Part 1. Digital Signatures (25 points). Suppose that a principal Alice (A) signs a message/record (R) that is received by another party Bob (B), who validates the signature. Suppose R is sent over an insecure network connection and that malicious Mark (M) can intercept R as it transits the network from A to B. Are the following statements True or False? Indicate your answer in the box at the left of each statement. You may comment further in the space on the right if you wish.

(a) As R’s content grows in size, the size of its signature also grows.
(b) As R’s content grows in size, the cost to sign or validate R also grows.
(c) M must obtain B’s public key in order to read (or decrypt) the content of R.
(d) B must obtain A’s public key in order to validate the signature of R.
(e) M must obtain A’s private key in order to modify the content of R without detection.

Part 2. Use Cases (40 points). Consider examples of such signed records R (as in Part 1) from systems we discussed.

(a) List four examples. Number them. Use names/keywords for the systems and records: don’t explain how or why they generate or sign the records.

(b) For which of your numbered examples in (a) does the signed R contain the public key of A or B (or a hash of the key)?

(c) For which of your numbered examples in (a) does R contain the public key (or its hash) of another principal C, other than A or B?

(d) For each of the examples in (c), what other information does R contain about C?
Part 3. Blockchain (35 points). These questions apply to a blockchain system: a set of nodes organized in a broadcast network to maintain a global ledger of operations. You may presume that the blockchain functions like the example we discussed: Bitcoin. The ledger is broken into blocks; each node accepts the longest chain of valid blocks (that it knows of) as the current ledger.

(a) On average, how many nonces must a miner try before it finds a nonce that yields a block hash that begins with 4 leading zero bits?

1/16 probability that any random nonce qualifies \(\rightarrow\) 16 trials on average.
The key here is to understand how the difficulty of the puzzle grows as we add bits to the zero prefix.

(b) How many if the block hash must begin with 8 leading zero bits?

1/256 probability that any random nonce qualifies \(\rightarrow\) 256 trials on average.

There is no alternative to brute force. The simplest is to try candidate nonces in a fixed pattern (e.g., increment by 1) to avoid trying the same candidate twice. The only way to speed it up is to add more compute power.

(c) Propose a strategy for the miner to select the candidate nonces to maximize its chances of finding a “good” one quickly.

If the miner controls 10% of the computing (mining) power in the network, what are its chances of succeeding to “mine” the next block in the chain?

10%. The probability of winning is proportional to the rate at which the miner generates candidate nonces. The competitors control 1 – 10% == 90%, so this miner generates only 10% of the candidate nonces.

(e) How does a node in the network know the order of the blocks in the longest chain?

Each block has a hash pointer to its predecessor. The question presumes that the node knows the blocks. It is just asking about the order.

(f) How does a node in the network know the order of operations in the ledger?

The ledger is an ordered sequence of blocks (per (e)), and each block has a sequence of transactions in an ordered array of slots.

(g) A malicious miner \(M\) could overwrite portions of the ledger history, if it can produce a new valid block chain that is longer than any other valid chain. What prevents this attack from succeeding?

If any block \(B\) is overwritten/replaced, then every successor of \(B\) must also be replaced, since any change invalidates all successor hashes. And every replaced \(B\) must have a freshly computed “good” nonce, or else other nodes reject \(B\) and its successors. If all the new blocks are good, then others accept the replacement chain, but only if it is longer than the current chain, which grows all the time as the attack is in progress. So the attacker must “outrun” the rest of the network: it is either very lucky or it controls a majority of compute power.
Part 4. DNS (35 points). Google Public DNS (GPDNS) is a massive-scale DNS resolver that is open for use by any DNS client. Clients of GPDNS request lookups for complete domain names (e.g., www.child.parent.net), and the resolver returns DNS response records for the requested domain name.

(a) A lookup request for parent.net returns a DNS server IP address (call it P) that is authoritative for the domain parent.net. How does GPDNS determine the value of P?

(b) A lookup for child.parent.net returns “no such domain”. How does GPDNS determine there is no such domain?

(c) Google says GPDNS responds faster (on average) than a local resolver, in part because GPDNS has many clients, including Google’s search engines. How does more client activity help GPDNS serve its clients faster?

(d) GPDNS uses DNSSEC to validate lookups since 2013. How many decrypt operations does the resolver perform to validate that there is no domain child.parent.net?

(e) How does GPDNS obtain the public key for the DNS server (P) that is authoritative for parent.net?

(f) GPDNS uses a sharding scheme internally. It has the property that two lookup requests for the same domain name are likely to be handled by the same server. What is the benefit of this property?

(g) The GPDNS sharding has the property that two lookups for different domain names are likely to be handled by different servers. What is the benefit of this property?

---

Ask .net, i.e., send a DNS lookup to the authoritative DNS server for top-level domain “.net”. The lookup responds with P.

Send a DNS lookup request to P, which authoritative for the parent domain parent.net. If the domain has no node or subdomain named “child”, the server returns the error “no such domain”.

Caching. More clients send more requests, so GPDNS can keep its name lookup caches warm. So a larger share of requests are cache hits and save the lookup cost.

I accepted 2 or 3. Root signs “.net” and “.net” signs “parent.net”. So that’s two decrypt operations to check those signatures. And “parent.net” signs the “no such domain” response.

For DNSSEC the “.net” server includes the public key (or hash) for P in its lookup reply (per (a)). Some answers talked about certificates and such, apparently confusing DNSSEC with SSL/TLS.

More effective caching. Requests for the same domains concentrate on the same servers, leading to more cache hits: if A looks up D and then B looks up D, B’s lookup likely finds the cached response from A. It was not enough to say “more efficient”.

Scale. Spreads the request load across more servers, without compromising cache performance. I accepted “load balancing” or other indications that you understood.
Part 5. Security of Google Public DNS (60 points). Google contends that GPDNS is more secure than other DNS resolvers because their infrastructure is secure and implements best practices (including DNSSEC). Even so, Google suggests that clients communicate with GPDNS over a secure (SSL/TLS) connection to further protect DNS responses.

(a) For an SSL/TLS connection as explained in class (e.g., for HTTPS), how does the client obtain the session key to use with the connection? (10)

(b) How does the server obtain the session key to use with the secure connection? (10)

(c) How does this use of SSL/TLS to connect to GPDNS offer additional security for its users? To answer, summarize an attack that it prevents. (20)

(d) Is it secure yet? Summarize any other attack(s) that a client might be vulnerable to, relating specifically to its use of GPDNS. (20)

The client just makes it up. It could be any random number.

The client sends it to the server, encrypted with the server’s public key, so that nobody else can obtain it.

Man-in-the-middle. An attacker M on the network could attempt to redirect DNS lookups to an attack server that masquerades as GPDNS, or to modify responses in flight. But an SSL connection precludes this attack because M lacks the session key, and so cannot read or modify the requests or responses.

This is a little bit of a thought question. Here as in 4(e) there was some confusion between the end-to-end protection of SSL/TLS with an HTTP web server and the use of SSL/TLS to protect communication for DNS lookups.

1. Google could spy on all of your DNS lookups to see what content you are accessing across the whole Internet. This is the most likely attack, but few mentioned it. No slur against Google here: it’s legal and it’s just business.

2. Google or GPDNS admins could be evil or “go rogue” (or get hacked) and return bad responses.

3. Various attacks could “break” SSL and allow M of part(c) to complete an attack: a compromised or rogue CA issues bogus certificates for M to masquerade as GPDNS, or Google loses its private key and M “finds” it.

4. Denial of service attacks, etc.