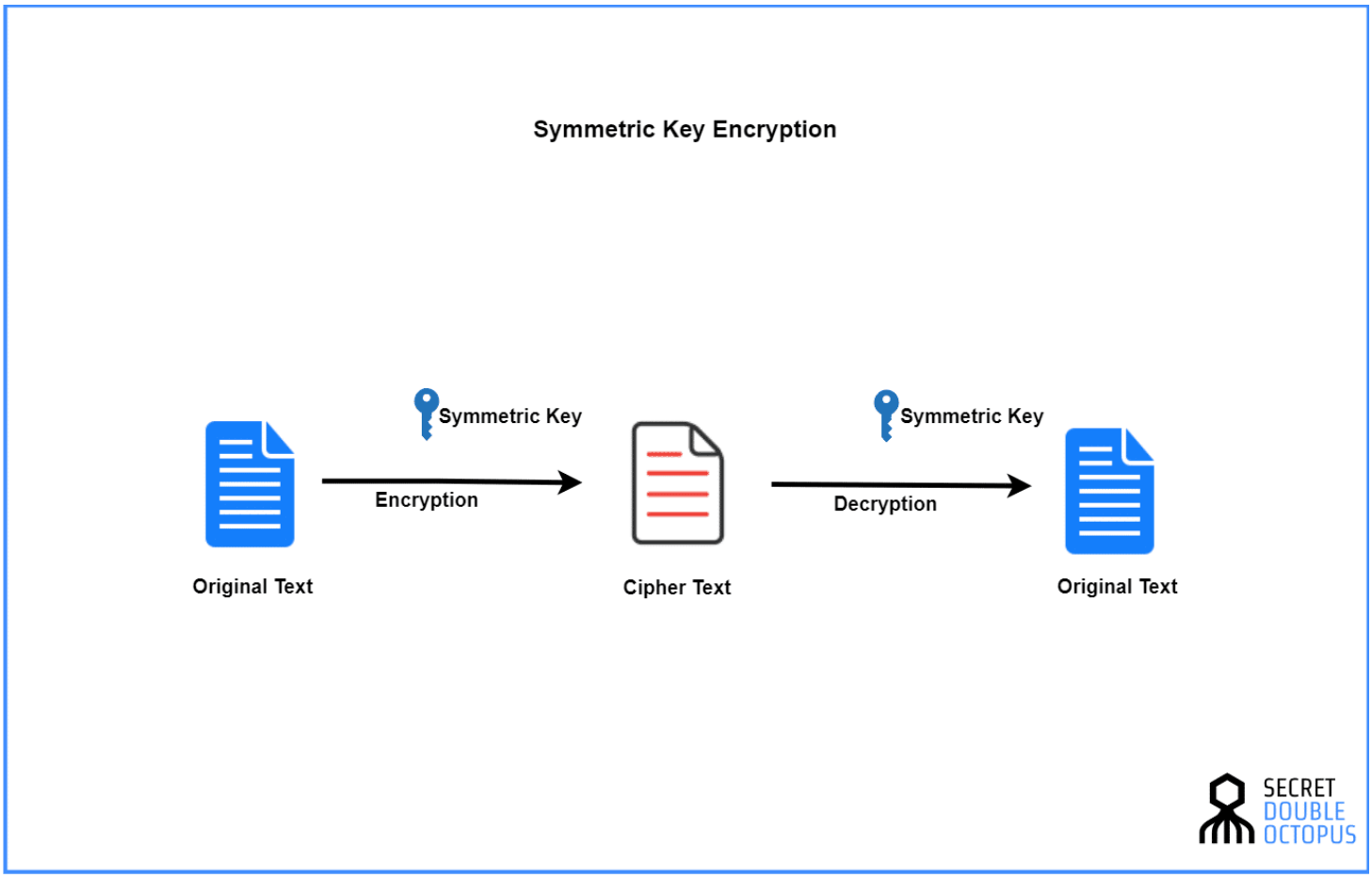
Data Encryptions and Authentications

Symmetric Encryption



Purpose

Two parties communicate each other privately with a common secret key

Usages

- Customers go to Banking website, or any other HTTPS website
- Any secure channel communications: file transfer, military communications

Police radio actually is not encrypted, so anyone can listen.

Encryption algorithms

- DES and 3DES are obsolete/insecure.
- AES128 and AES256 are popular

  Coursara class for AES

  Intel processor has hardware implementation of AES

Asymmetric encryption

Purpose

Two parties to communicate each other privately, even without common secret key.

Usage

Unless there is a private venue to communicate a secret, such as home wifi password, any customer goes to a bank website, will not have a secret key with the bank.

So a public / private key mechanism allows you to communicate each other privately (all the while other people are listening) to exchange a common secret, which can be used for further data exchange with symmetric encryption.

Asymmetric encryption is resource intensive, usually is only used for key exchange, to establish a shared secret "encryption key".

Another usage of asymmetric encryptions is for digital signature, to confirm the message comes from the stated sender, and the message is not altered in transit – more details in **Digital Signature** section.

Math Premise

- Square root is an order of magnitude more complex than multiplications
- Number theory: you can find 2 large numbers e and d, such that, for any integer m 0 <= m < n (if n=1024, m is any number between 0 to 1024, aka, 10 bit binary number)

  $((m^e) \pmod n)^d \pmod n == m$

Algorithms

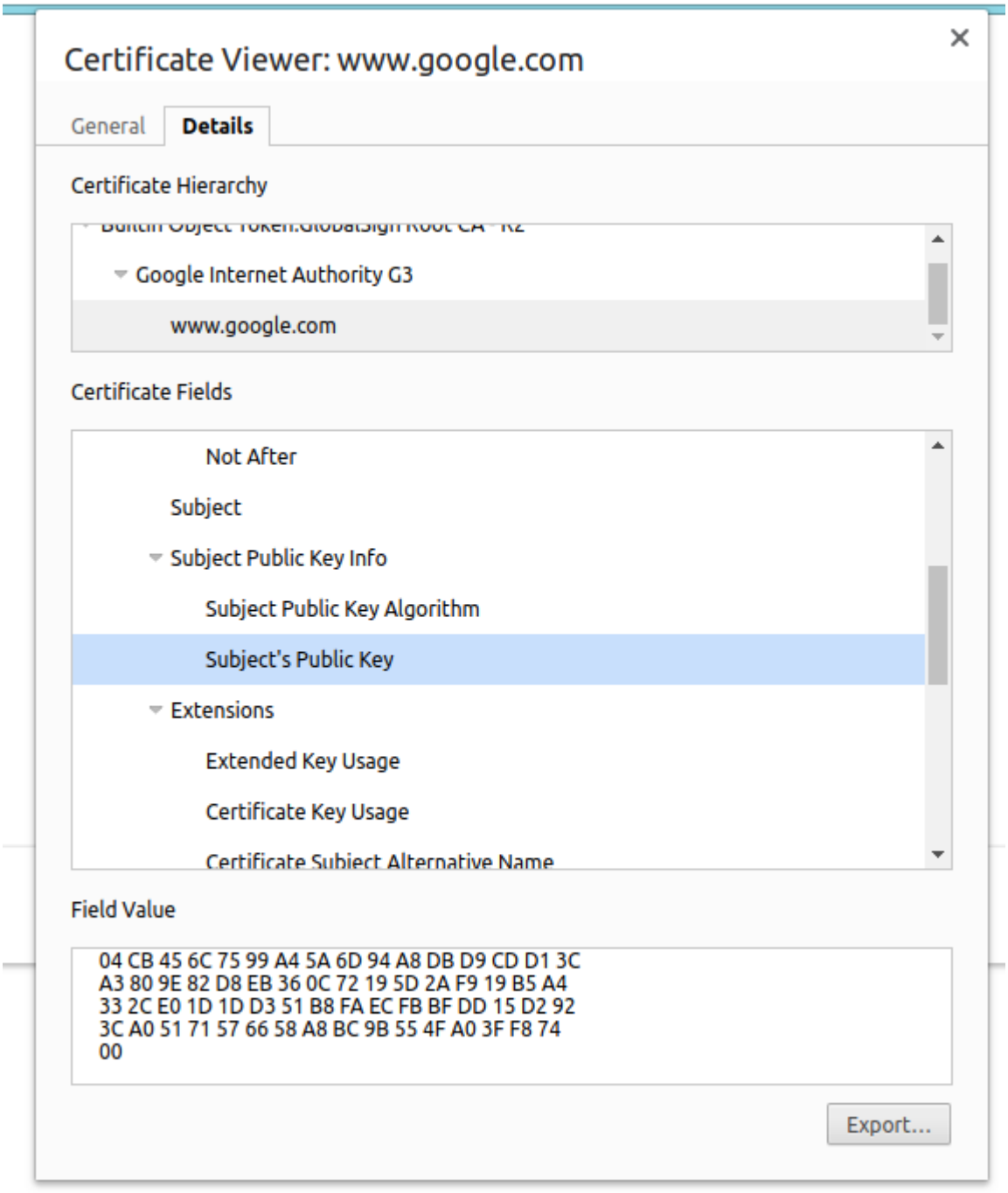
RSA

Choose two primes p and q and let $n=pq$.

1. Let $e \in \mathbf{Z}$ be positive such that $\gcd\left(e, \varphi(n)\right) = 1$.
2. Compute a value for $d \in \mathbf{Z}$ such that $de \equiv 1 (\mathrm{mod} \varphi(n))$.
3. Our public key is the pair (n, e) and our private key is the triple (p, q, d).
4. For any non-zero integer $m < n$, encrypt $\boldsymbol{m}$ using $c \equiv m^e \pmod{n}$.
5. Decrypt $\boldsymbol{c}$ using $m \equiv c^d \pmod{n}$.

(m : original plain text message; c: cipher text, encrypted message)

Sample public key size



- Web client will get public key from a website
- Web client will come up with a secrete key **m**,
- Client compute and send: $(m^e) \pmod n \Rightarrow c$
- Web server compute $(c^d) \pmod n \Rightarrow m$

Diffie-Hellman

The basic idea works like this:

1. I come up with two prime numbers **g** and **p** and tell you what they are.
2. You then pick a secret number (**a**), but you don't tell anyone. Instead you compute $\mathbf{g}^{\mathbf{a}}$ *mod* **p** and send that result back to me. (We'll call that **A** since it came from **a**).
3. I do the same thing, but we'll call my secret number **b** and the computed number **B**. So I compute $\mathbf{g}^{\mathbf{b}}$ *mod* **p** and send you the result (called "**B**")
4. Now, you take the number I sent you and do the exact same operation with *it*. So that's $\mathbf{B}^{\mathbf{a}}$ *mod* **p**.
5. I do the same operation with the result you sent me, so: $\mathbf{A}^{\mathbf{b}}$ *mod* **p**.

The "magic" here is that the answer I get at step 5 is *the same number* you got at step 4. Now it's not really magic, it's just math, and it comes down to a fancy property of modulo exponents. Specifically:

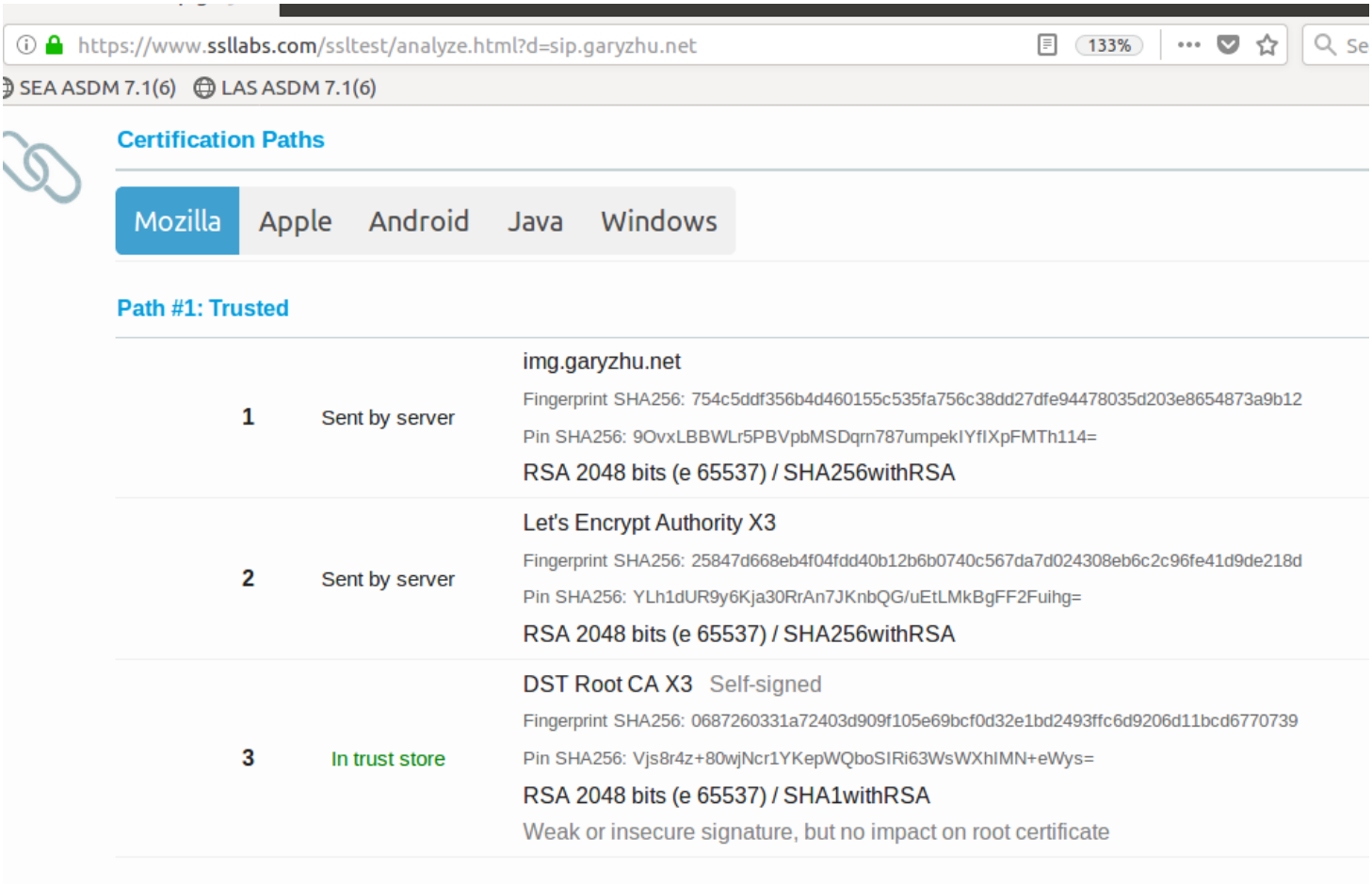
$$(\mathbf{g}^{\mathbf{a}} \textit{ mod } \mathbf{p})^{\mathbf{b}} \textit{ mod } \mathbf{p} = \mathbf{g}^{\mathbf{ab}} \textit{ mod } \mathbf{p}$$
$$(\mathbf{g}^{\mathbf{b}} \textit{ mod } \mathbf{p})^{\mathbf{a}} \textit{ mod } \mathbf{p} = \mathbf{g}^{\mathbf{ba}} \textit{ mod } \mathbf{p}$$

Public key infrastructure

A set of of root certificates from trusted Certificate Authorities are pre-installed in OS, devices.

A website certificate should be signed by a chain certificates with the root certificate being trusted and pre-installed in client system.

An example of a certificate chain to root certificate “DST Root CA X3”



Exceptions

Many cloud companies will create public/private key pair for each customer, especially for secure API calls. The website is different in that it has one pair of public/private keys and used for all clients/browsers.

Hash

Purpose

An algorithm to convert any length data into a fixed length of data.

In the realm of security, the algorithm should be made to be very difficult (or impossible) to reverse the function to re-construct original data; the algorithm should also minimize collisions, i.e., many different data are converted to the same hash value.

Usages

One-way password conversion

User's password is stored in database as SHA-2 hash values, when a user logs in, inputed password would be converted to SHA-2 hash, and verify it with the stored hash value.

The requirement of "hard to reverse", so even database with stored password hash values are stolen, attackers would not be able to re-construct original password; the requirement of minimum collision is to avoid mis-typed password, or wrongly guessed password happens to have the same hash value, and hence considered password match. Even for an algorithm with minimum collision, a common practice for password hash is to add 'salt', a random string that is hashed along with the password string, so even the same password string, would be hashed into a different value, because of different salt.

Message integrity and authentications

Sending over original data, along with computed hash values, the receiver can do the same hash computation and compare the hash value to make sure there is no data loss/corruption in transit.

The hashed value can further be 'signed' by the sender, the receiver can verify the signature to make sure the message is un-altered, and it is indeed from the sender – more in **Message Authentication Code** section.

Algorithms

- MD5, SHA-1 proven weak, not recommended
- SHA-2 current/popular standard; SHA-3 is a new standard.

Message Authentication Code (MAC) and Digital Signature

Purpose

While sending the original message, add a short piece of information (MAC or signature), to confirm that the message came from the stated sender (its authenticity) and has not been changed.

MAC is computed with a pre-shared key (known to both parties), so the receiver who also has the key can decrypt and verify the message.

Digital signature is also to assure integrity (message is not altered) and authenticity (from the stated sender), but without common secret keys. In this case, the sender would use private key to encrypt

the messages, while the receivers would look for senders public key to decrypt and verify the message.

MAC and Signature alone do not provide privacy. Sometimes, especially while using Digital Signature, the message is intended for everyone to see.

Usages

- *Hypothetical:*
  - an emperor is issuing a decree, he wants everyone to read it, he does not want anyone to modify it or even create a fake one.
  - the emperor uses his private key to “sign over the entire message” (detailed algorithm below), attaching the signature at the end.
  - People can read the original decree, people can use emperor’s public key and the decree (text) to verify that signature is valid.
  - Any modification of original decree or wrong private key generated signature would cause failure in signature verification, hence the integrity and authenticity of the emperor’s decree is preserved.
- Over simplified security description of Bitcoin:
  - Person S wants to send person R 1 bitcoin,
  - S will write a public note, saying I am giving person R 1 bitcoin, and then use his private key to sign it.
  - This note along with the signature (which is verified with person S’s public key) is published to public ledger.
  - Nobody can create a fake transaction, or change the amount in the note.

Algorithms

To guarantee that message is authenticated and not modified, certain encryption is used, so that nobody else can create that encrypted data except for the ones who knows either pre-shared encryption key or sender’s private key, and decryption with pre-shared encryption key or sender’s public key is used for verification

As the messages are in variable length, could be very long, and you want to have a fixed length MAC or signature. So hash function is used, which creates a fixed length of message from any length of message.

The combination of these is “Cryptographic hash function”, some common algorithms are HMAC-SHA256 or HMAC-SHA3