University of Pittsburgh

Applied Cryptography and Network Security
CS 1653

Summer 2023

Sherif Khattab
ksm73@pitt.edu

(Slides are adapted from Prof. Adam Lee's CS1653 slides.)

Announcements

- Homework 8 due this Friday @ 11:59 pm
- Project Phase 3 Due tonight @ 11:59 pm
- Homework 9 due next Friday @ 11:59 pm
- Project Phase 4 Due on 7/31 @ 11:59 pm
  - Teams must meet with me on or before Thursday 7/27

TLS Protocol summary

At a high level, this protocol proceeds in four phases

- Setup and parameter negotiation
- Key exchange/derivation
- Authentication
- Data transmission

For security reasons, both parties participate in (almost) all phases

- Setup: Client proposes, server chooses
- Key derivation: Randomness contributed by both parties
- Authentication: Usually, just the server is authenticated (Why?)
- Data transmission: Both parties can encrypt and integrity check

The low-level details are not much more complicated than that!

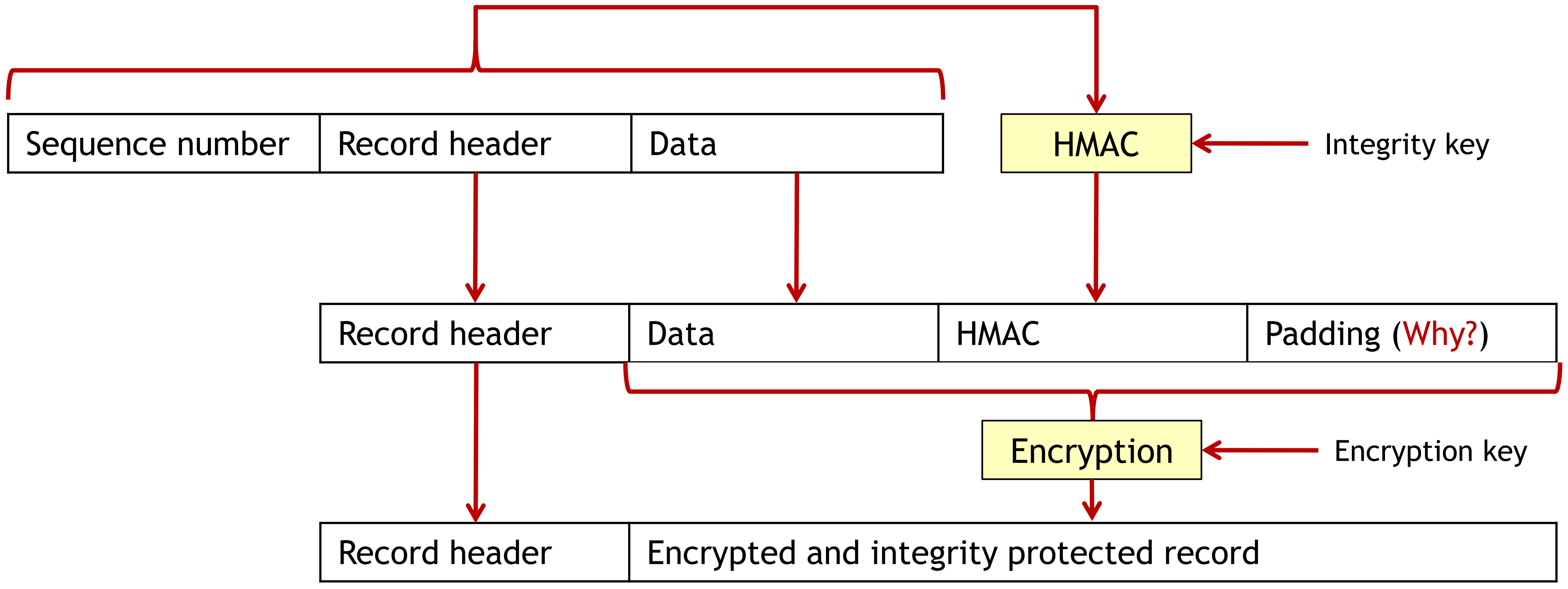
All records sent after ChangeCipherSpec are encrypted and integrity protected

All of this protection is afforded by the algorithms identified in the cipher suite chosen by the server

Example: `TLS_RSA_WITH_AES_128_CBC_SHA`

- Encryption provided using 128-bit AES in CBC mode
- Integrity protection provided by HMAC-SHA-1

Note: Data is protected one record at a time



This handshake procedure is fairly heavyweight

Public key cryptography is used by both parties

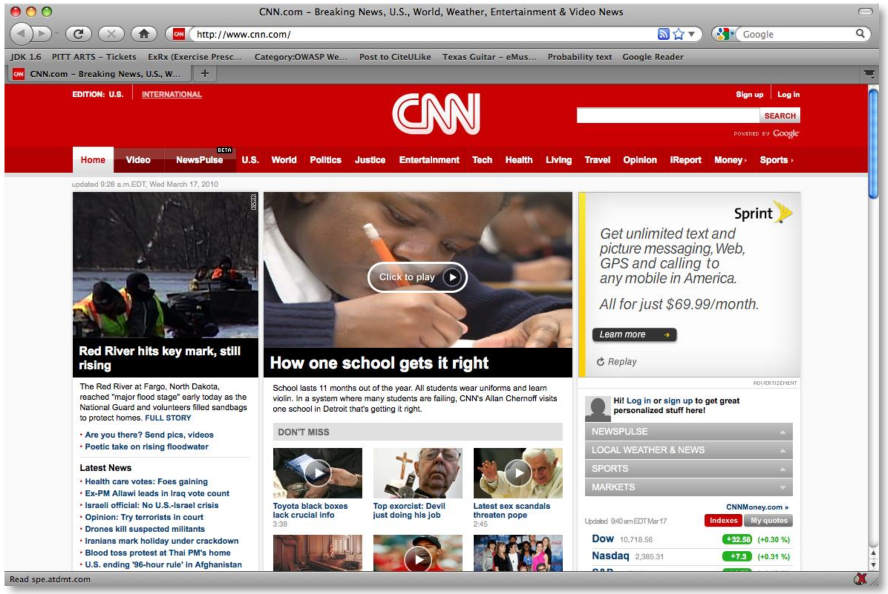
- Alice encrypts her pre-master secret using the server's public key
- The server decrypts this pre-master secret

So what! Aren't connections long lived?

Example: Visiting http://www.cnn.com

Visiting this single web page triggers over 130 separate HTTP connections!

This is less than optimal.
Can we do better?



CNN.com – Breaking News, U.S., World, Weather, Entertainment & Video News

http://www.cnn.com/

JDK 1.6 · PITT ARTS – Tickets · ExRx (Exercise Presc... · Category:OWASP We... · Post to CiteULike · Texas Guitar – eMus... · Probability text · Google Reader

CNN.com – Breaking News, U.S., W...

updated 9:28 a.m.EDT, Wed March 17, 2010

Red River hits key mark, still rising

The Red River at Fargo, North Dakota, reached "major flood stage" early today as the National Guard and volunteers filled sandbags to protect homes. FULL STORY

- Are you there? Send pics, videos
- Poetic take on rising floodwater

How one school gets it right

School lasts 11 months out of the year. All students wear uniforms and learn violin. In a system where many students are failing, CNN's Allan Chernoff visits one school in Detroit that's getting it right.

Get unlimited text and picture messaging, Web, GPS and calling to any mobile in America.

All for just $69.99/month.

Hi! Log in or sign up to get great personalized stuff here!

Latest News

- Health care votes: Foes gaining
- Ex-PM Allawi leads in Iraq vote count
- Israeli official: No U.S.-Israel crisis
- Opinion: Try terrorists in court
- Drones kill suspected militants
- Iranians mark holiday under crackdown
- Blood toss protest at Thai PM's home
- U.S. ending '96-hour rule' in Afghanistan

DON'T MISS

Toyota black boxes lack crucial info 3:38

Top exorcist: Devil just doing his job

Latest sex scandals threaten pope 2:45

CNNMoney.com »

Updated 9:40 am EDT Mar 17

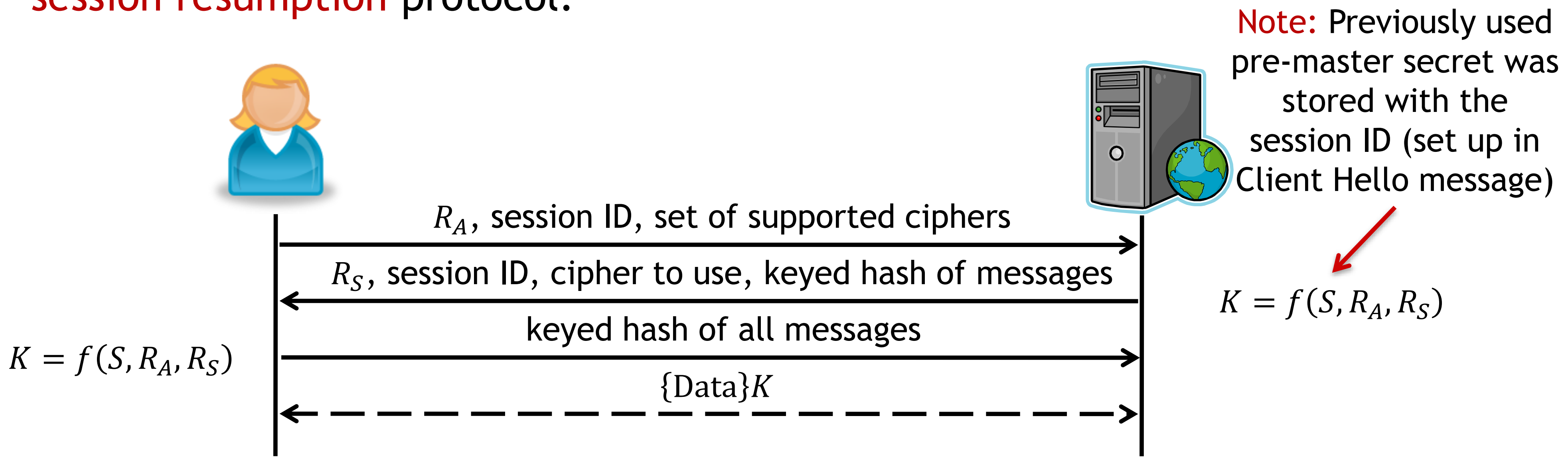
Dow	10,718.56	+32.58	(+0.30 %)
Nasdaq	2,385.31	+7.3	(+0.31 %)

Read spe.atdmt.com

Sessions vs. Connections

In TLS, sessions are assumed to be long-lived entities that may involve many smaller connections

Connections can be spawned from an existing session using a streamlined session resumption protocol:



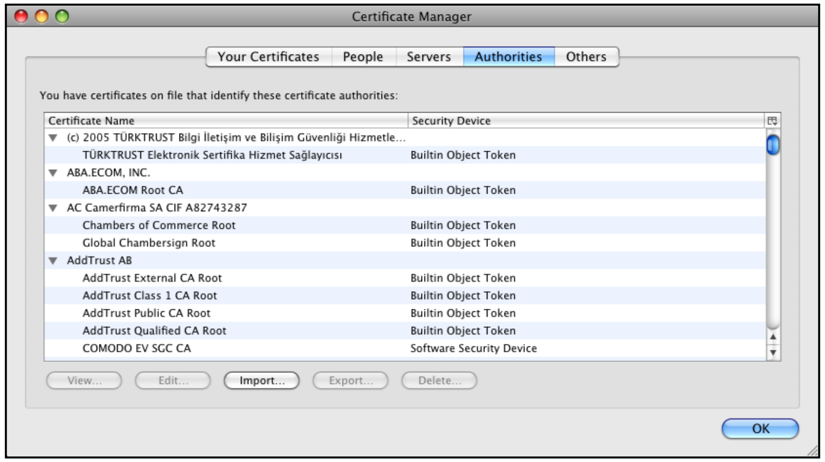
Note: Previously used pre-master secret was stored with the session ID (set up in Client Hello message)
R_A, session ID, set of supported ciphers
R_S, session ID, cipher to use, keyed hash of messages
keyed hash of all messages
$\{\text{Data}\}K$
$K = f(S, R_A, R_S)$
$K = f(S, R_A, R_S)$

In this model, a single session could be set up with CNN and connections can be spawned as needed to retrieve content

Using HTTP/2, this concept can be taken even farther: server can respond with data for queries before they're even requested

Servers authenticate themselves using certificates. How do we authenticate the certificates?

Most TLS deployments use an oligarchy PKI model



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
▼ (c) 2005 TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetle...	
TÜRKTRUST Elektronik Sertifika Hizmet Sağlayıcısı	Builtin Object Token
▼ ABA.ECOM, INC.	
ABA.ECOM Root CA	Builtin Object Token
▼ AC Camerfirma SA CIF A82743287	
Chambers of Commerce Root	Builtin Object Token
Global Chambersign Root	Builtin Object Token
▼ AddTrust AB	
AddTrust External CA Root	Builtin Object Token
AddTrust Class 1 CA Root	Builtin Object Token
AddTrust Public CA Root	Builtin Object Token
AddTrust Qualified CA Root	Builtin Object Token
COMODO EV SGC CA	Software Security Device

That is, as long as the server presents a certificate chain that uses one of our trusted roots, we're happy

What about naming?

- Servers are usually known by their DNS names
- X.509 is not set up for DNS naming
- Usually the CN field of the X.509 certificate contains the DNS name

Example: C = US, ST = Washington, L = Seattle, O = Amazon.com Inc.,
CN = www.amazon.com ← Why is this safe?

Summary of TLS

Although TCP provides a reliable data transfer protocol, it is not secure

- TCP can recover from bit-flips and dropped packets
- But malicious adversaries can alter data undetected

TLS provides cryptographic confidentiality and integrity protection for data sent over TCP

The security afforded by TLS is defined by using cipher suites

- Developers to easily incorporate new algorithms
- Security professionals can tune the level of security offered
- Breaking a cipher does not break the protocol

Breaking Crypto!

- Breaking PKI by breaking collision resistance
- Brute force attacks without using brute force
    - SSL key generation
    - Kerberos v4
- Brute force attacks against symmetric key ciphers
    - massively parallel attack
    - Deep Crack
- Subverting public key cryptography protocols
- Attacking real world implementations
    - Timing analysis of RSA
    - Power analysis of DES
    - Keyjacking cryptographic APIs

What is a hash function?

Recall: A hash function is a function that maps a variable-length input to a fixed-length code

Hash functions should possess the following 3 properties:

- Preimage resistance: Given a hash output value z, it should be infeasible to calculate a message x such that $H(x) = z$
- Second preimage resistance: Given a message x, it is infeasible to calculate a second message y such that $H(x) = H(y)$
- Collision resistance: It is infeasible to find two messages x and y such that $H(x) = H(y)$

Question: Why are cryptographic hash functions important to PKIs?

Digital signature operations are expensive to compute! Instead of signing a certificate c, we actually sign $H(c)$.

What happens if we don’t have these properties?

Attack 1: No preimage resistance

- One attack is being able to recover a password from $H(\text{password})$
- Not critical to the security of PKIs, but would cause issues in general

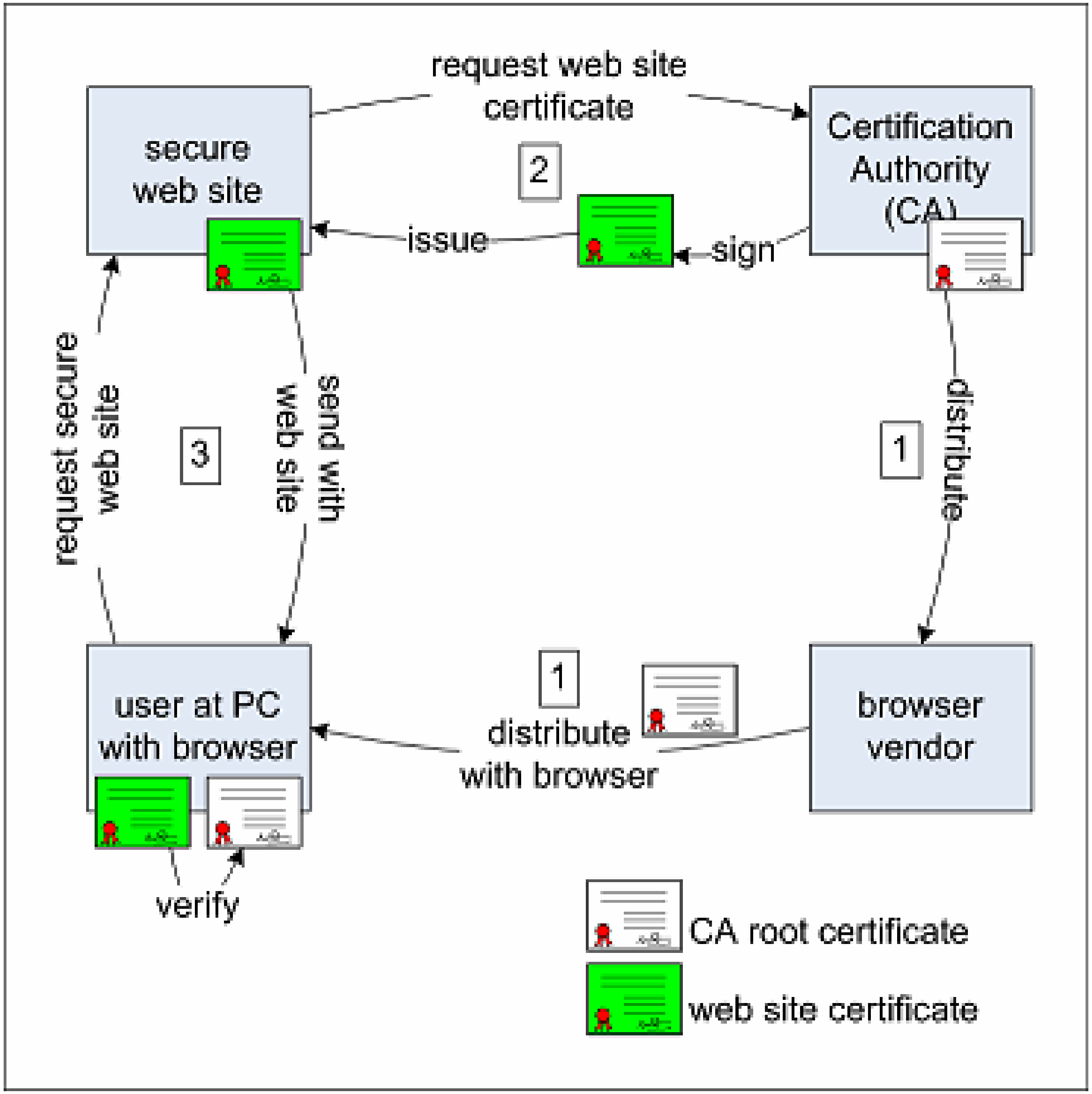
Attack 2: No second preimage resistance

- Assume that we have a message m_1 with a signature computed over $H(m_1)$
- If we don’t have second preimage resistance, we can find a message m_2 such that $H(m_1) = H(m_2)$
- The result: It looks like the signer signed m_2!

Attack 3: No collision resistance

- This means that we can find two messages that have the same hash
- The authors use a clever variant of this type of attack to **construct two certificates that have the same MD5 hash**
- This is bad news…

How TLS should work…

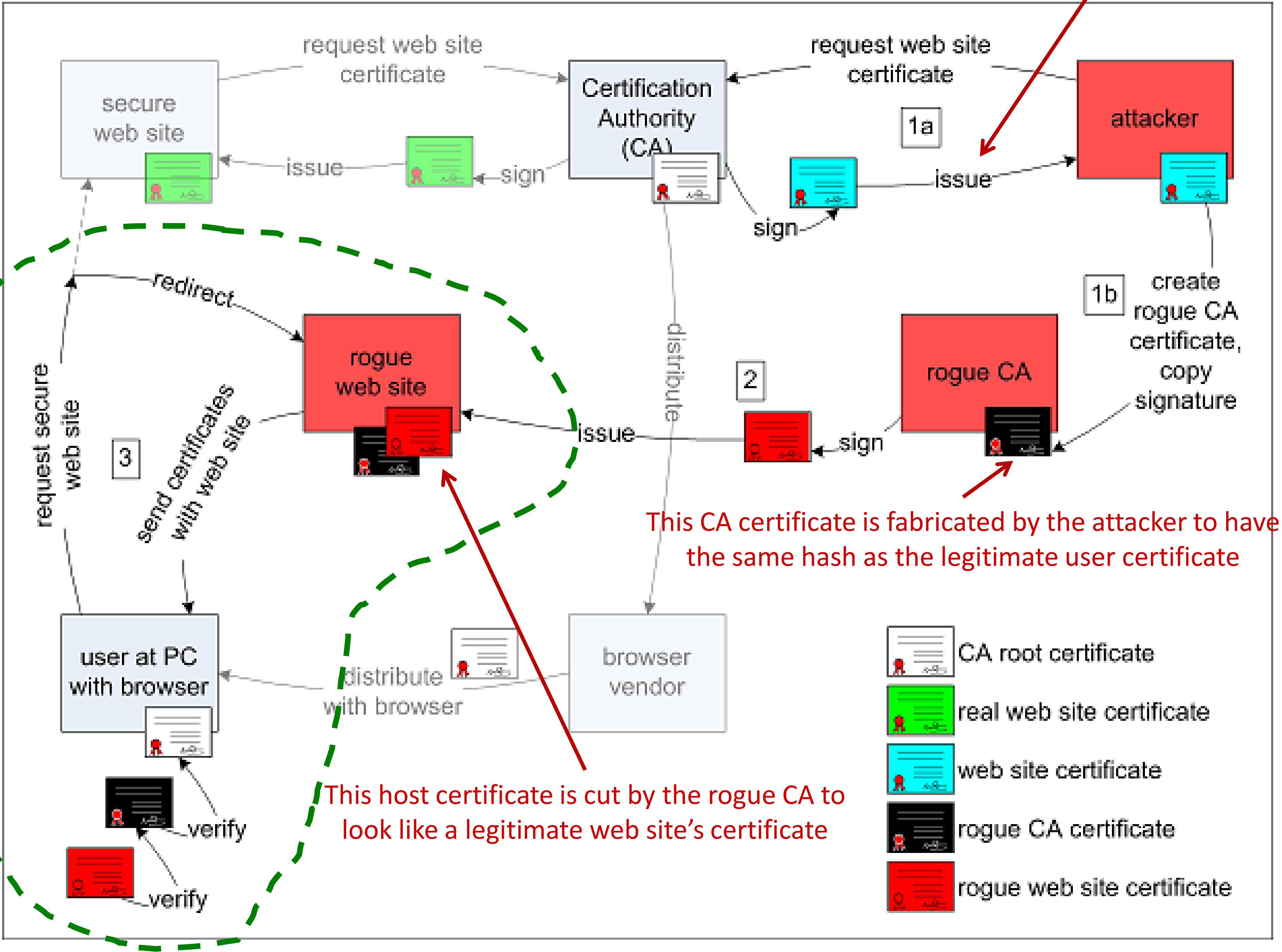


What’s the big deal?

Question: Can you describe a scenario in which having two certificates with the same hash would be problematic?

Attack Overview

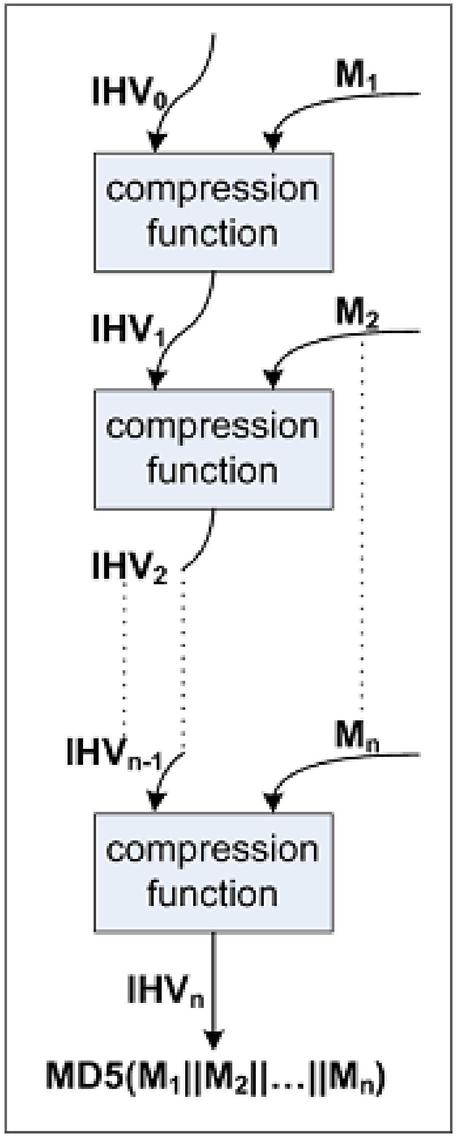
This is a legitimate host certificate obtained through standard channels



This CA certificate is fabricated by the attacker to have the same hash as the legitimate user certificate

This host certificate is cut by the rogue CA to look like a legitimate web site's certificate

How does MD5 work?



Given a variable length input string, MD5 produces a 128-bit hash value

MD5 uses a Merkle-Damgård iterative construction

- 128-bit intermediate hash value (IHV) and a 512-bit message chunk are fed into a compression function that generates a 128-bit output
- This output is compressed with the next 512-bit message chunk and the process is repeated
- The final IHV is the hash value for the message

Note that:

- The initial message must be padded out to a multiple of 512 bits
- IHV$_0$ is a publicly-known fixed value (0x67452301 0xEFCDAB89 0x98BADCFE 0x10325476)

Early attacks against MD5

Early partial attacks on MD5 were suggestive of larger troubles

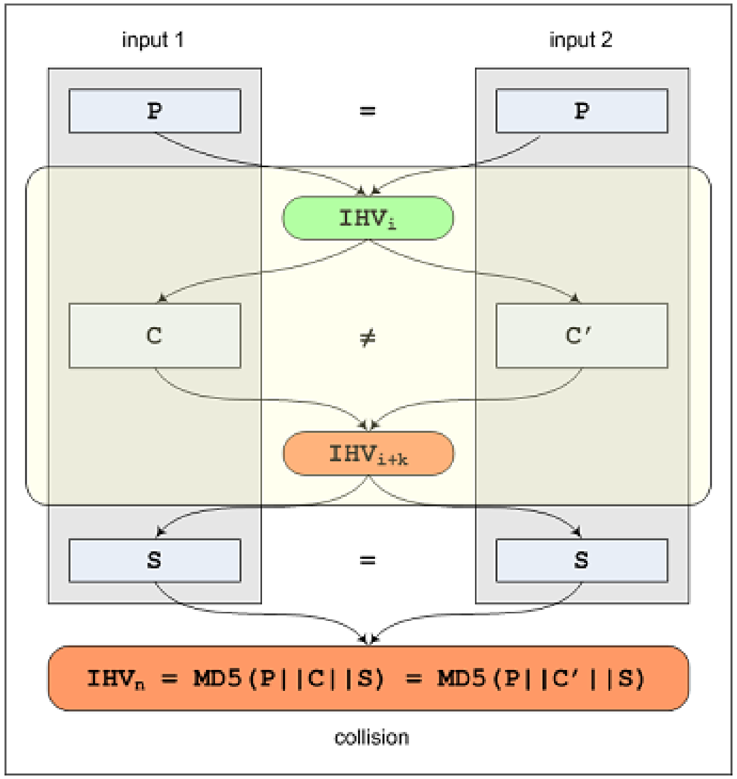
- [1993] Den Boer and Bosselaers found a partial collision of the MD5 compression function (two different IHVs lead to the same output value)
- [1996] Dobbertin found a full collision of the MD5 compression function

The first real attack on MD5 was found by Wang and Yu in 2004

Given any IHV, they showed how to compute two pairs $\{M_1, M_2\}$ and $\{M_1', M_2'\}$ such that:

- $IHV_1 = CF(IHV,M_1) \neq IHV_1' = CF(IHV,M_1')$
- $IHV_2 = CF(IHV_1,M_2) = CF(IHV_1',M_2')$

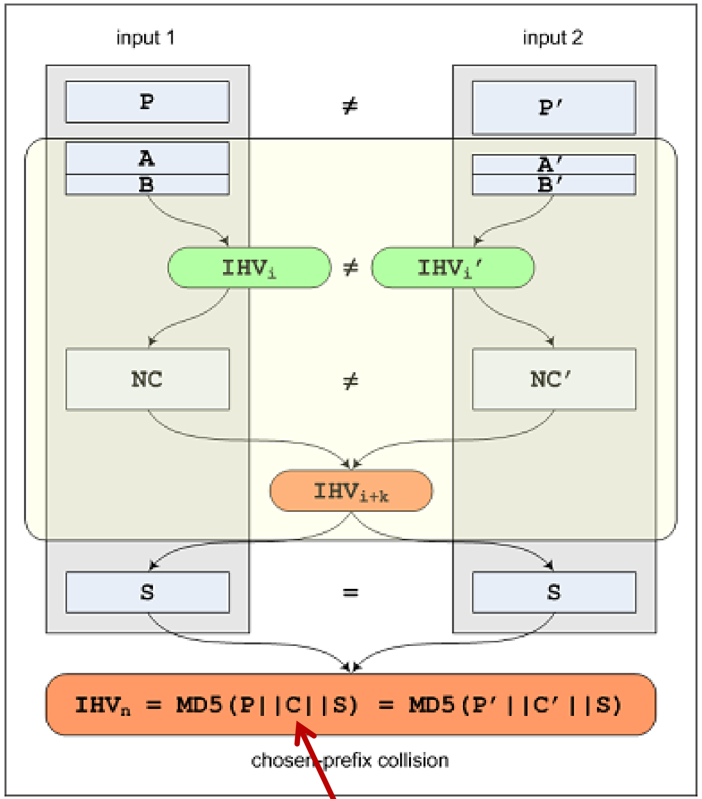
This yields a fairly scary generalization



In 2007 Stevens refined the Wang and Yu attack to allow differing message prefixes

How does this process work?

1. Choose P and P’ at will
2. Choose A and A’ s.t. (P || A) and (P’ || A’) are of the same bit length
3. In a “birthday step”, choose B and B’ such that (P || A || B) and (P’ || A’ || B’) are a multiple of 512 bits and the output IHVs have a special structure
4. This special structure allows the attacker to find two near collision blocks NC and NC’ such that the resulting MD5 function collides!



Note: C = A || B || NC

One use of this attack is generating different documents (plus some hidden content) that end up with the same hash value/signature!

This also opens the door to forged certificates…

How can we launch this attack against X.509?

An X.509 certificate contains quite a bit of information

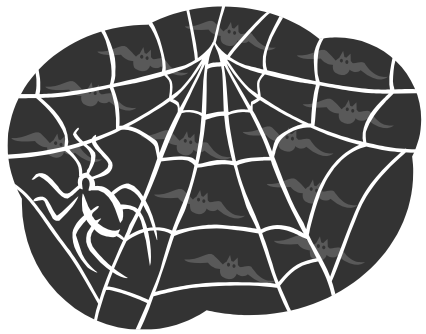
- Version
- Serial number: Must be unique amongst all certificates issued by the same CA
- Signature algorithm ID: What algorithm was used to sign this certificate?
- Issuer's distinguished name (DN): Who signed this certificate?
- Validity interval: Start and end of certificate validity
- Subject's DN: Who is this certificate for?
- Subject's public key information: The public key of the subject
- Issuer's unique ID: Used to disambiguate issuers with the same DN
- Subjects unique ID: Used to disambiguate subjects with the same DN
- Extensions: Typically used for key and policy information
- Signature: A digital signature of (a hash of) all other fields

---

Insight 1: Stevens' chosen prefix attack means that it might be possible to generate two certificates with different subjects, but the same signature!

Insight 2: Collision blocks can potentially be hidden as (part) of the public key block and/or in extension fields!

Successfully launching this attack means finding a CA that will grant a certificate using an MD5 hash



To find such a server, the authors wrote a web crawler

- Crawler ran for about a week
- Found 100,000 SSL certificates
- 30,000 certificates signed by "trusted" CAs
- Of these, 6 issued certificates with MD5-based signatures signed in 2008

Of the certificates found with signatures over MD5 hashes, 97% were issued by RapidSSL (http://www.rapidssl.com/)

RapidSSL issues certificates in an online manner, so they actually made an ideal target for launching this attack

- Predictable timing
- Not human-based, so multiple requests would not seem strange

Question: Why attack a production server instead of a test server?

Unfortunately, we have no control over the certificate serial number or validity fields!

Question: Why might this pose a problem?

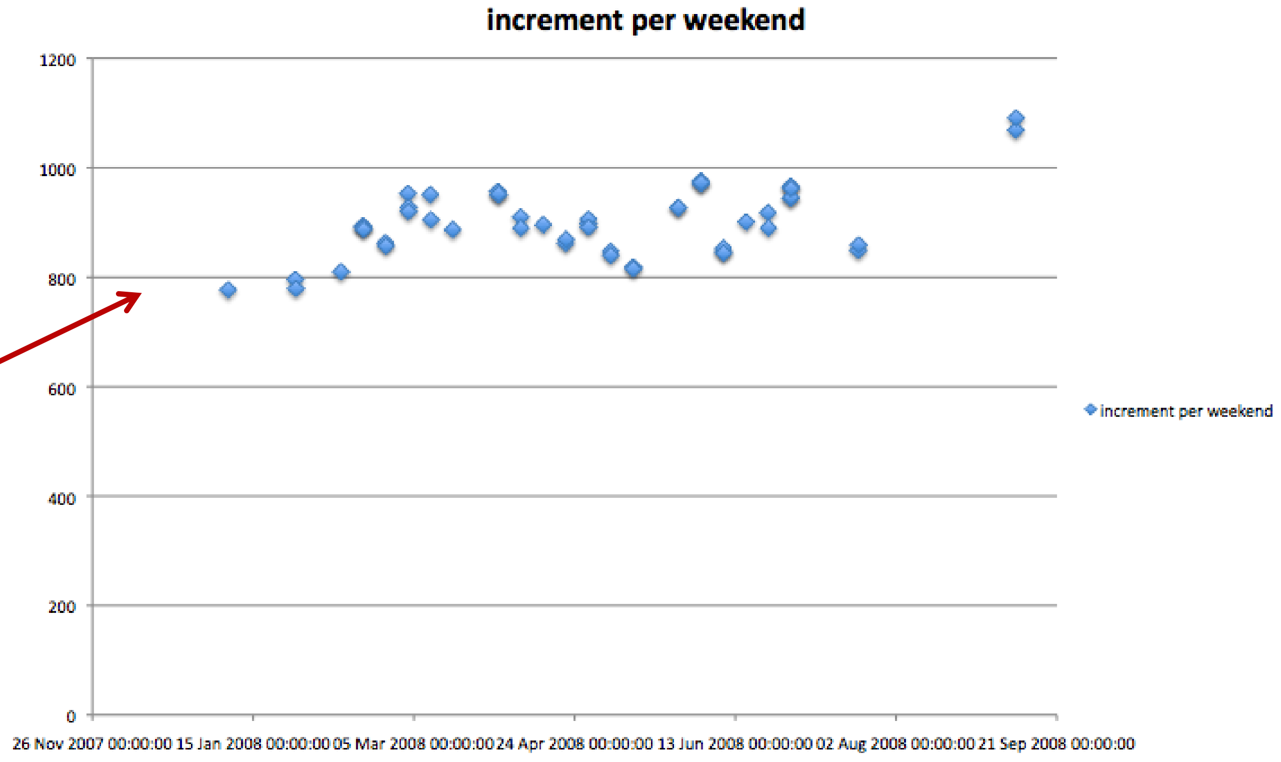
- Because these fields occur in the "chosen" prefix of the certificate and thus must be known before the collision blocks can be computed!

Predicting the validity period turned out to be trivial

- Certificate always issued 6 seconds after it was requested
- Valid for exactly one year

Furthermore, it turns out that RapidSSL uses a counter to populate the serial number field!

800-1000 certificates issued per weekend



Setting up a legitimate certificate request

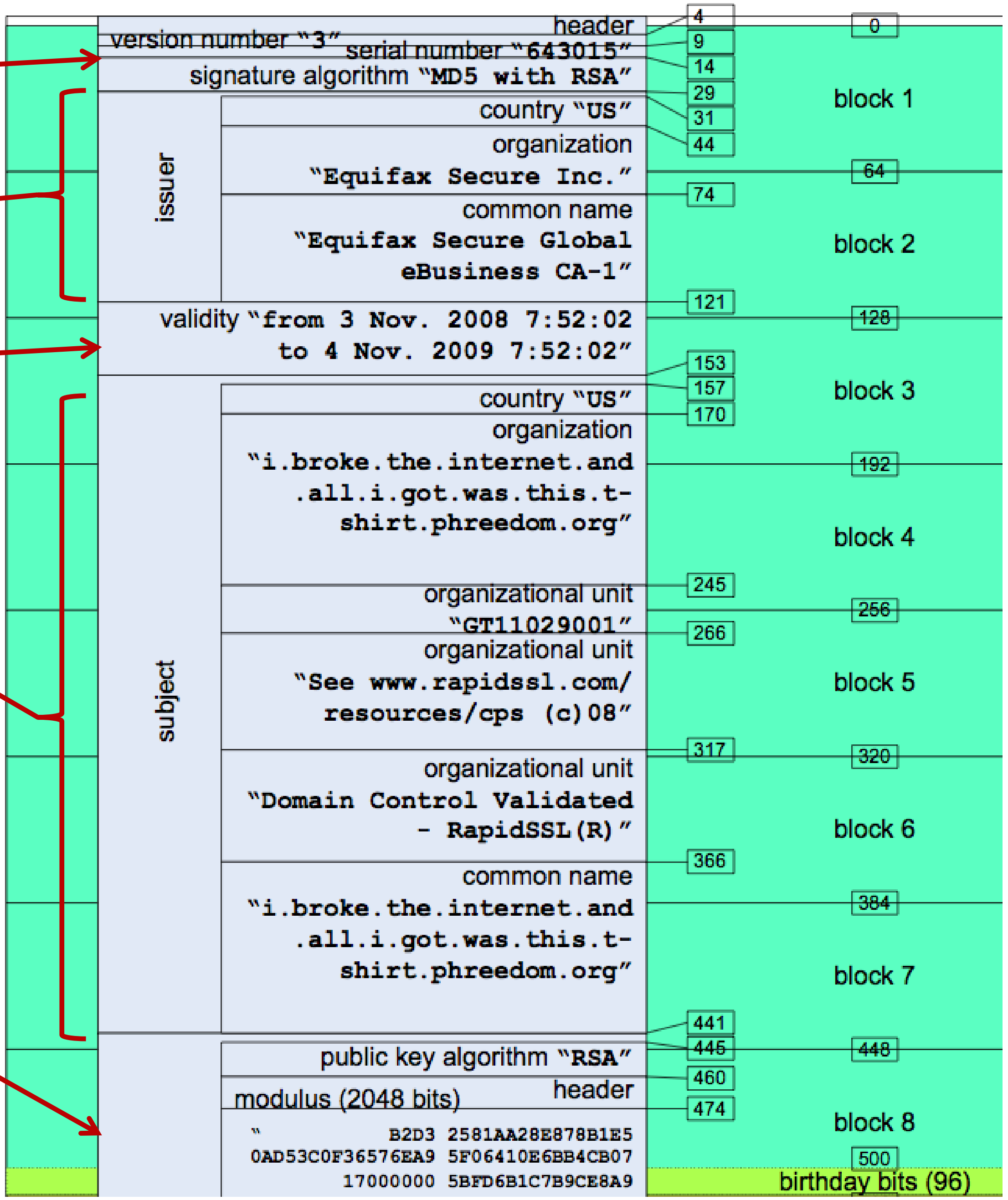
Predicted serial number

Info pulled from CA certificate

Derived validity period

Subject information completely chosen by the attacker

Question: How do you choose a public key (2048-bit modulus and an exponent) when part of it is the collision blocks needed to make the attack work?!?!


header
version number "3"
serial number "643015"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 3 Nov. 2008 7:52:02 to 4 Nov. 2009 7:52:02"
subject
country "US"
organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
organizational unit "GT11029001"
organizational unit "See www.rapidssl.com/resources/cps (c)08"
organizational unit "Domain Control Validated - RapidSSL(R)"
common name "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
public key algorithm "RSA"
modulus (2048 bits)
header
" B2D3 2581AA28E878B1E5
0AD53C0F36576EA9 5F06410E6BB4CB07
17000000 5BFD6B1C7B9CE8A9
4
9
14
29
31
44
74
121
153
157
170
245
266
317
366
441
445
460
474
0
64
128
192
256
320
384
448
500
block 1
block 2
block 3
block 4
block 5
block 6
block 7
block 8
birthday bits (96)

Answer: By being extremely clever

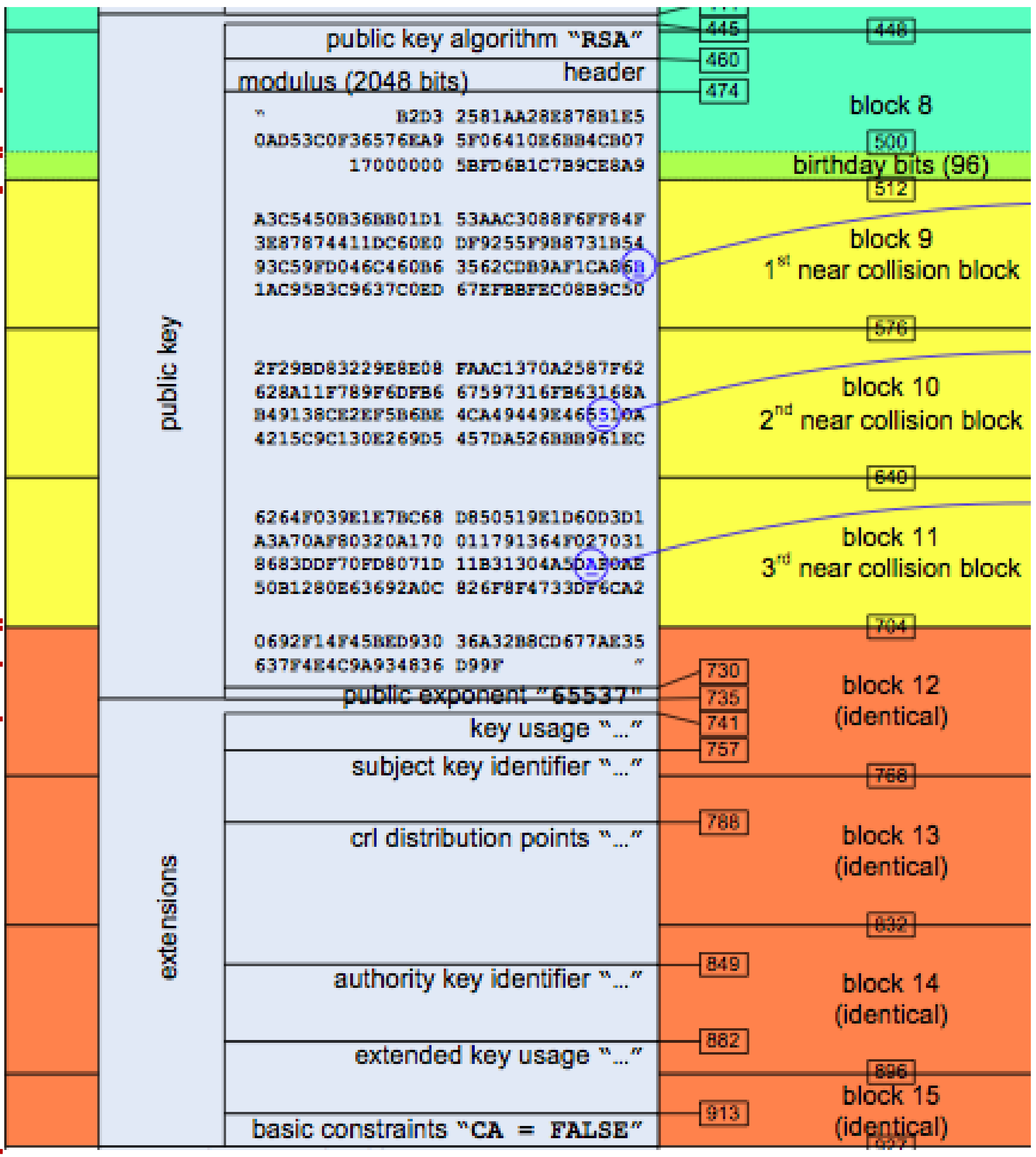
(a) chosen uniformly at random

(b) chosen to set up relationship between this certificate and the rogue CA cert to be generated next

(c) output of the authors' collision-finding algorithm

(d) chosen by the attacker such that when the bits comprising parts (a)-(d) are interpreted as an integer, the result is a number "n" such that n = pq

This is just standard PKI junk…



```
"                B2D3 2581AA28E878B1E5
0AD53C0F36576EA9 5F06410E6BB4CB07
        17000000 5BFD6B1C7B9CE8A9

A3C5450B36BB01D1 53AAC3088F6FF84F
3E87874411DC60E0 DF9255F9B8731B54
93C59FD046C460B6 3562CDB9AF1CA86B
1AC95B3C9637C0ED 67EFBBFEC08B9C50

2F29BD83229E8E08 FAAC1370A2587F62
628A11F789F6DFB6 67597316FB63168A
B49138CE2EF5B6BE 4CA49449E465510A
4215C9C130E269D5 457DA526BBB961EC

6264F039E1E7BC68 D850519E1D60D3D1
A3A70AF80320A170 011791364F027031
8683DDF70FD8071D 11B31304A5DAF0AE
50B1280E63692A0C 826F8F4733DF6CA2

0692F14F45BED930 36A32B8CD677AE35
637F4E4C9A934836 D99F               "
```

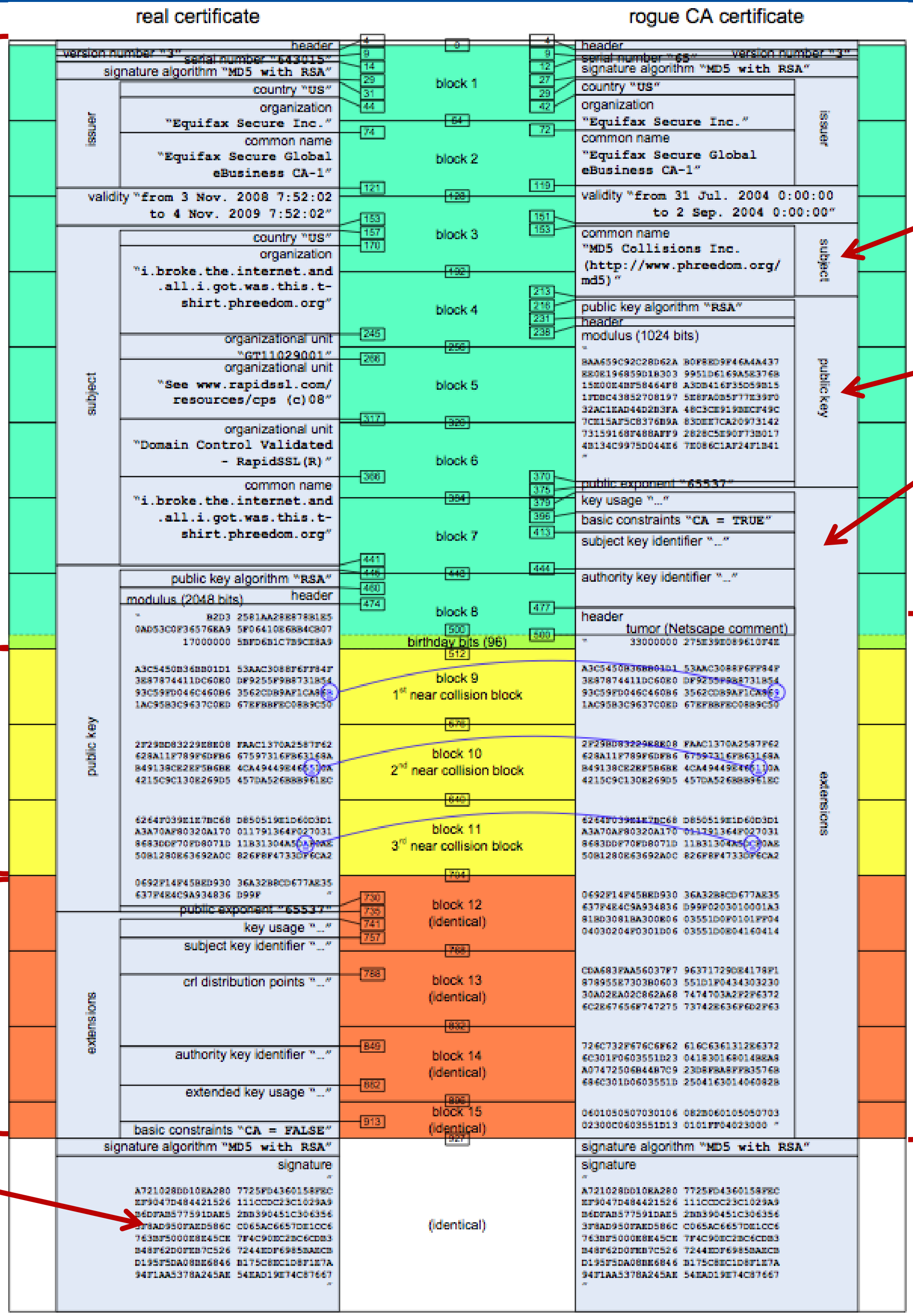
Creating the rogue CA certificate

Chosen prefix

Collision blocks (different but related)

Identical trailers

Signature of CA


real certificate
rogue CA certificate
header
version number "3"
serial number "643015"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 3 Nov. 2008 7:52:02 to 4 Nov. 2009 7:52:02"
subject
country "US"
organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
organizational unit "GT11029001"
organizational unit "See www.rapidssl.com/resources/cps (c)08"
organizational unit "Domain Control Validated - RapidSSL(R)"
common name "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
public key
public key algorithm "RSA"
modulus (2048 bits)
public exponent "65537"
extensions
key usage "..."
subject key identifier "..."
crl distribution points "..."
authority key identifier "..."
extended key usage "..."
basic constraints "CA = FALSE"
signature algorithm "MD5 with RSA"
signature
block 1
block 2
block 3
block 4
block 5
block 6
block 7
block 8
birthday bits (96)
block 9
1st near collision block
block 10
2nd near collision block
block 11
3rd near collision block
block 12 (identical)
block 13 (identical)
block 14 (identical)
block 15 (identical)
(identical)
serial number "65"
validity "from 31 Jul. 2004 0:00:00 to 2 Sep. 2004 0:00:00"
common name "MD5 Collisions Inc. (http://www.phreedom.org/md5)"
public key algorithm "RSA"
modulus (1024 bits)
public exponent "65537"
basic constraints "CA = TRUE"
tumor (Netscape comment)

Shorter subject info

Small public key

Standard public key stuff (Note that this cert is for a CA!)

(Typically ignored) certificate extension to hold collision blocks and trailer from original certificate

Some specifics…

The attack used by the authors had two stages:

1. Finding the 96 “birthday bits”
2. Finding the three near collision blocks

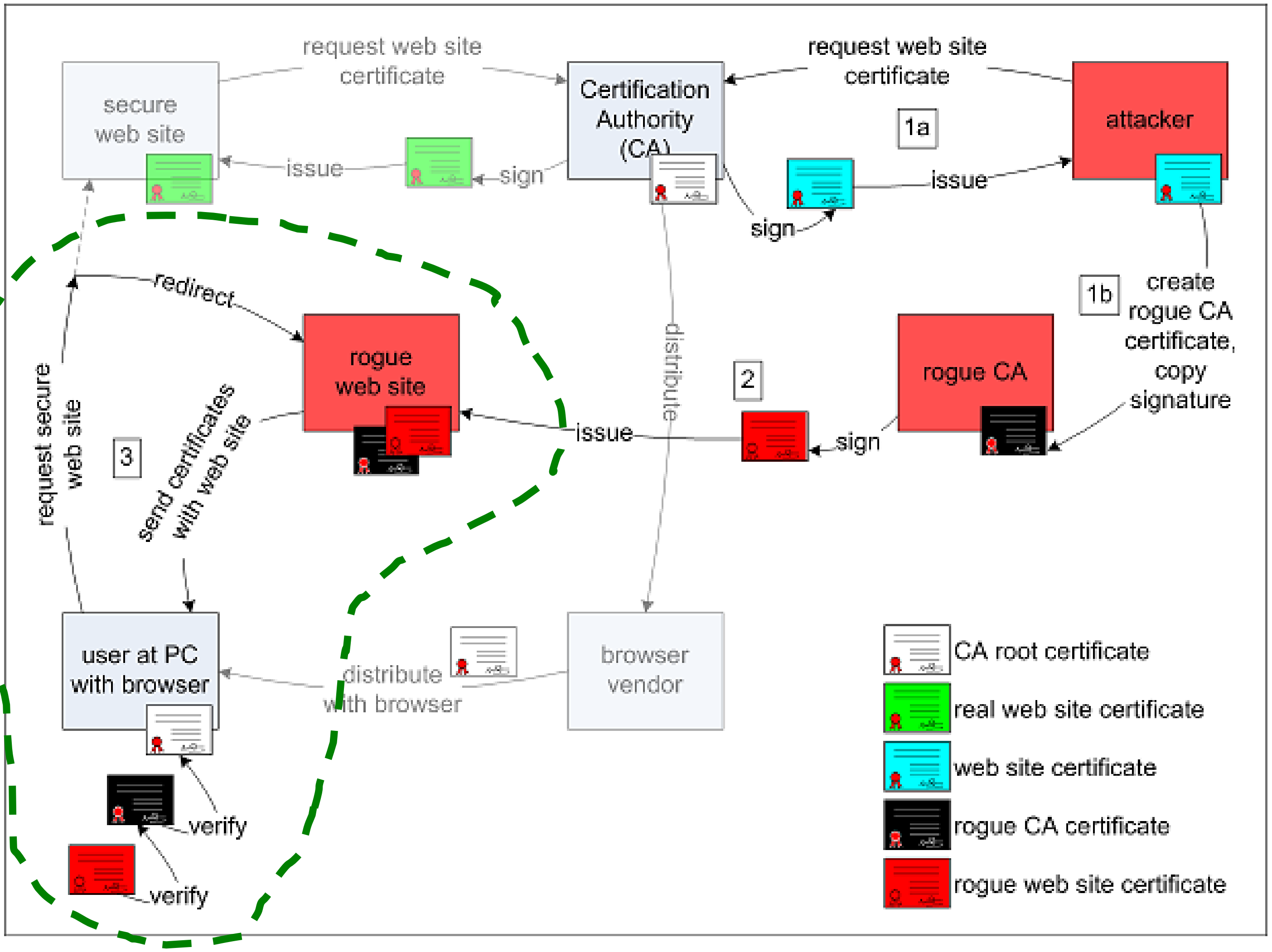
Stage 1 is computationally expensive, but well-suited for execution on the processors used by PlayStation3 game consoles. This took about 18 hours on a cluster of 200 PS3s.

Stage 2 is not terribly expensive, and is better suited to run on commodity PCs. This takes 3-10 hours on a single quad-core PC.

Attacking only on weekends (due to reduced CA load) the authors were able to carry out their attack in 4 weeks for a cost of $657.

This attack has very real implications…

In short, it is possible for a malicious principal to get a cheap X.509 host certificate, and leverage this to create a trusted CA certificate!



The Bottom Line: Consider MD5 broken!

Broken? What do you mean by broken?

Since MD5 does not have all three properties required of cryptographic hash functions, it is not considered to be "cryptographically strong"

While you could still use MD5 for non-cryptographic applications (such as?), it is better to avoid it altogether

---

SHA-1 is the cryptographic hash function that is now most often used

Attacks against SHA-1 have been announced, but still require a large amount of computing effort to mount

- e.g., Collision finding in 2^{63} steps, rather than 2^{80}

NIST recently ran a competition to design SHA-3, a new hash family to complement the SHA-{1,2} family of hash functions

- Welcome, Keccak

How did vendors react to the MD5 break?

2008, December 30th: This work was presented at the Chaos Computing Congress

December 31st

- Verisign issues a statement, stops using MD5
- Microsoft issues Security Advisory (961509): “Research proves feasibility of collision attacks against MD5”
- Mozilla has a short item in the Mozilla Security Blog: “MD5 Weaknesses Could Lead to Certificate Forgery”

January 2nd

- RSA has an entry in the Speaking of Security blog: “A Real New Year’s Hash”
- US-CERT, the US Department of Homeland Security’s Computer Emergency Readiness Team, published Vulnerability Note VU#836068: “MD5 vulnerable to collision attacks”

January 15th

- Cisco published “Cisco Security Response: MD5 Hashes May Allow for Certificate Spoofing”

Discussion

Question 1: People have had evidence that MD5 was weak since the mid 1990s. Why did it take this long to finally convince vendors to stop using MD5? Do you think that this will change the response to future cryptographic vulnerabilities?

Question 2: In the 1980s, Adi Shamir and Eli Biham “discovered” differential cryptanalysis, which is a general means of attacking block ciphers. It was later revealed that NSA and IBM actually discovered this technique first, but kept it a secret. What if this MD5 attack was known before it was discovered in the public domain? What would the implications be?

The aftermath: Flame malware

“[A]rguably, it is the most complex malware ever found”

Developed by NSA and GCHQ for cyber espionage targeting middle eastern countries. Records:

- Screenshots
- Audio
- Network activity
- Keyboard activity
- Nearby devices’ contacts via bluetooth

Used components signed by Microsoft?!

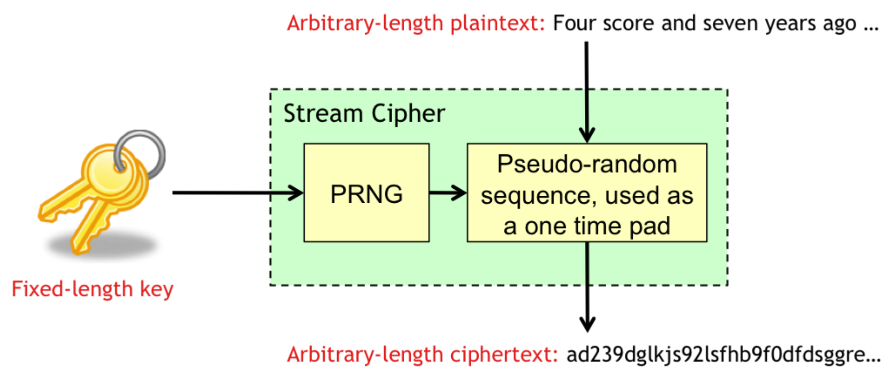
Counterfeit certificate crafted from a terminal service certificate with code-signing enabled, signed using MD5

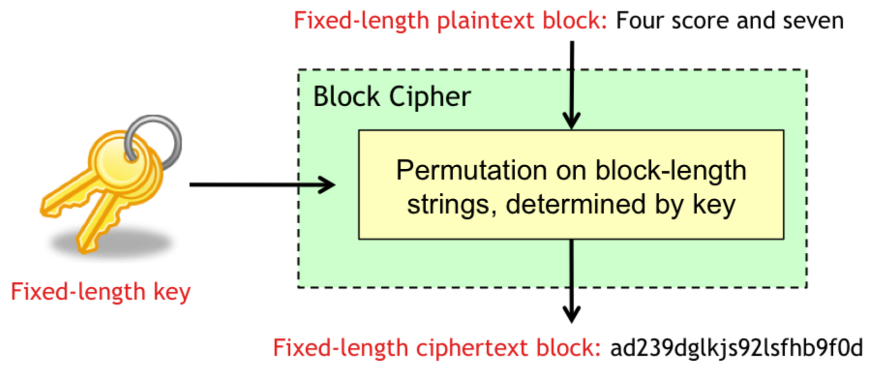
Previously unknown variant of the chosen-prefix collision attack

First detected in 2012…

… but its main component was first seen in 2007

So far, we have treated cryptography largely as a black box


Arbitrary-length plaintext: Four score and seven years ago …
Stream Cipher
PRNG
Pseudo-random sequence, used as a one time pad
Fixed-length key
Arbitrary-length ciphertext: ad239dglkjs92lsfhb9f0dfdsggre…


Fixed-length plaintext block: Four score and seven
Block Cipher
Permutation on block-length strings, determined by key
Fixed-length key
Fixed-length ciphertext block: ad239dglkjs92lsfhb9f0d

This certainly simplifies the engineering process

- Smart people build the boxes, and we can just put them together
- Just worry about I/O, not the (messy) details

Unfortunately, it also abstracts away lots of details…

Today, we'll see how these types of details can cause the downfall of otherwise hardened black boxes

Can we break symmetric key cryptography without brute forcing the keyspace?

In theory, there is no difference between theory and practice. In practice, there is.

-- Not Yogi Berra

In theory, cryptographic keys are chosen randomly.

- For an m-bit keyspace, each of the 2^m keys is equally likely
- As such, figuring the key out requires a brute force search

In practice, things are not really random

- Computers are deterministic machines!
- Keys are picked using a pseudo-random process
- Figuring out this process can yield keys with less than brute-force effort

Question: How many people have used a PRNG? How did you seed it?

Goldberg and Wagner used this insight to break Netscape's key generation routines

All web browsers that use TLS/SSL need to generate secret keys that are exchanged during the connection setup process

- Typically, this is done using a cryptographically strong PRNG
- The initial seed to the PRNG determines the keys that are generated

After some reverse engineering, PRNG was found to be seeded using

- The current time
- The process ID of the web browser (PID)
- The parent process ID of the web browser (PPID)

The result: 128-bit keys were generated from 47 bits of randomness!

With a user account on the machine, the search could be reduced to about 10^6 = 1,000,000 guesses!

How can we prevent this?

What is one way that we might be able to prevent these types of well-known coding errors?