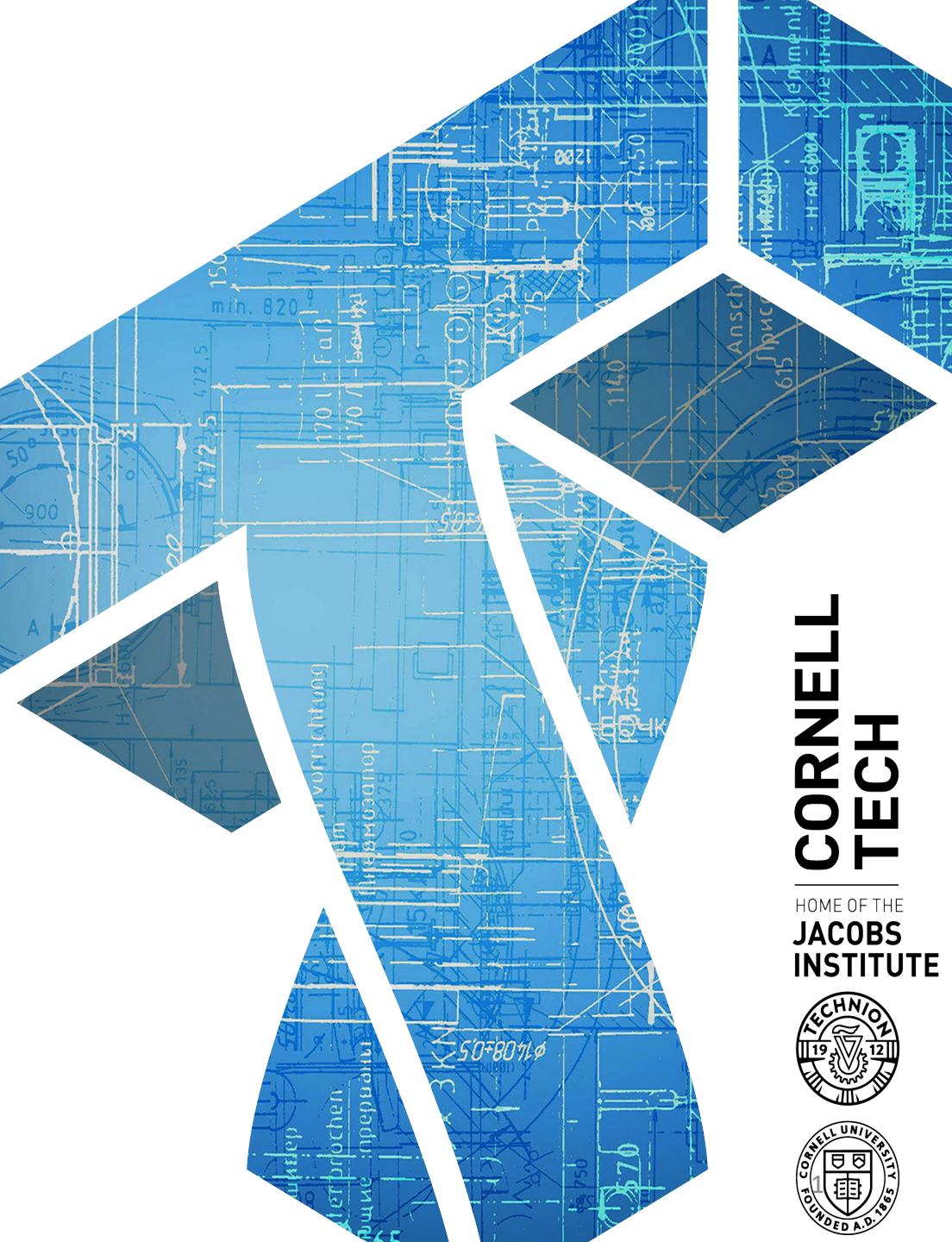


CS 5830

Cryptography

Instructor: Tom Ristenpart

TAs: Yan Ji, Sanketh Menda



**CORNELL
TECH**

HOME OF THE
**JACOBS
INSTITUTE**





Client



Server

TLS handshake for Diffie-Hellman Key Exchange

Pick random Nc

Check CERT
using CA public
verification key
Check σ

Pick random y
 $Y = g^y$

$PMS = g^{xy}$

Bracket notation
means contents
encrypted

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

$CERT = (pk_s, \text{signature over it})$

$p, g, X, \sigma = \text{Sign}(sk_s, Nc || Ns || p || g || X)$

Y

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

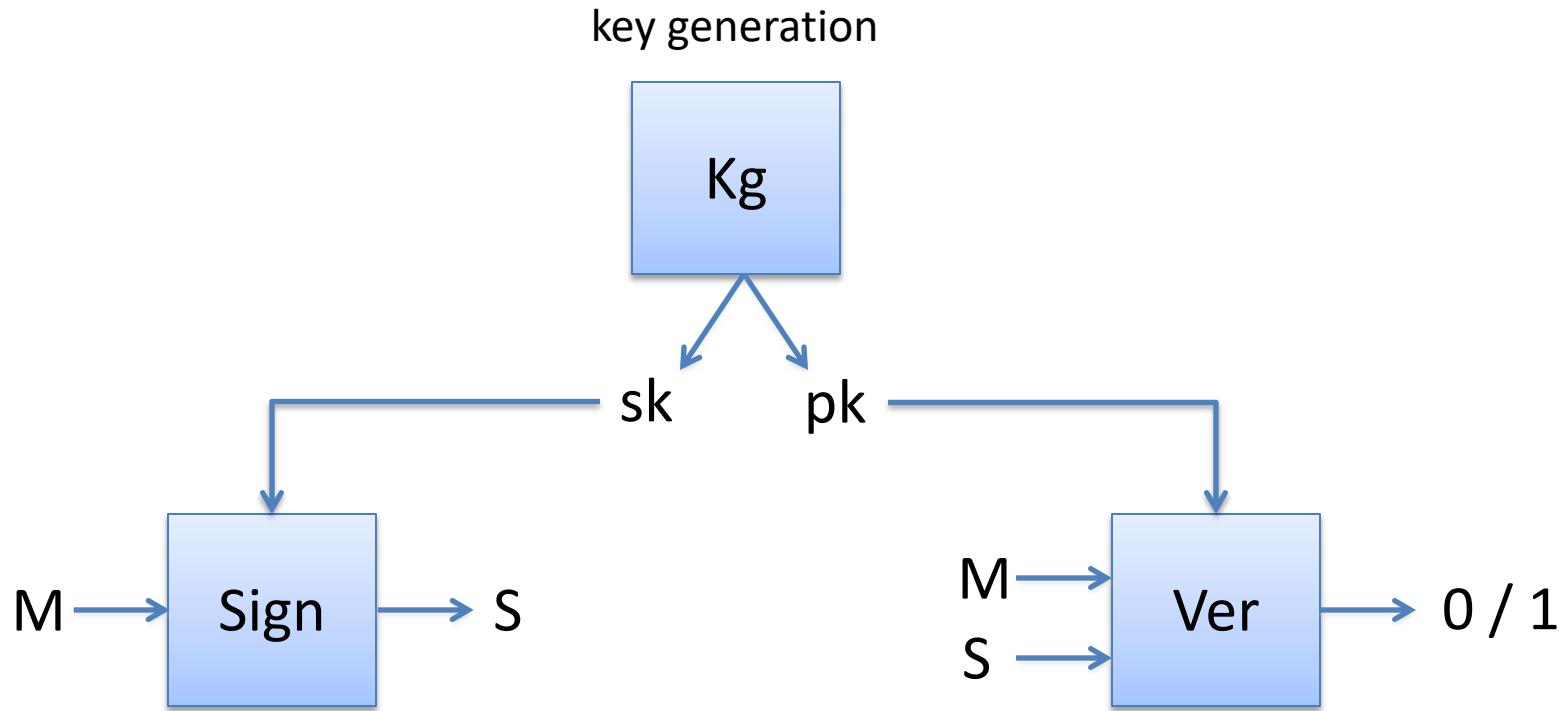
ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) }

$MS \leftarrow \text{PRF}(PMS, \text{"master secret"} || Nc || Ns)$

Pick random Ns
Pick random x
 $X = g^x$

$PMS = g^{xy}$

Digital signatures



Signing can be deterministic or randomized

Anyone with public key can verify a signature

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

Digital signatures

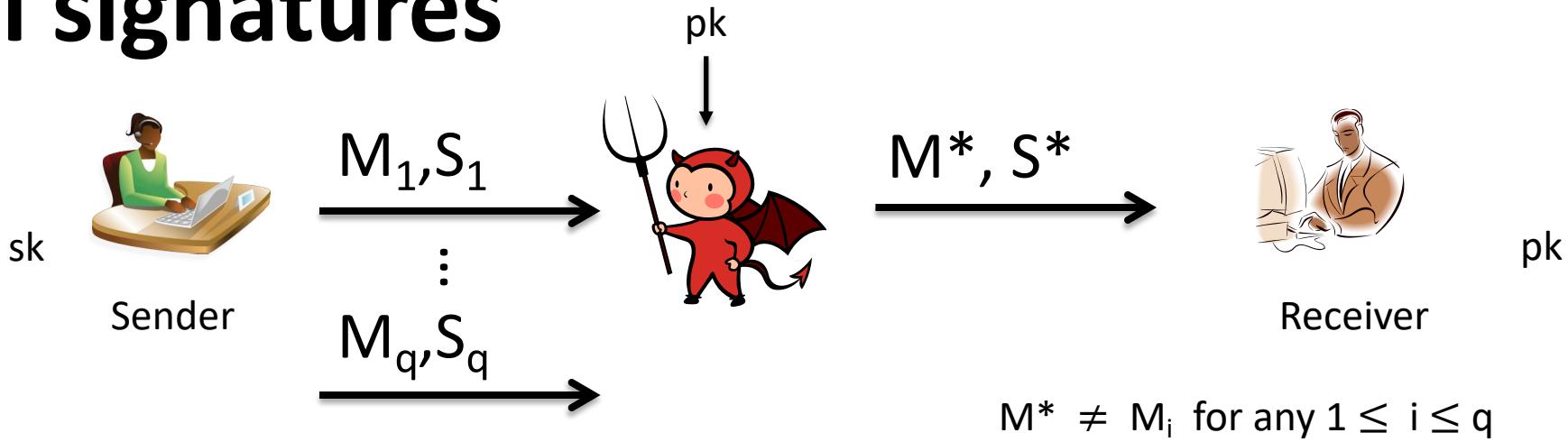


Three algorithms:

- (1) Key generation outputs (pk, sk)
- (2) $\text{Sign}(sk, M)$ outputs a signature S (may be randomized)
- (3) $\text{Ver}(pk, M, S)$ outputs 0/1 (invalid / valid)

Correctness: $\text{Ver}(pk, M, \text{Sign}(sk, M)) = 1$ always

Digital signatures



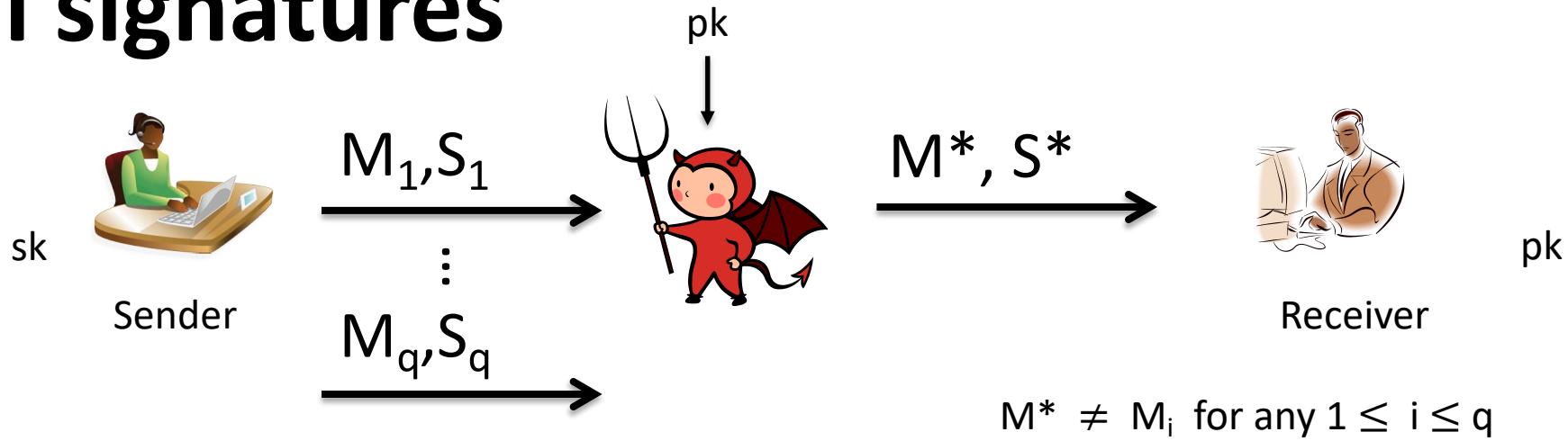
Unforgeability under chosen message attack (UF-CMA)

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

$$(M_1, S_1), (M_2, S_2), \dots, (M_q, S_q)$$

for messages of his choosing and reasonably large q .

Digital signatures



“Raw” RSA as a signature scheme:

Key generation gives (N,e) , (N,d)

$\text{Sign}((N,d),M) = M^d \bmod N$

$\text{Verify}((N,e),M,S)$ checks if $S^e \bmod N = M$

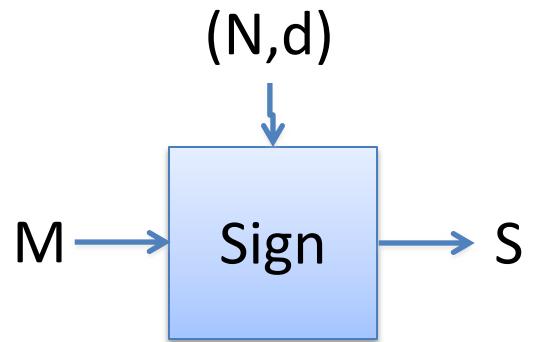
Secure? No!

PKCS #1 v1.5 RSA signing

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

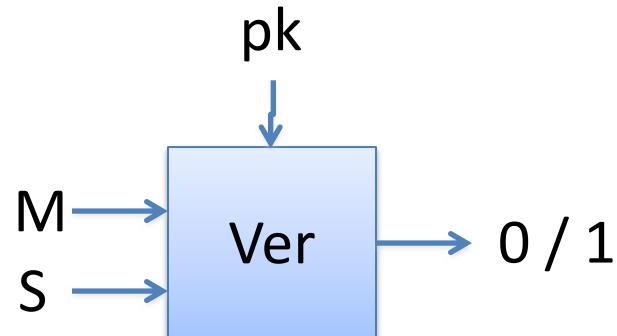
Want to sign using hash with output length m bytes

Let $p = n - m - 3$



Sign((N,d) , M)

$Y = 00 \parallel 01 \parallel FF^p \parallel 00 \parallel H(M)$
Return $Y^d \bmod N$



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

$Y = S^e \bmod N$; $aa \parallel bb \parallel cc \parallel dd \parallel h = Y$
If $(aa \neq 00)$ or $(bb \neq 01)$ or $(cc \neq FF^p)$
or $(dd \neq 00)$

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 signing oracle can be used to forge

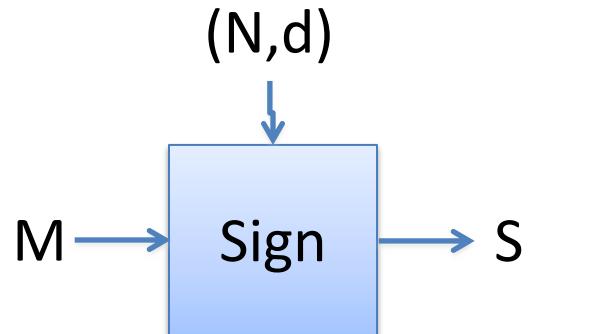
If one uses same RSA key pair for signing and encryption, then decryption and signing oracles offer similar forging / decryption opportunities

- 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 $k = \text{ceil}((n - 1)/m)$

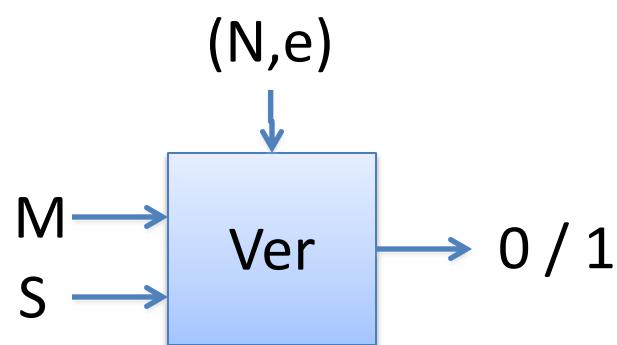


Sign((N, d) , M)

$$X = 00 || H(1||M) || \dots || H(k||M)$$

$$S = X^d \bmod N$$

Return S



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

$$X = S^e \bmod N$$

$$X' = 00 || H(1||M) || \dots || H(k||M)$$

If $X = X'$ then Return 1

Return 0

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

Groups for Schnorr and DSA Signatures

Let p be a large prime number

Let q be a prime such that q divides $p-1$

Example: $p = 2q + 1$ (so-called safe prime p)

Fix the group $G = \mathbb{Z}_p^* = \{1, 2, 3, \dots, p-1\}$

Let g be generator of sub-group of order q :

$$\{g^0, g^1, g^2, \dots, g^{q-1}\} \text{ subset of } G$$

How to pick g ?

$g = h^{(p-1)/q} \bmod p$ for some h and check $g \neq 1 \bmod p$

If so, try repeat with another h . Usually start with $h = 2$

(Variant of) Schnorr signatures

p,q,g specified

sk = x chosen randomly from \mathbb{Z}_q pk = X = g^x

Sign(x, M)

$r \leftarrow \mathbb{Z}_q$

$R = g^r$; $c = H(M || R)$; $z = r + cx \pmod q$

Return (R,z)

Ver(X, M, (R,z))

$c = H(M || R)$

If $g^z = RX^c$ then Return 1

Return 0

Correctness?

$$g^z = g^{r+cx} = g^r g^{xc} = RX^c$$

Security intuition

Assume an adversary that can output forgery $(M, (R, z))$

Then to be valid:

$$g^z = RX^c \text{ implies } z = r + cx$$

for $c = H(M \parallel R)$.

Assume c is random (H is random oracle)



Imagine we can run adversary twice but force forgery to be on same R , different c .

In second execution, getting $(M', (R, z'))$

Then success second time around gives:

$$g^{z'} = RX^{c'} \text{ implies } z' = r + c'x$$

But now can compute $z - z' / (c - c') = x$ the secret key

Fragility of signatures

Repeat randomness failure:

Sign two messages $M \neq M'$ and reuse randomness

$$\text{Sign}(x, M) \rightarrow (R, z) = (R, r + cx \bmod q)$$

$$\text{Sign}(x, M') \rightarrow (R, z') = (R, r + c'x \bmod q)$$

$$\text{Then: } x = (z - z') / (\mathcal{H}(M || R) - \mathcal{H}(M' || R))$$

If r is predictable/leaked, can recover secret from (R, z)

Actual Schnorr signatures

p, q, g specified

$sk = x$ chosen randomly from \mathbb{Z}_q $pk = X = g^x$

Sign(x, M)

$r \leftarrow \mathbb{Z}_q$

$R = g^r ; c = H(M || R) ; z = r + cx \pmod{q}$

Return (c, z)

Ver($X, M, (c, z)$)

$R' = g^z X^{-c}$

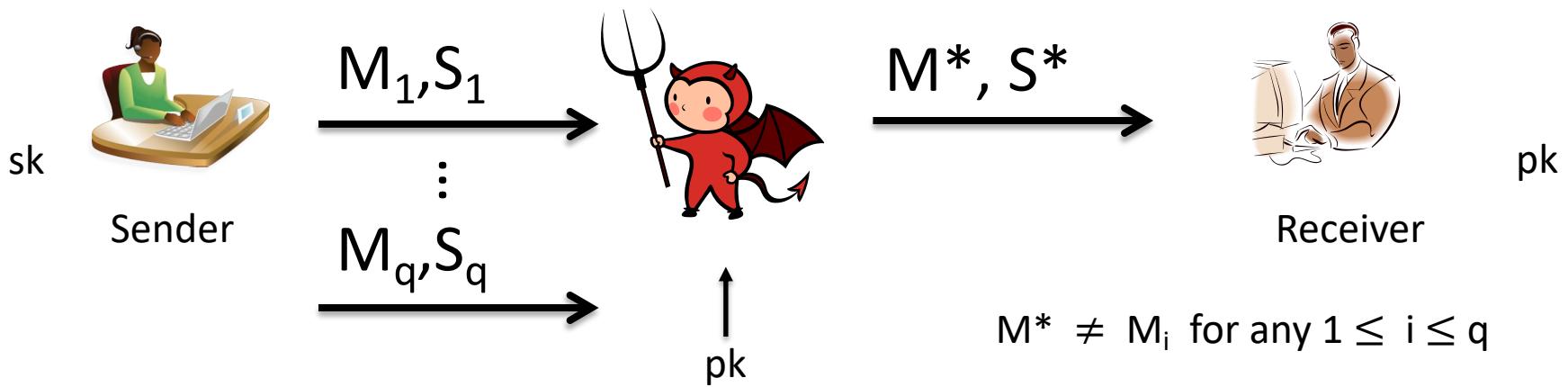
$c' = H(M || R')$

If $c' = c$ then Return 1

Return 0

Correctness? $R' = g^z X^{-c} = g^{r + cx} g^{x/(H(M||R))} = g^r$

Formalizing unforgeability



UF-CMA for digital signatures

UF-CMA(MA, \mathcal{A}):

$(pk, sk) \leftarrow \mathcal{K}_G$
 $(M^*, S^*) \leftarrow \mathcal{A}^{Sign}(pk)$

If $M^* \in \text{MsgSet}$ then

Return 0

Return $\text{Verify}(pk, M^*, S^*)$

Sign(M)

$\text{MsgSet} \leftarrow \text{MsgSet} \cup \{M\}$
Return $\text{Sign}(sk, M)$

UF-CMA: Symmetric vs asymmetric settings

UF-CMA(MA, \mathcal{A}):

$K \leftarrow \$ Kg$
 $(M^*, T^*) \leftarrow \$ \mathcal{A}^{Tag}$
If $M^* \in \text{MsgSet}$ then
 Return 0
Return Verify(K, M^*, T^*)

Tag(M)

$\text{MsgSet} \leftarrow \text{MsgSet} \cup \{M\}$
Return Tag(K, M)

UF-CMA(DS, \mathcal{A}):

$(pk, sk) \leftarrow \$ Kg$
 $(M^*, S^*) \leftarrow \$ \mathcal{A}^{Sign}(pk)$
If $M^* \in \text{MsgSet}$ then
 Return 0
Return Verify(pk, M^*, S^*)

Sign(M)

$\text{MsgSet} \leftarrow \text{MsgSet} \cup \{M\}$
Return Sign(sk, M)

Unforgeability under chosen message attack for message authentication schemes

Unforgeability under chosen message attack for digital signature schemes

Schnorr formal security analysis

- UF-CMA for digital signatures
- Requires modeling H as a *random oracle*
 - Public random function all parties have access to
 - Very successful modeling approach
- Fiat-Shamir analysis : security if DL is hard
 - Fiat-Shamir transform converts any of a class of zero-knowledge protocols to a digital signature
 - Security of protocol shown to imply UF-CMA in random oracle model
- Time allowing we will do a lecture on zero-knowledge proofs

UF-CMA(MA, \mathcal{A}):
 $(pk, sk) \leftarrow \mathcal{K}_G$
 $(M^*, S^*) \leftarrow \mathcal{A}^{\text{Sign}}(pk)$
If $M^* \in \text{MsgSet}$ then
 Return 0
Return $\text{Verify}(pk, M^*, S^*)$

Sign(M)
 $\text{MsgSet} \leftarrow \text{MsgSet} \cup \{M\}$
Return $\text{Sign}(sk, M)$

Formal security analysis

- Schnorr formally shown to be UF-CMA assuming discrete log problem is hard
- Schnorr signatures highly vulnerable if per-message randomness known or repeated
- Why are these not contradictory?
- “Hedged” cryptography (e.g., [\[Ristenpart, Yilek NDSS 2010\]](#)
 - Formal models that capture randomness failures
 - New schemes that “hedge” against bad randomness: $r = H(x \mid\mid M \mid\mid \text{randomness})$
 - Full deterministic versions: <https://tools.ietf.org/id/draft-pornin-deterministic-dsa-00.xml>

UF-CMA(MA, \mathcal{A}):
 $(pk, sk) \leftarrow \mathcal{K}_G$
 $(M^*, S^*) \leftarrow \mathcal{A}^{\text{Sign}}(pk)$
If $M^* \in \text{MsgSet}$ then
 Return 0
Return $\text{Verify}(pk, M^*, S^*)$

Sign(M)
 $\text{MsgSet} \leftarrow \text{MsgSet} \cup \{M\}$
Return $\text{Sign}(sk, M)$

DSA (digital signature algorithm)

p,q,g specified

sk = x chosen randomly from \mathbb{Z}_q

pk = X = g^x

Sign(x, M)

$r \leftarrow \mathbb{Z}_q ; R = (g^r \text{ mod } p) \text{ mod } q$

$z = r^{-1} (H(M) + x R) \text{ mod } q$

Return (R,z)

Ver(X, M, (R,z))

$w = z^{-1} \text{ mod } q$

$u_1 = H(m) * w \text{ mod } q$

$u_2 = R * w \text{ mod } q$

If $R = (g^{u_1} X^{u_2} \text{ mod } p) \text{ mod } q$

then Return 1

Else Return 0

Correctness?

$$\begin{aligned} g^{u_1} X^{u_2} &= g^{H(M)w} g^{xRw} = g^{(H(M)+xR)w} \\ &= g^{(H(M)+xR)(H(M)+xR)^{-1}r} = g^r \end{aligned}$$

Fragility of DSA

Repeat randomness failure:

Sign two messages $M \neq M'$ and reuse random

$$\begin{aligned} \text{Sign}(x, M) \rightarrow (R, z) &= (R, r^{-1} (H(M) + x R) \bmod q) \\ \text{Sign}(x, M') \rightarrow (R, z') &= (R, r^{-1} (H(M') + x R) \bmod q) \end{aligned}$$

Then: Solve for r^{-1} , solve for x

If r is predictable/leaked, can recover secret from (R, z)

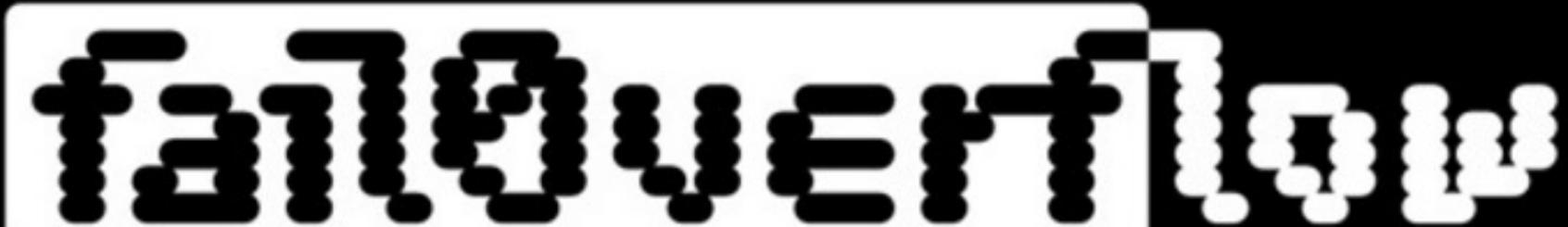
Again, can improve security by “hedging”:

$$\text{choose } r = H(x \parallel M \parallel \text{randomness})$$

Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access

BY MIKE BENDEL

DECEMBER 29, 2010 @ 11:19 AM



<http://psx-scene.com/forums/content/sony-s-ps3-security-epic-fail-videos-within-581/>

Digital signature schemes

	Problems?	Proofs?	Uses
RSA PKCS#1 v1.5	Bleichenbacher attacks		TLS, Certificates, XML
RSA PSS (PKCS#1 v2)		Security reduces to hardness of inverting RSA	
Schnorr	Randomness fragility	Security reduces to discrete log problem	
DSA	Randomness fragility	No formal security reduction	Bitcoin (ECC version), TLS, SSH, elsewhere

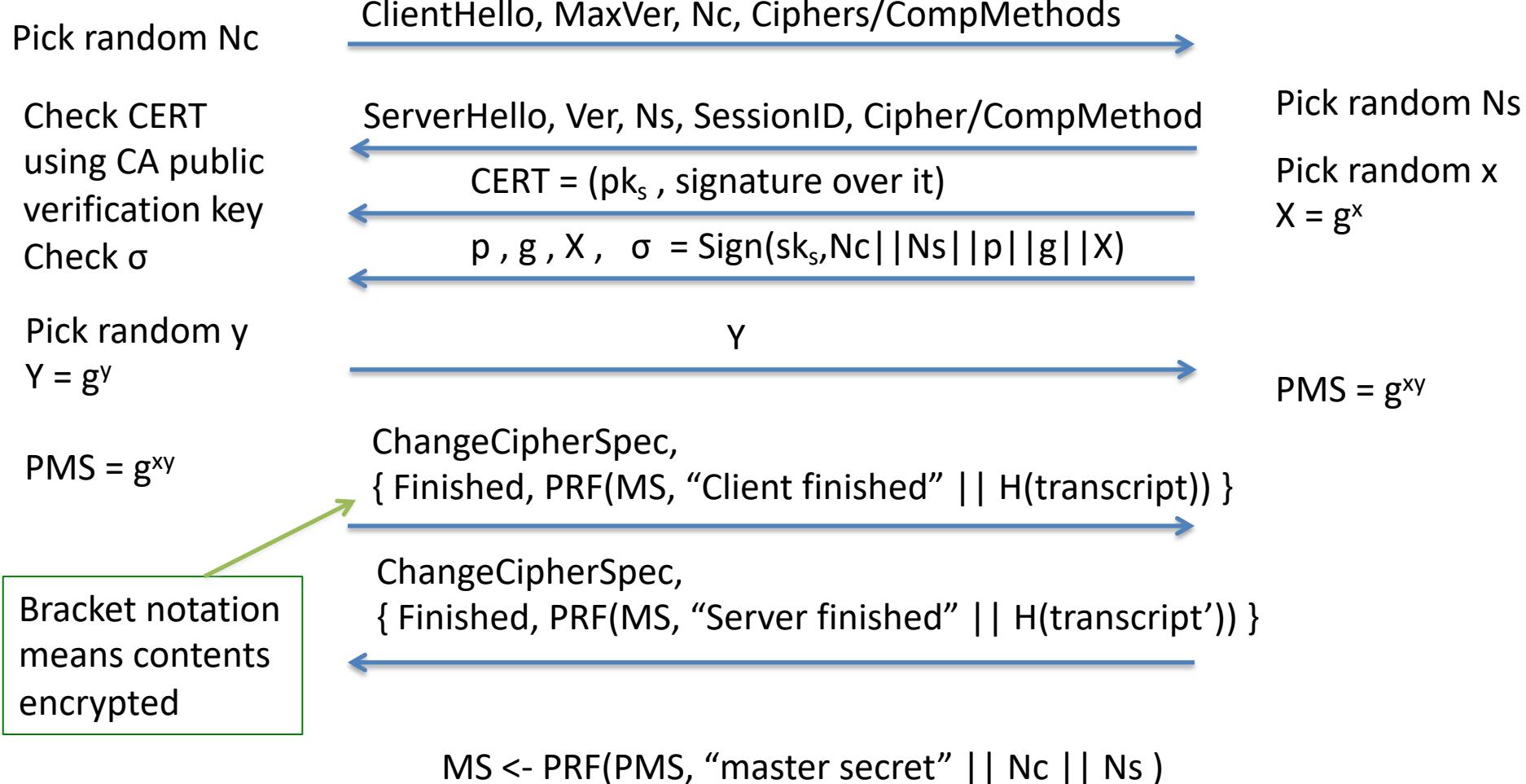


Client



Server

TLS handshake for Diffie-Hellman Key Exchange



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;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

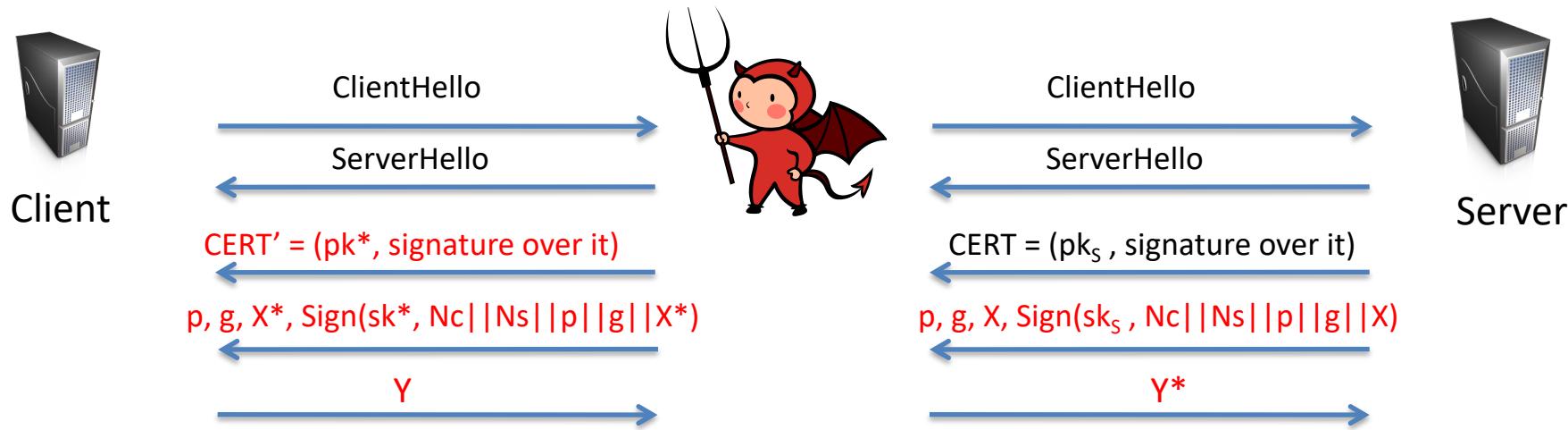
err = sslRawVerify(ctx,
                    ctx->peerPubKey,
                    dataToSign,                                /* plaintext */
                    dataToSignLen,                             /* plaintext length */
                    signature,
                    signatureLen);

if(err) {
    sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                "returned %d\n", (int)err);
    goto fail;
}

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

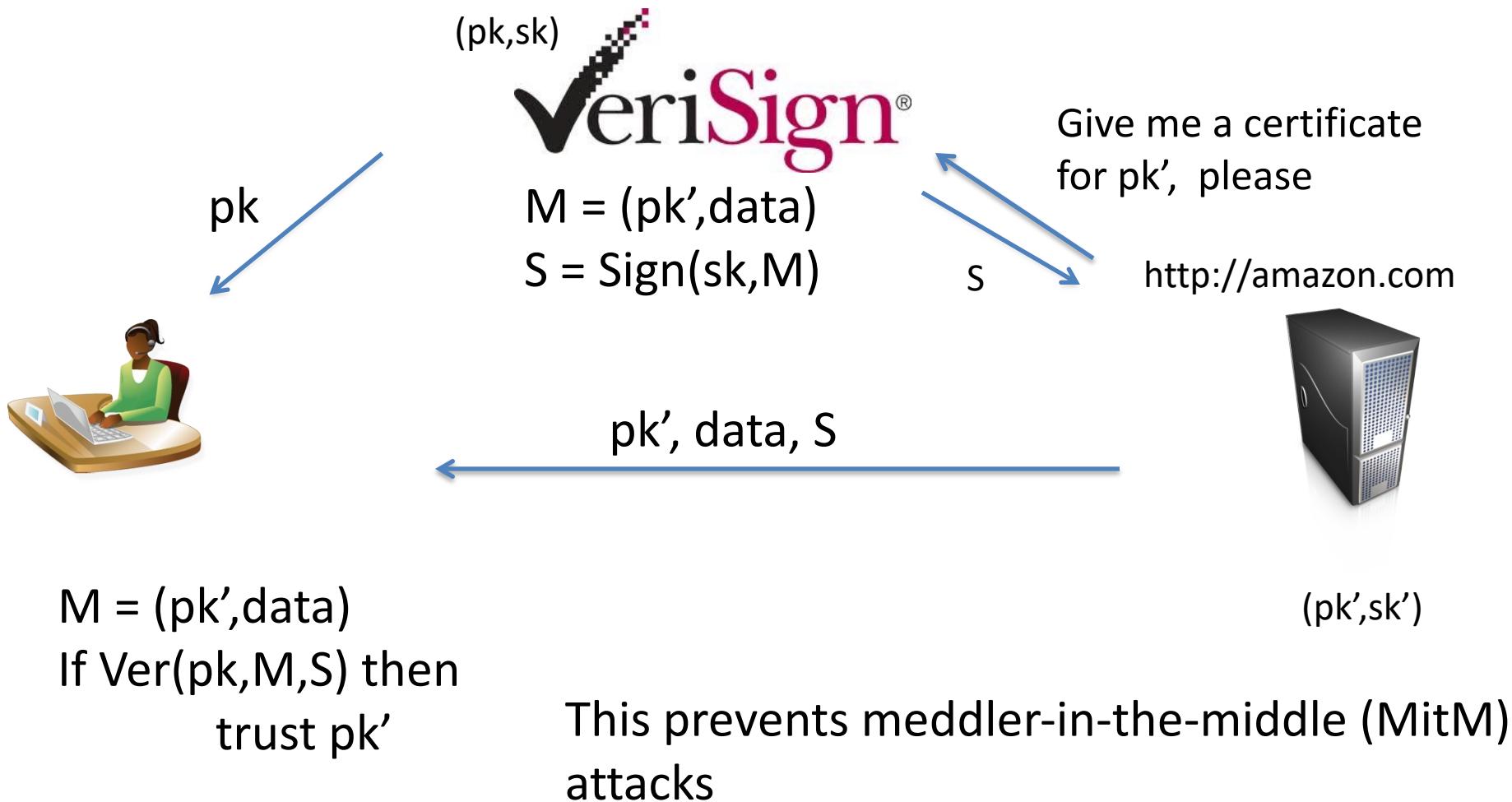
Meddler-in-the-middle attacks

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

Certificate Authorities and Public-key Infrastructure (PKI)



Certificate:

Data:

 Version: 1 (0x0)
 Serial Number: 7829 (0x1e95)
 Signature Algorithm: md5WithRSAEncryption
 Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
 OU=Certification Services Division,
 CN=Thawte Server CA/emailAddress=server-certs@thawte.com

Validity

 Not Before: Jul 9 16:04:02 1998 GMT
 Not After : Jul 9 16:04:02 1999 GMT

Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
 OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org

Subject Public Key Info:

 Public Key Algorithm: rsaEncryption
 RSA Public Key: (1024 bit)

 Modulus (1024 bit):

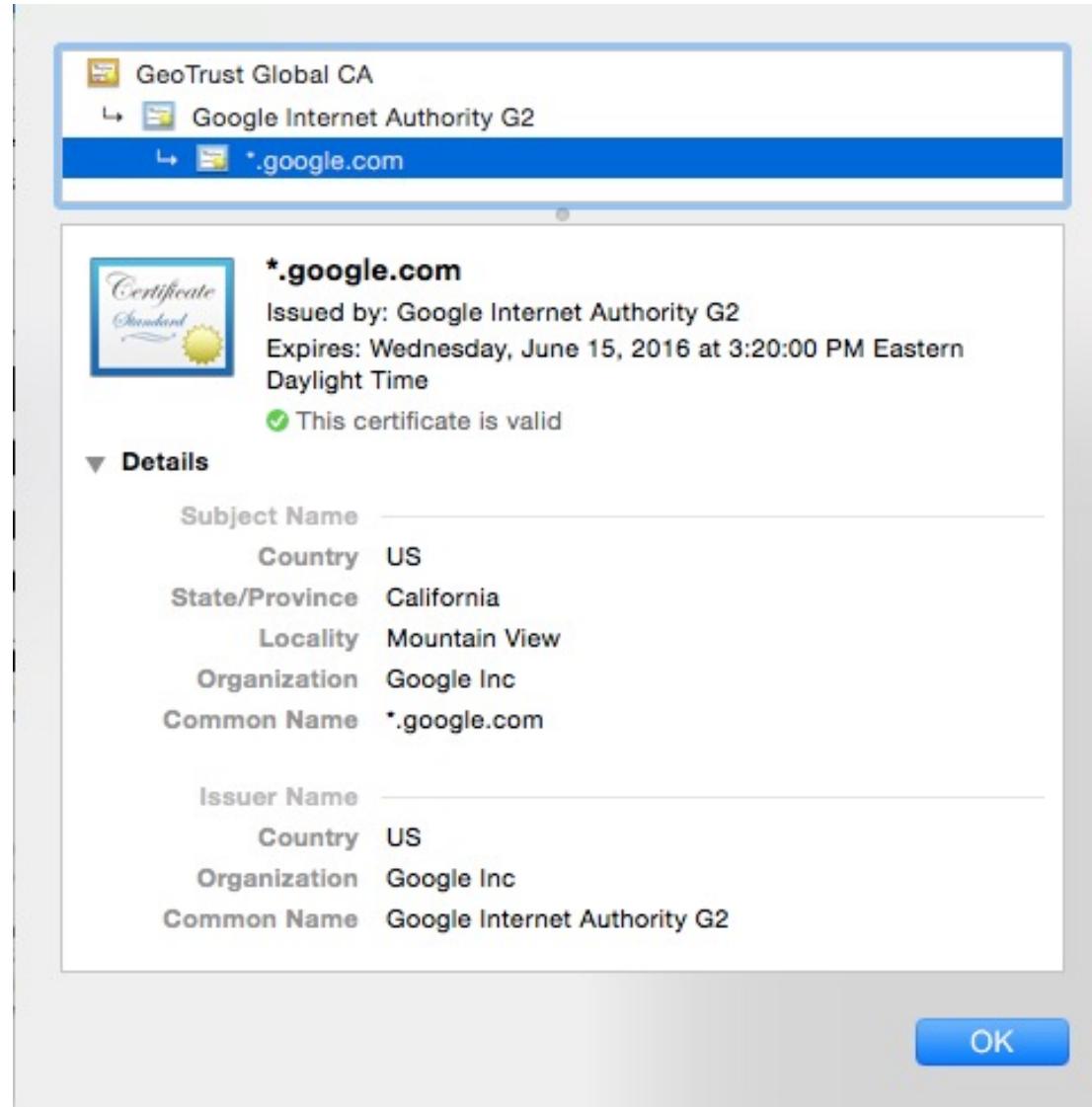
 00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
 33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
 66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
 70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
 16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
 c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
 8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
 d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
 e8:35:1c:9e:27:52:7e:41:8f

 Exponent: 65537 (0x10001)

 Signature Algorithm: md5WithRSAEncryption

 93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
 92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
 ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
 d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
 0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
 5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
 8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
 68:9f

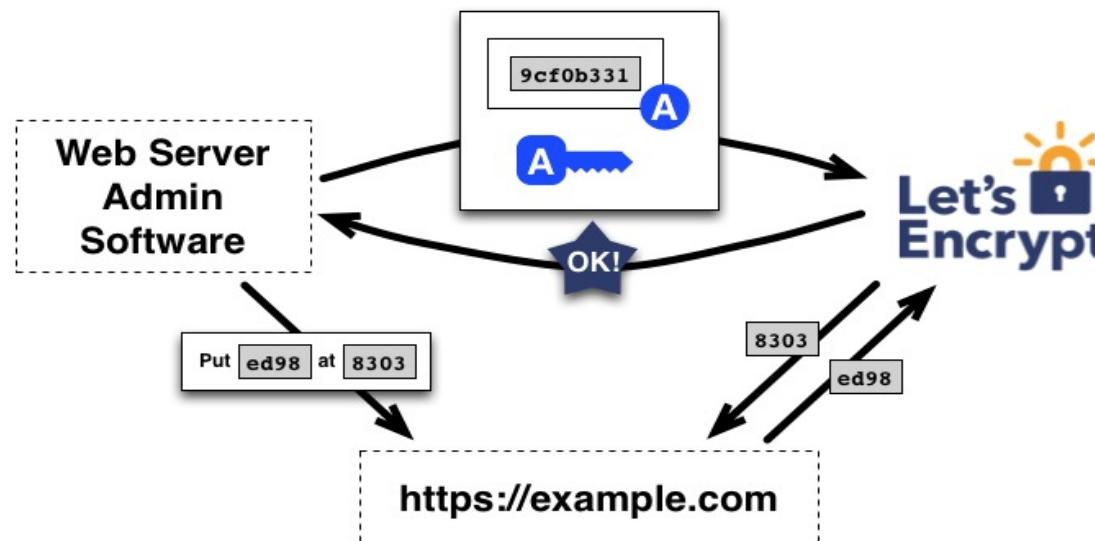
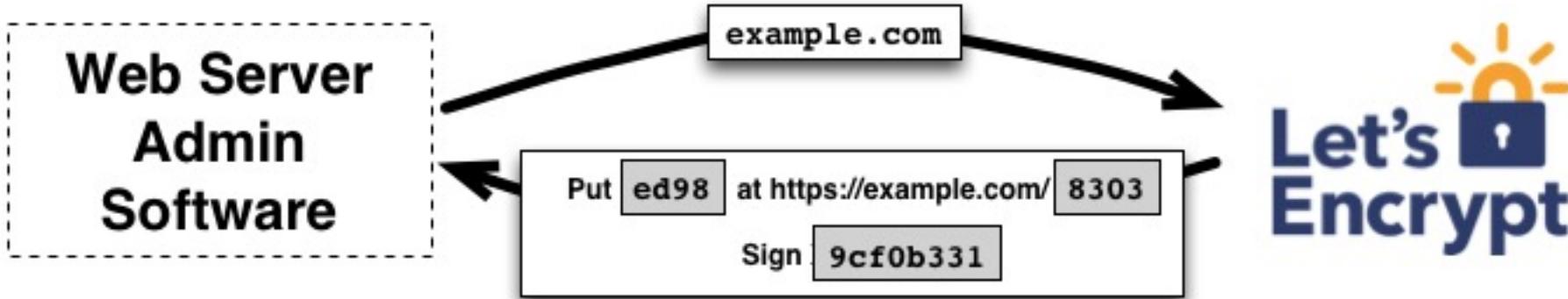
Certificate chains



Identity checks?

- CA's must check that requestor of cert is who they say they are
- Domain validated
 - Prove ownership of domain
- Extended validation
 - Establish legal identity of requestor
 - Physical presence of website owner
 - Confirm ownership of domain
 - Etc.

Free CAs



Revocation

- Certificates must often be revoked, big pain point in practice
 - 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

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
- DigiNotar compromise, GoDaddy bug, Turktrust incident, ...

Buggy Domain Validation Forces GoDaddy To Revoke SSL Certificates



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.



search...

Search



Know for sure with whom you have an agreement
How do you check the identity of someone who's doing business online?

EV SSL + | Contact + | FAQ +

Go to ...

DigiNotar®, Internet Trust Provider
An independent Internet Trust Certificate Provider

Announcements > Publication report Fox-IT

Managed PKI

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>

Measurement studies of TLS ecosystem

- Use internet-wide scanning to measure ecosystem
 - Fewer than 2^{32} IP addresses: ~3,706,452,992
 - Exclude private addresses (e.g., 192.168.0.0/16), multicast addresses, ...
- Zmap is a state-of-art tool: can scan one port of entire Internet in ~45 minutes
- Extend to perform TLS handshakes with servers that are open on port 443

Some scans

- 2011 scan (Heninger et al.)
 - 12,828,613 certificate chains recovered
 - 5,656,519 distinct RSA public keys
 - 6,241 distinct DSA public keys
 - Found many *factorable* RSA keys, DSA keys signing with repeat randomness
- 2013 scan (Durumeric et al.)
 - Almost all CA's using RSA keys for signing certs
 - ~8 million unique certificates across ~30 million hosts

2013 scan: breakdown by CA

Parent Company	Signed Leaf Certificates	
Symantec	1,184,723	(34.23%)
GoDaddy.com	1,008,226	(29.13%)
Comodo	422,066	(12.19%)
GlobalSign	170, 006	(4.90%)
DigiCert Inc	145,232	(4.19%)
StartCom Ltd.	88,729	(2.56%)
Entrust, Inc.	76,990	(2.22%)
Network Solutions	62,667	(1.81%)
TERENA	42,310	(1.22%)
Verizon Business	32,127	(0.92%)

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