTP header that allows servers to set a hash of public key they will use

```
Public-Key-Pins:
    pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    max-age=259200
```

https://developers.google.com/web/updates/2015/09/HPKP-reporting-with-chrome-46?hl=en

Certificate transparency

- Force CAs to log the certificates they sign in a public tamper-evident register
  - Experimental IETF standard

- Google has been pushing this
  - Chrome requires it for “extra validation” certs
  - DigiCert has implemented

Summary

- Web PKI relies on various trust assumptions
  - Can be undermined in many ways

- Digital signature schemes power PKI and verifying identities:
  - unforgeability under chosen message attack
  - RSA based schemes PKCS#1 1.5, FDH, PSS