

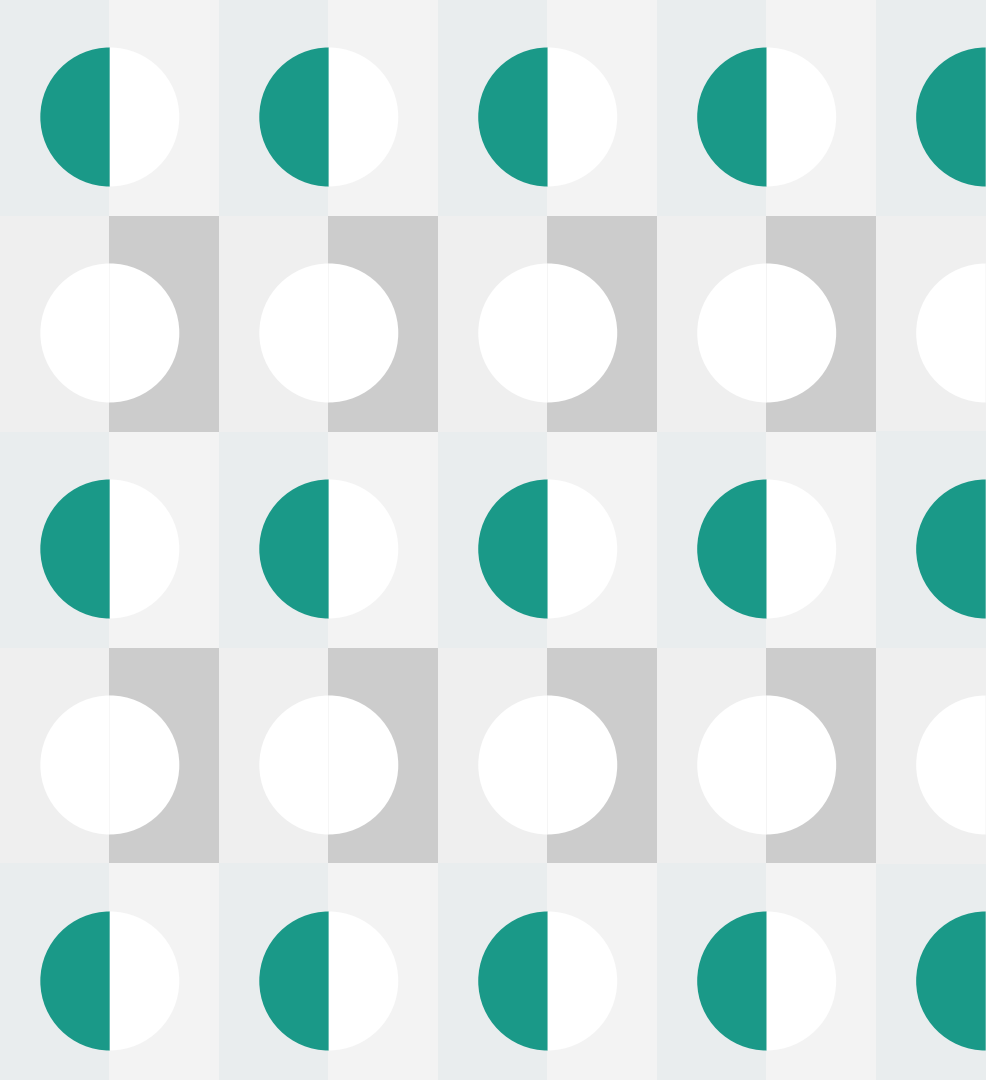
Section 5

Public Key Crypto Topics:

RSA, Cryptanalysis with CBC-MAC

Administrivia

- Homework 2 due next Friday (11/5)
 - Individual assignment
 - Hands-on cryptography
- Final Project checkpoint #1 due next next Friday (11/12)
 - Group members' names and UWNetIDs
 - Presentation topic

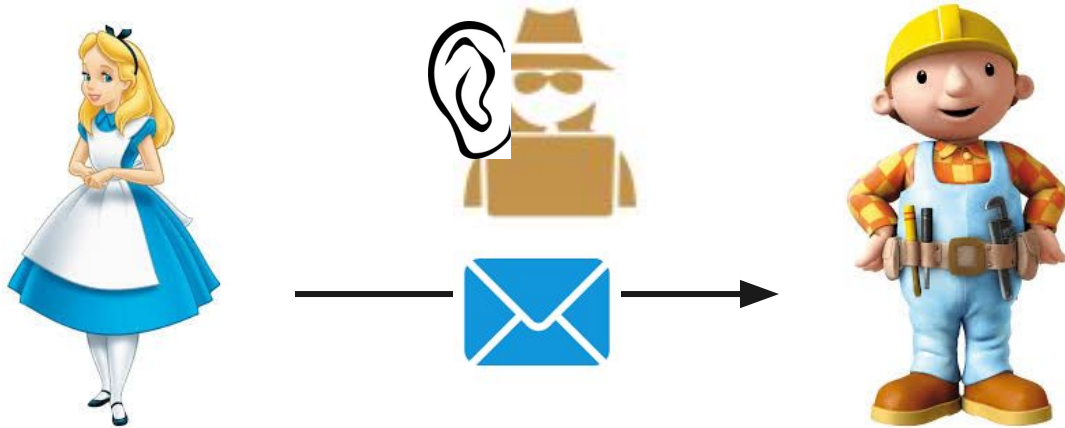


RSA: **Key generation,** **encryption, and** **decryption**

Public Key Cryptography Review

Alice wants to send Bob an encrypted message

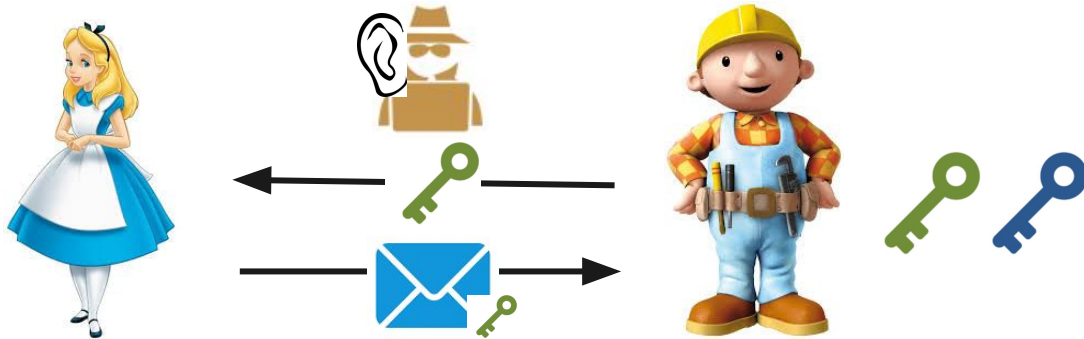
- Goal: Confidentiality
- Problem: Eve can intercept key



Public Key Cryptography Review

Solution: public key cryptography (aka asymmetric cryptography)

- Public-private keypair
- Alice encrypts using Bob's public key
- Bob decrypts using Bob's private key



RSA Cryptosystem Review

Key generation:

- Generate large primes p, q
- Compute $N=pq$ and $\phi(N)=(p-1)(q-1)$
- Choose e coprime to $\phi(N)$
 - Typically $e=3$ or $e=2^{16}+1=65537$
- Find (unique) d such that $ed \equiv 1 \pmod{\phi(N)}$
 - (equivalent to solving $1 = (e \cdot d) \pmod{\phi(n)}$)

Public key = (e, N) ; Private key = (d, N)

Encryption of m : $c = m^e \pmod N$

Decryption of c : $c^d \pmod N = (m^e \pmod N)^d \pmod N = m^1 \pmod N = m$



Adi Shamir, Ron Rivest, Len Adleman
[Photo from Dan Wright]

RSA Practice

Public key: $N = 33$, $e = 7$

Step 1: Find $\phi(N)$

Step 2: Find the decryption key, d

- $ed \equiv 1 \pmod{\phi(N)}$

Step 3: Decrypt the cryptogram

- $c^d \pmod N = m$

- 'A' = 1, 'B' = 2, ...

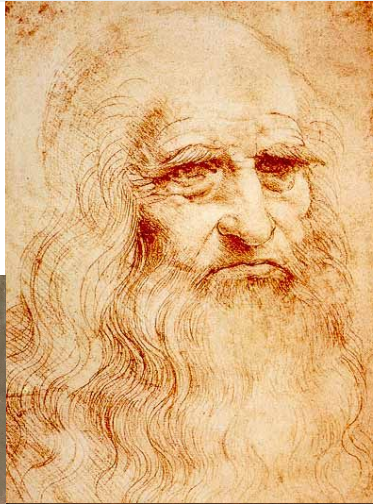
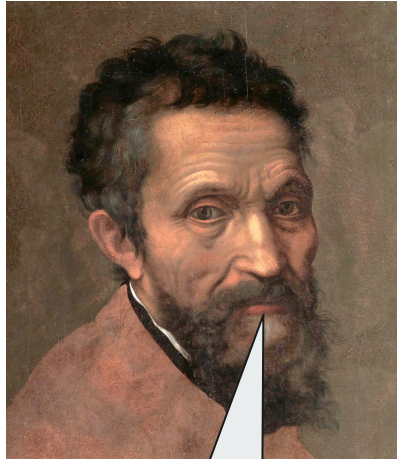
Cryptograms:

12 14 27 20 1 6 16 27

6 1 25 2 1 14 12

7 15 9 2 14 12 1 20 28 14 12 27

16 27 20 1 26 14 12 12 27



Cowabunga!

TEENAGE MUTANT NINJA

TURTLES



SG 2012

RSA Strength

“RSA problem”: decrypt only using the public key

- Factoring N is hard
- No known efficient algorithm
- Trapdoor function: easy to go forward, hard to go back

RSA Factoring Challenge (1991-2007)

- Cash prizes for factoring large N values (up to \$200,000 (!))
- Only the smallest 23 of 54 factored so far...

Shor’s Algorithm

- Quantum computer algorithm to factor integers
- Largest number factored so far: 21 🕶️

RSA-2048:

25195908475657893494027183240
04839857142928212620403202777
71378360436620207075955562640
18525880784406918290641249515
08218929855914917618450280848
91200728449926873928072877767
35971418347270261896375014971
82469116507761337985909570009
73304597488084284017974291006
42458691817195118746121515172
65463228221686998754918242243
36372590851418654620435767984
23387184774447920739934236584
82382428119816381501067481045
16603773060562016196762561338
44143603833904414952634432190
11465754445417842402092461651
57233507787077498171257724679
62926386356373289912154831438
16789988504044536402352738195
13786365643912120103971228221
20720357

RSA Today

- Still used today but mostly in legacy applications
 - SSH keys, TLS, etc.
 - But not preferred...
- Need big keys for RSA
 - At least 2048 bits
- Bigger keys \Rightarrow slower computation
- Modern encryption schemes exist, such as Elliptic-Curve Cryptography (ECC)

[Explanation and Diagram for ECC](#)



**Demonstration:
Finding vulnerabilities
in CBC-MAC with
cryptanalysis**

Is encryption (confidentiality) enough?

Scenario: David wants to send out an email about exam times - and a hacker has learned the encryption key



dkohlbre@cs

“Final!!!
KNE 110
Monday
2:30PM”



AES 128-bit key,
CBC mode

ok



In this case, an adversary
doesn't gain anything
important by learning the
content of this message.



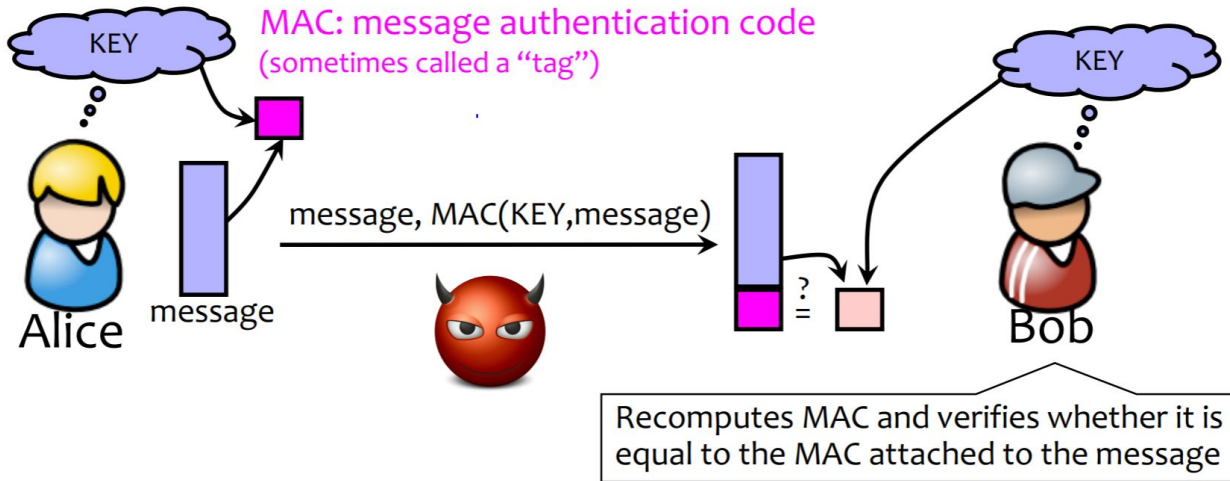
Is encryption (confidentiality) enough?

But, the attacker could tamper with the message during transmission, and the recipient would not know - so we need to ensure **integrity**

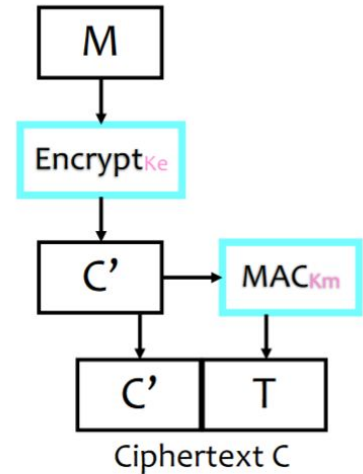


MAC (Message Authentication Code)

Provides integrity and authentication: only someone who knows the KEY can compute correct MAC for a given message.



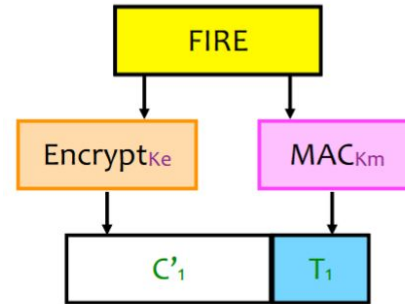
When do we MAC?



Encrypt-then-MAC

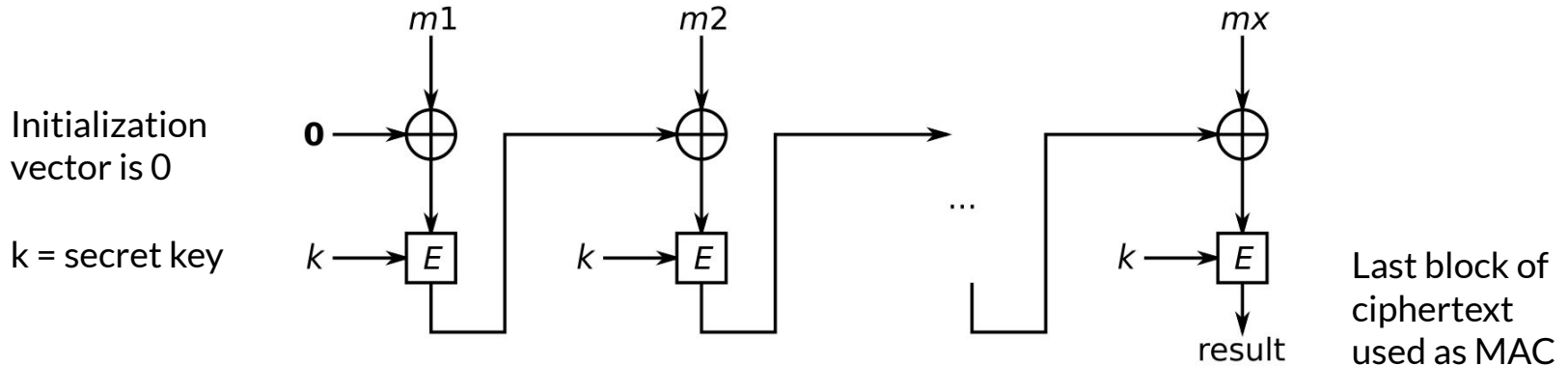
The good:
Encrypt-then-MAC
MAC-then-encrypt
Not as good as
Encrypt-then-MAC

The bad (& ugly):
Encrypt-and-MAC
MAC is deterministic! Same
plaintext \rightarrow same MAC



How do we create a MAC?

CBC-MAC: Encrypt the message in CBC mode, use the last block as the MAC



*CBC-MAC is not the only MAC algorithm - today most use HMAC; we'll show why next



Is CBC-MAC vulnerable?

- How could we find out?
 - Cryptanalysis: using mathematical analysis to rigorously reason about a cryptographic system
- Let's use cryptanalysis to find a collision
 - two different inputs leading to the same MAC tag
 - (violating collision resistance)

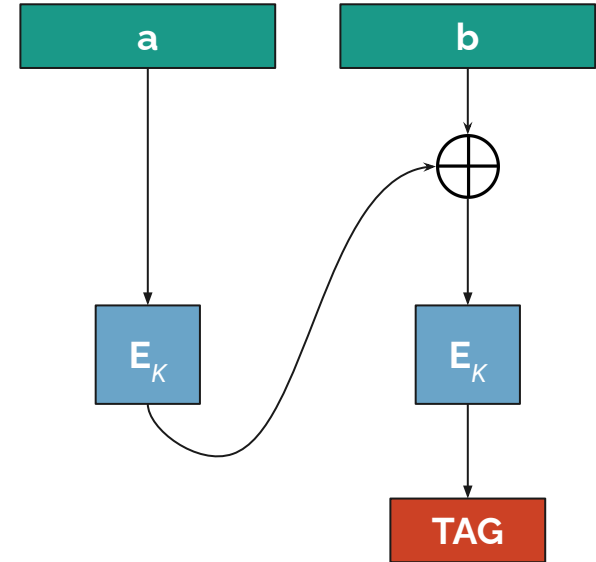
Exercise: CBC-MAC collision vulnerability

Suppose a and b are both one block long, and suppose the sender MACs a , b , and $a || b$ with CBC-MAC.

An attacker who intercepts the MAC tags for these messages can now forge the MAC for the message

$$b || (M_K(b) \oplus M_K(a) \oplus b)$$

which the sender never sent. The forged tag for this message is equal to $M_K(a || b)$, the tag for $a || b$. Justify mathematically why this is true.



$a || b$: a and b concatenated
 $M_K(a)$: MAC for message a
 $E_K(a)$: ciphertext for message a

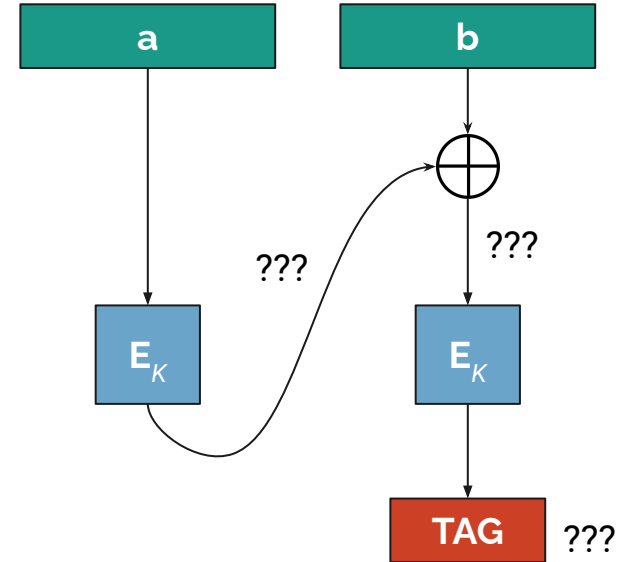
Exercise: CBC-MAC collision vulnerability

Prove:

$$M_K(b || (M_K(b) \oplus M_K(a) \oplus b)) = M_K(a || b)$$

Step 1: Figure out what $M_K(a)$, $M_K(b)$, and $M_K(a || b)$ in terms of the encryption key.

Annotate sketch with the sender's messages and MACs.



Exercise: CBC-MAC collision vulnerability

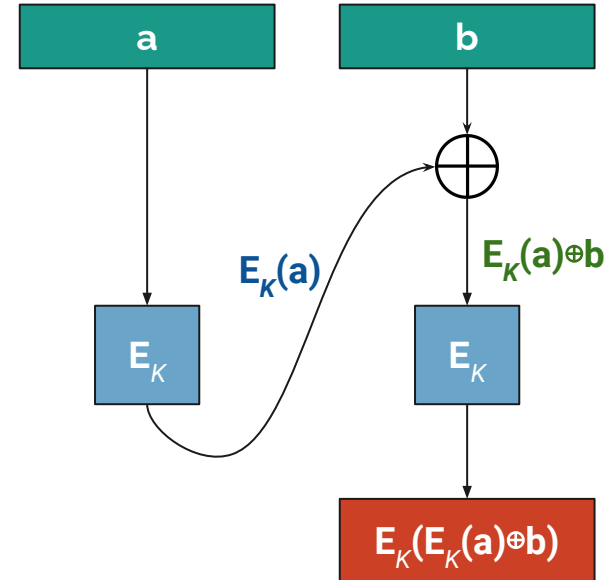
Prove:

$$M_K(b || (M_K(b) \oplus M_K(a) \oplus b)) = M_K(a || b)$$

$$M_K(a) = E_K(a)$$

$$M_K(b) = E_K(b) \text{ (not shown)}$$

$$M_K(a || b) = E_K(E_K(a) \oplus b)$$



Exercise: CBC-MAC collision vulnerability

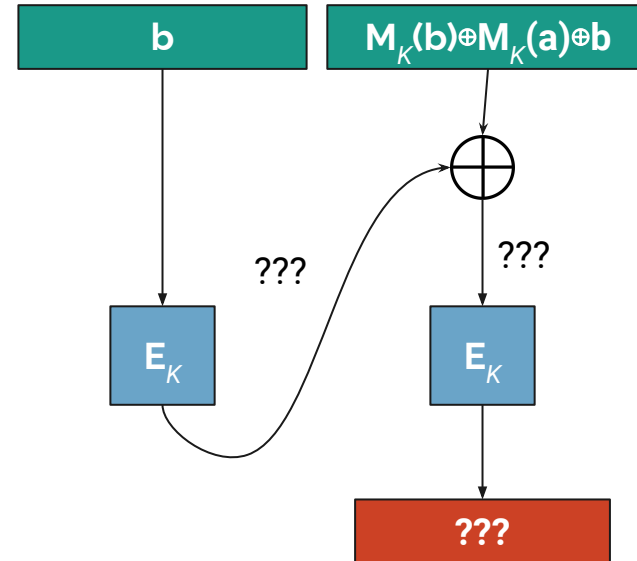
Prove:

$$M_K(b \parallel (M_K(b) \oplus M_K(a) \oplus b)) = M_K(a \parallel b)$$

Step 2: Figure out $M_K(b \parallel (M_K(b) \oplus M_K(a) \oplus b))$.

For the MAC of the attacker's message

$b \parallel (M_K(b) \oplus M_K(a) \oplus b)$, what are the values of the ???'s?



Exercise: CBC-MAC collision vulnerability

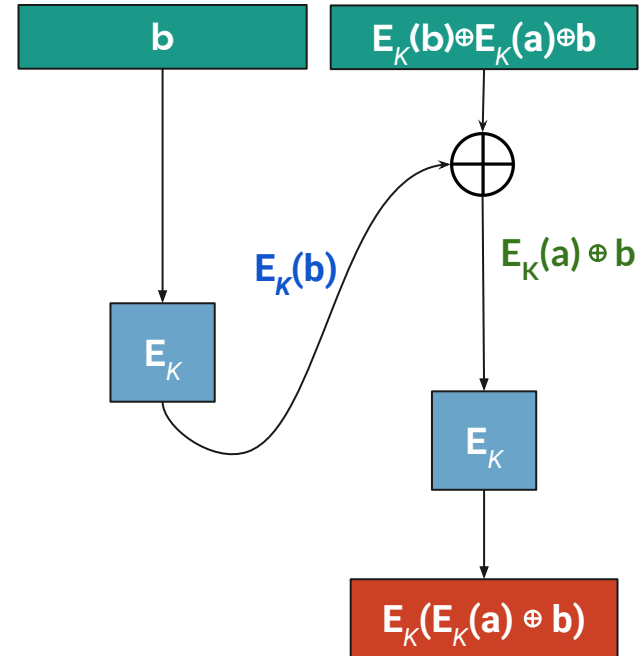
Prove:

$$M_K(b || (M_K(b) \oplus M_K(a) \oplus b)) = M_K(a || b)$$

$$\begin{aligned} & M_K(b || (M_K(b) \oplus M_K(a) \oplus b)) \\ &= M_K(b || (E_K(b) \oplus E_K(a) \oplus b)) \\ &= E_K(\boxed{E_K(b) \oplus E_K(b)} \oplus E_K(a) \oplus b) \end{aligned}$$

These terms
cancel out

$$= E_K(E_K(a) \oplus b) \leftarrow \text{This is the same as } M_K(a || b)!$$



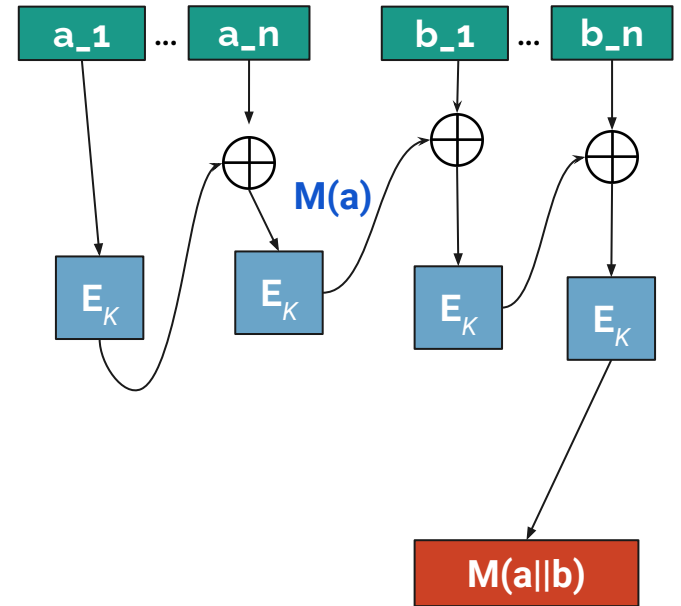


So what?

- We can prove, just using the specification of CBC-MAC, that the messages $b || (M(b) \oplus M(a) \oplus b)$ and $a || b$ share the same tag. This approach is a common method used in cryptanalysis.
- We broke the *theoretical* guarantee that no two different messages will never share a tag.
- If you were to use CBC-MAC in a protocol, it provides information about specific weaknesses and how not to use it.

Generalized

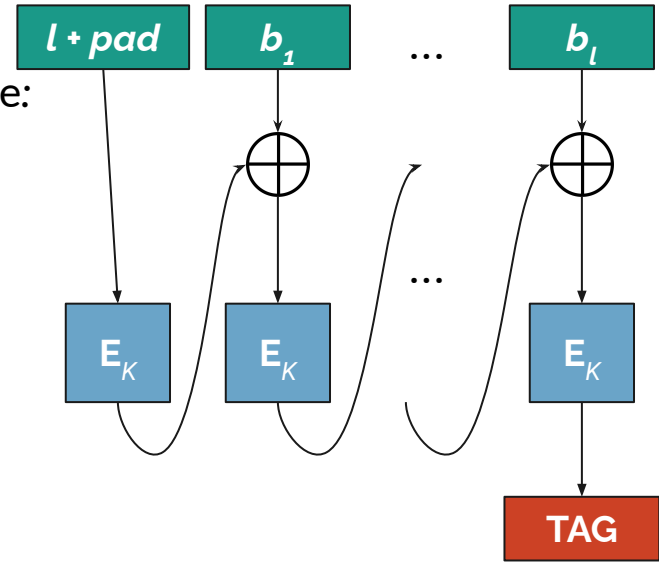
- For any length a, b : $M(a) \oplus b, a || b$ have same tag
- $M(a || b) = M(M(a) \oplus b)$



Safer CBC-MAC for variable length messages

For a message m of length l :

1. Construct s by prepending the length of m to the message:
 $s = \text{concat}(l, m)$
 2. Pad s until the length is a multiple of the block size
 3. Apply CBC-MAC to the padded string.
 4. Output the last ciphertext block, or a part of it. Don't output intermediates.
- Now $sM(a || b) \neq sM(sM(a) \oplus b)$
 - Because $sM(a||b)=M(\text{concat}(l, a || b))$



Or....

- Or encrypt output with another block cipher under a different key (CMAC). Or use HMAC, UMAC, GMAC.

THANKS FOR COMING TO SECTION!



Eve

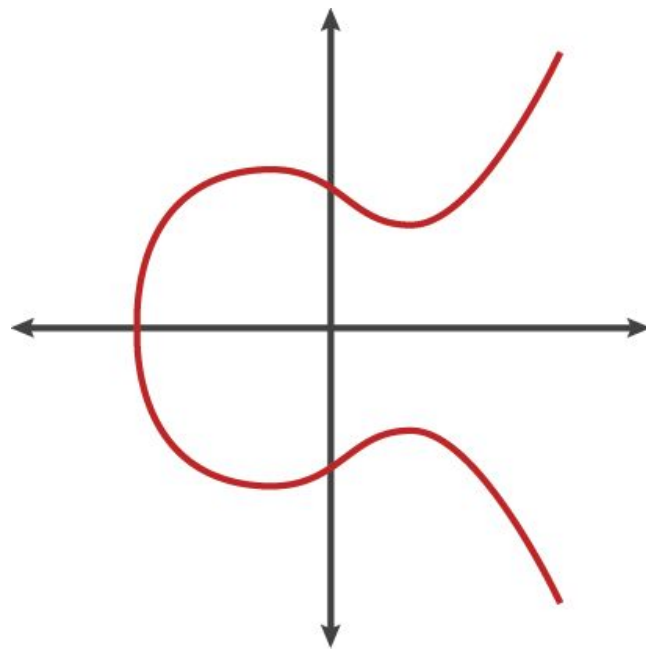
**Bob's
message**

Alice

Elliptic-Curve Cryptography (ECC)

$$y^2 = x^3 + ax + b$$

- First suggested independently by Neal Koblitz (UW Math faculty!) and Victor S. Miller in 1985
- Widespread adoption started in the last 2 decades



[visuals from Cloudflare]

Elliptic-Curve Cryptography (ECC)

Special operation: \circ (“dot”)

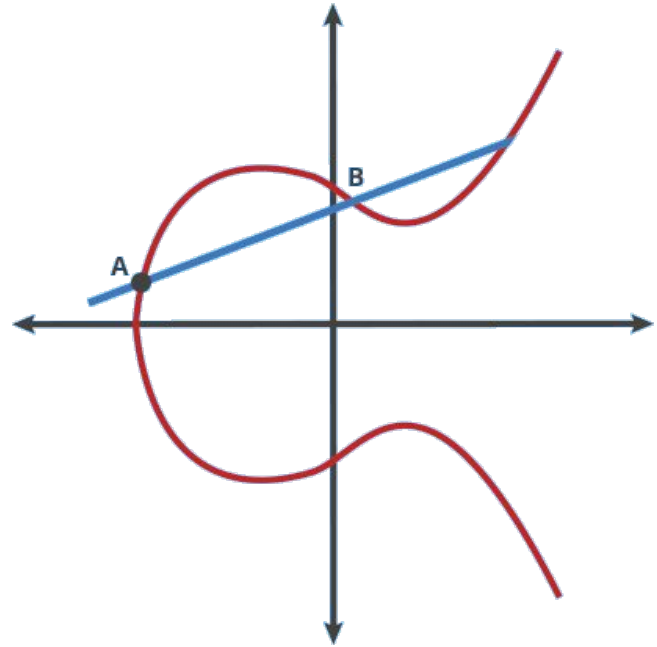
- $A \circ B = C, A \circ C = D, \dots$
- $nA = A \circ \dots \circ A$ (n times)
- $x(yA) = y(xA) = xyA$
- Given point P, hard to find n s.t. $nA = P$
- Pattern behaves “randomly”

Private key: n (integer)

Public key: P (point on curve, $P = nG$)

Public knowledge: G (generator point)

and curve parameters



[visuals from Cloudflare]

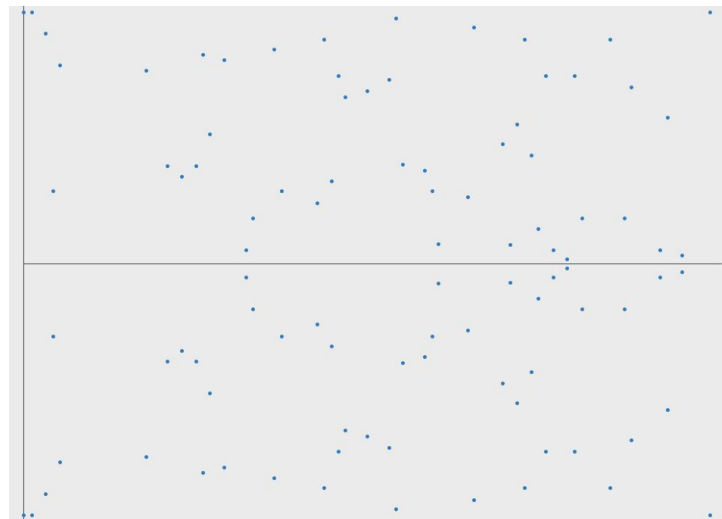
ECC In Practice

Wrap the graph about x and y axes

- Achieves the same effect as modulo, in RSA
- Want prime numbers as the bounds
- Elliptic Curve Discrete Logarithm Problem™

“Safe” Curves?

- NIST recommendations are “fast”, but suspicious
- djb et al. show their work for recommendations
- More: <https://safecurves.cr.yt.to/>



[visuals from Cloudflare]

ECC vs RSA

Pros:

- Same strength using smaller keys
- Smaller keys \Rightarrow faster computation
- ECDLP harder(?) than DLP

Cons:

- Hard to understand
- Hard to implement correctly
- Suspicious implementations (NSA 🤔)

Security Strength	Symmetric Key Algorithms	FFC (DSA, DH, MQV)	IFC* (RSA)	ECC* (ECDSA, EdDSA, DH, MQV)
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256-383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384-511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

[table from NIST (SP 800-57 PART 1 REV. 5)]

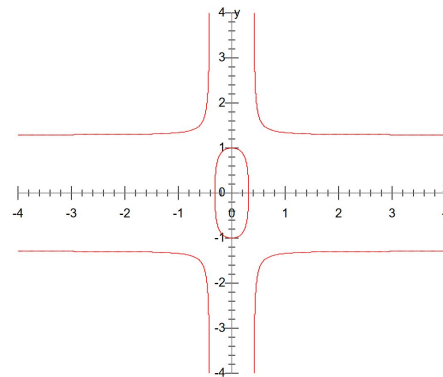
Ultimately: ECC can achieve the same security with smaller keys and faster operations.

ECC In The Wild

ECC can be substituted for $(\mathbb{Z}_p)^\times$ in DL-based protocols:

- Elliptic Curve **D**iffie-**H**ellman
- Elliptic Curve Integrated **E**ncryption Scheme
- Elliptic Curve Digital **S**ignature Algorithm
- Edwards-curve* Digital **S**ignature Algorithm

Most digital certificates use ECDSA (e.g. P-256)
or EdDSA (e.g. ed25519)



*Twisted Edwards curve
[Wikipedia]



**Certificates in Practice
&
Certificate Authority
(CA)**

What are certificates

- A security certificate is a small data file used to establish the identity, authenticity and reliability of a website.

Think of it as a passport!

- TLS/SSL: Encryption and authentication for connections

Note that certificates are not dependent on protocols.



Information on a certificate

- An X.509 certificate a standard format for public key certificates.
 - Different versions, most common: X.509 v3
 - Not all certificates require public trust
- Includes:
 - public key
 - digital signature
 - Issuing CA
 - Additional information about the certificate




Example: Chrome

🔒 google.com

Connection is secure ✕

Your information (for example, passwords or credit card numbers) is private when it is sent to this site.
[Learn more](#)

📍 Location Allow ▾

📄 Certificate (Valid) 

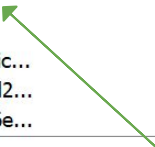
🍪 Cookies (27 in use)

⚙️ Site settings

Certificate ✕

General Details Certification Path

Show <All> ▾

Field	Value
📄 Issuer	GTS CA 101, Go...
📄 Valid from	Tuesday, Januar...
📄 Valid to	Tuesday, March ...
📄 Subject	*.google.com, G...
📄 Public key	ECC (256 Bits) 
📄 Public key para...	ECDSA_P256
🔑 Enhanced Key ...	Server Authentic...
🔑 Subject Key Id...	92429cb273a2d2...
🔑 Authority Key I...	KeyID=98d1f86e...

EC Encrypted with key size = 256

Edit Properties... Copy to File...

OK

Example: Firefox

https://getpocket.com/explore/item/johnny-cash-s-at-folsom-prison-at-50-an-oral-hist

Page Info — https://getpocket.com/explore/item/johnny-cash-s-at-folsom-prison...

General Media Permissions Security

Website Identity

Website: getpocket.com
Owner: This website does not supply ownership information.
Verified by: Amazon
Expires on: Friday, December 17, 2021

Privacy & History

Have I visited this website prior to today? Yes, once
Is this website storing information on my computer? Yes, cookies [Clear Cookies and Site Data](#)
Have I saved any passwords for this website? No [View Saved Passwords](#)

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

[Help](#)

[View Certificate](#)



Certificate

getpocket.com Amazon Amazon Root CA 1

Subject Name

Common Name getpocket.com

Issuer Name

Country US
Organization Amazon
Organizational Unit Server CA 1B
Common Name Amazon

Validity

Not Before 11/17/2020, 4:00:00 PM (Pacific Standard Time)
Not After 12/17/2021, 3:59:59 PM (Pacific Standard Time)

Subject Alt Names

DNS Name getpocket.com
DNS Name readitlater.com
DNS Name pocket.co
DNS Name www.getpocket.com
DNS Name l.getpocket.com
DNS Name theproductivitypack.com
DNS Name www.readitlater.com
DNS Name aproductiveyear.com
DNS Name readitlaterlist.com
DNS Name www.readitlaterlist.com
DNS Name api.getpocket.com

Public Key Info

Algorithm RSA
Key Size 2048
Exponent 65537
Modulus 98:EC:74:12:DA:E3:35:DA:79:4A:EC:68:74:99:A4:A8:E9:49:E4:F2:9B:F4:94:2A:7D:...

Miscellaneous

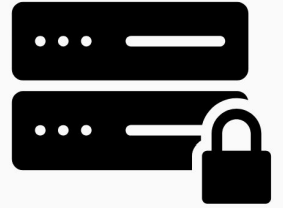
Serial Number 0E:83:4D:9F:38:A0:D9:5A:AA:50:25:7B:C6:98:00:27
Signature Algorithm SHA-256 with RSA Encryption
Version 3

RSA Encrypted (SHA-256) with key size = 2048

The Handshake



Client says hello



The Handshake



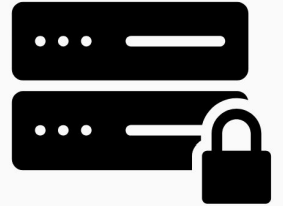
- Server hello
- Client certificate request

The Handshake



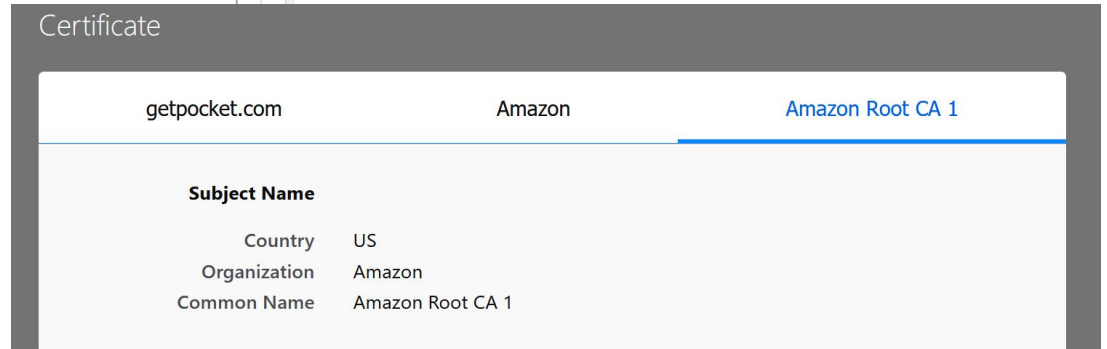
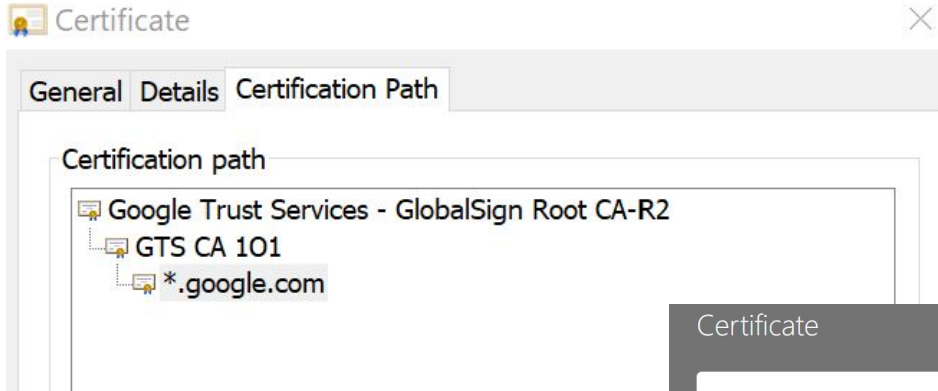
- Client certificate
- Client sends key info (encrypted with server's public key)
- Certificate verify (with digital signature)
- Finished message (encrypted with symmetric key)

The Handshake



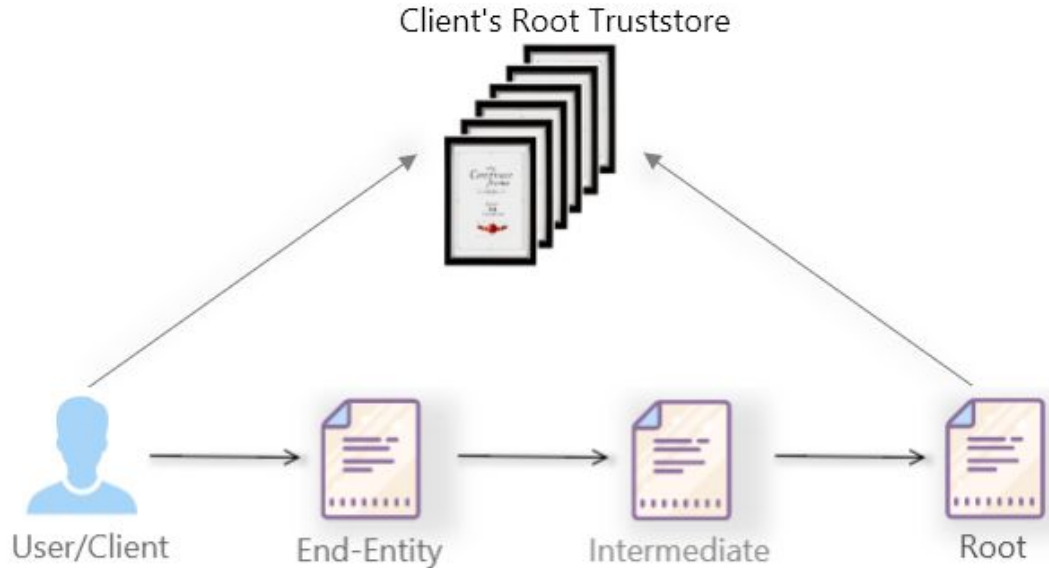
Finished message (encrypted with symmetric key)

Chain of Trust

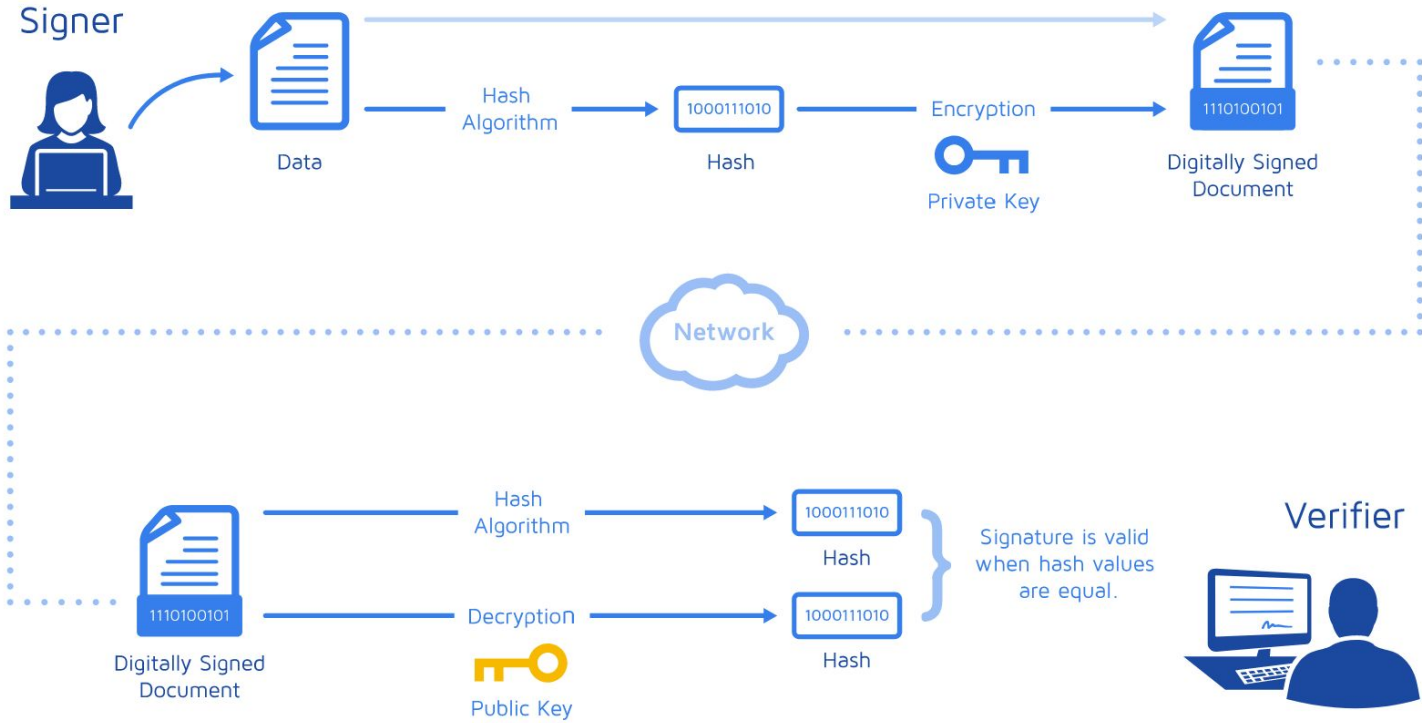


Certificate Authority (CA)

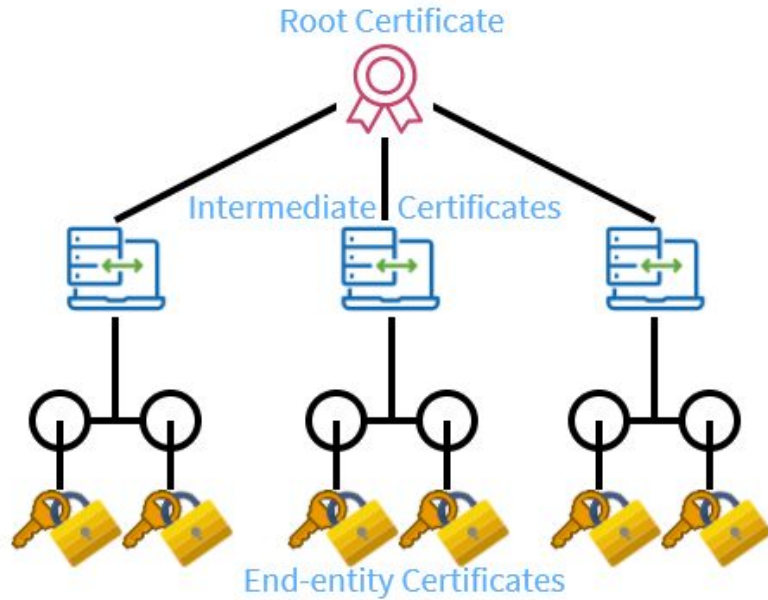
A company or organization that acts to validate the identities of entities and bind them to cryptographic keys through the issuance of digital certificates.



Digital Signatures & Root Certificates



Certification Path



- The hierarchy:
Website certificate - Intermediate CA certificate - Root CA certificate
- Multiple certification paths could exist - could lead to errors

Certificate Errors

The image displays four overlapping browser security warning dialog boxes:

- Firefox:** A yellow warning icon and a lock icon are next to the text "Warning: Potential Security Risk Ahead". Below it, the Firefox logo is visible. The text reads: "Firefox detected a potential security threat and did not continue to self-signed.badssl.com. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details."
- Opera:** A red circle with a white 'O' icon is next to the text "Invalid certificate". Below it, the Opera logo is visible. The text reads: "Opera cannot verify the identity of the server 'www.facebook.com', due to a certificate problem. The server could be trying to trick you. Would you like to continue to the server?". Buttons for "Show certificate", "Continue Anyway", and "Cancel" are at the bottom.
- Edge:** A red shield with a white 'X' icon is next to the text "There is a problem with this website's security certificate." Below it, the Edge logo is visible. The text reads: "The security certificate presented by this website was not issued by a trusted certificate authority. The security certificate presented by this website has expired or is not yet valid. The security certificate presented by this website was issued for a different website's address. Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server. We recommend that you close this webpage and do not continue to this website." Below this are three links: "Click here to close this webpage.", "Continue to this website (not recommended).", and "More information".
- Chrome:** A red padlock with a white 'X' icon and the Chrome logo are visible. The text reads: "Your connection is not private. Attackers might be trying to steal your information from google.com (for example, passwords, messages, or credit cards)." A "Reload" button is at the bottom right.
- Safari:** A padlock icon and the Safari logo are visible. The text reads: "Safari is using an encrypted connection to macreports.com. Encryption with a digital certificate keeps information private as it's sent to or from the https website macreports.com." A "Show Certificate" button and an "OK" button are at the bottom.

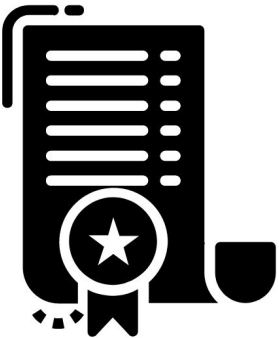
The Heartbleed Bug

- In March 2014, Google discovered a programming mistake in the popular OpenSSL library's implementation of the TLS Heartbeat Extension.
- Allows attackers to read sensitive memory from vulnerable servers, potentially including cryptographic keys, login credentials, and other private data.
- Recovery:
Patching, revocation of the keys, reissuing keys and replacing certificates.
- Lesson:
Support for critical projects;
Develop a method for scalable revocation that can gracefully accommodate mass revocation events;
Vulnerability disclosure;
Notification and patching;



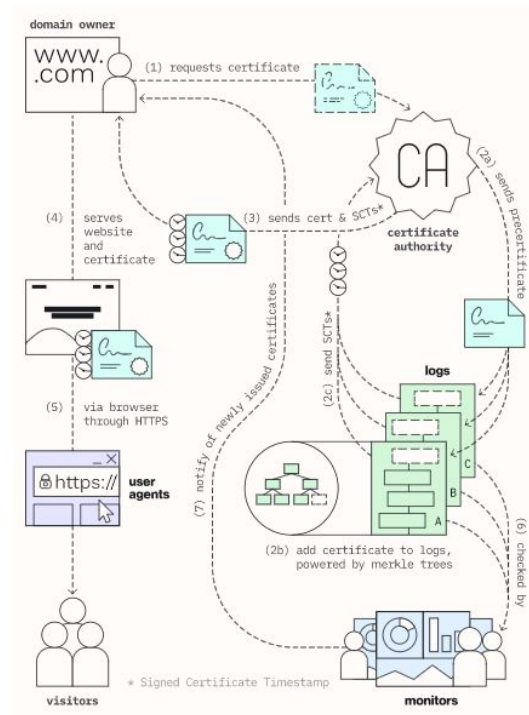
Certificate Rotation

- The replacement of existing certificates with new ones
Happens when:
 1. Any certificate expires.
 2. A new CA authority is substituted for the old; thus requiring a replacement root certificate for the cluster.
 3. New or modified constraints need to be imposed on one or more certificates.
 4. A security breach has occurred, such that existing certificate-chains can no longer be trusted.
- Example:
Internal certificate rotation within a company: use of thumbprints vs subject name



Certificate Transparency

- Used for monitoring and auditing digital certificates
- Steps:
 - Website owner requests a certificate from the CA
 - CA issues a precertificate
 - CA sends precertificates to logs
 - Precertificates are added to the logs
 - Logs returns signed certificate timestamps (SCTs) to the CA
 - CAs send the certificate to the domain owner
 - Browsers and user agents help keep the web secure
 - Logs are cryptographically monitored





Thanks for
coming to
section!