Question 1 ***DNSSEC / TLS*** **(15 min)**

(a) Oski wants to securely communicate with CalBears.com using TLS. Which of the following entities must Oski trust in order to communicate with confidentiality, integrity, and authenticity?

1. The operators of CalBears.com
2. Oski's computer
3. Cryptographic algorithms
4. Computers on Oski's local network
5. The operators of CalBears.com's authoritative DNS servers
6. The entire network between Oski and CalBears.com
7. CalBears.com's CA
8. All of the CAs that come configured into Oski's browser
9. All of the CAs that come configured into CalBears.com's software
10. The operators of `.com`'s Authoritative DNS servers
11. The operators of the Authoritative DNS root servers

(b) Suppose we didn't want to trust any of the existing CAs, but DNSSEC was widely deployed and we were willing to trust DNSSEC and the operators of the root zone and of `.com`. How could TLS be modified, to avoid the need to trust any of the existing CAs, under these conditions?

(c) Assume end-to-end DNSSEC deployment as well as full deployment of your change. Oski wants to securely communicate with CalBears.com using TLS. What changes are there to the list in part A (i.e., what must Oski trust in order to communicate with confidentiality, integrity, and authenticity)?

(d) Is this change good or bad? List at least one positive and one negative effect that would result from this change.

Question 2 ***NSEC*** **(20 min)**

In class, you learned about DNSSEC, which uses certificate-style authentication for DNS results.

(a) In the case of a negative result (the name requested doesn't exist), what is the result returned by the nameserver to avoid dynamically signing a statement such as "`aaa.google.com` does not exist"? (This should be a review from lecture.)

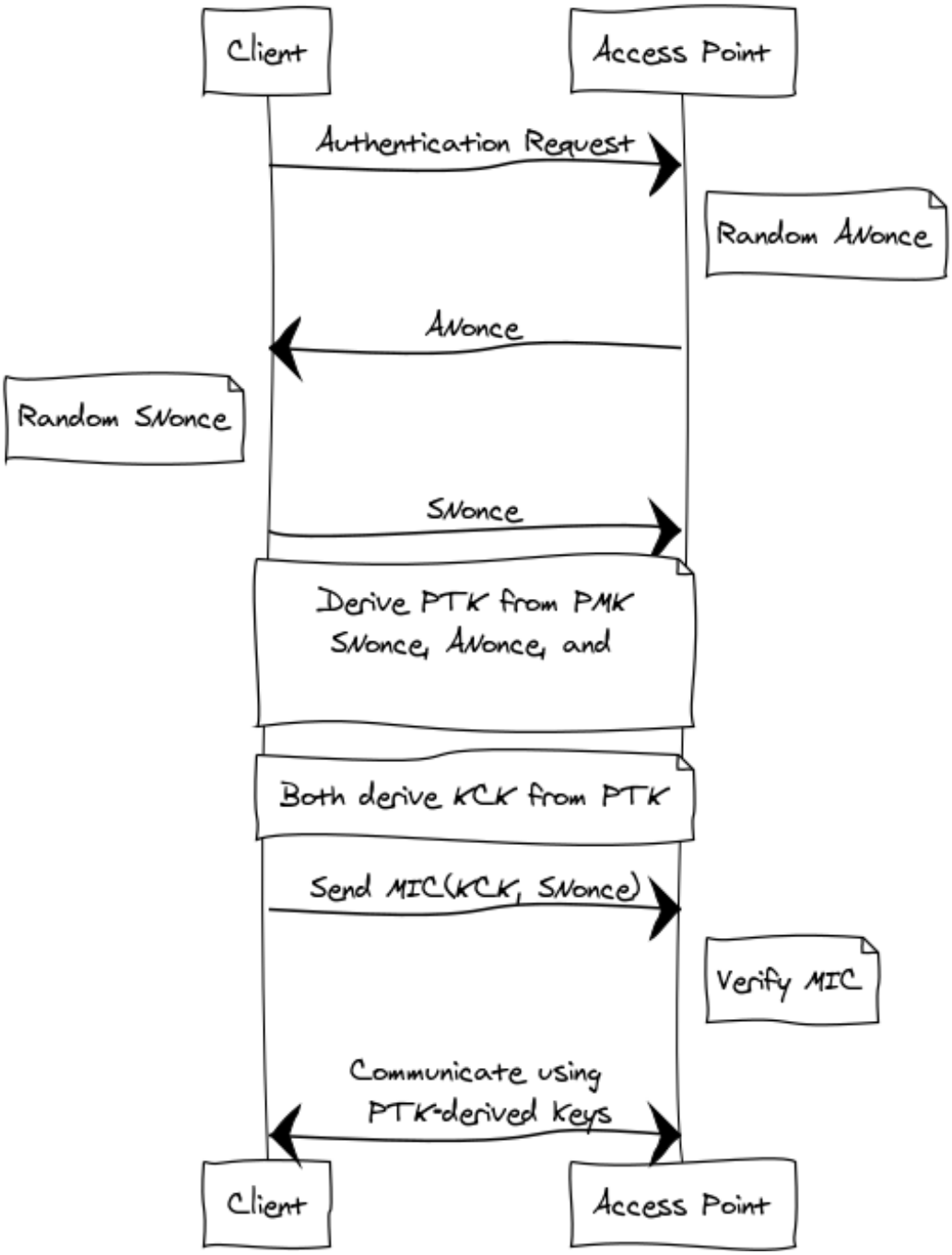
(b) One drawback with this approach is that an attacker can now enumerate all the record names in a zone. Why might this be a security concern?

(c) Louis proposes to modify NSEC as follows. First, the site operator will take a hash of each domain that does exist. Then, the site operator proceeds as in NSEC: they sort the hashes and sign each adjacent pair. How can this be used to provide authenticated denial? How does this help mitigate enumeration attacks?

Question 3 ***WPA2*** **(15 min)**

Let's review WPA2. You might find some of the definitions below helpful.

- PMK is the *premaster key*, also known as "the WiFi password".
- PTK is the *pairwise transient key*, which is used to derive symmetric keys.
- KCK is the *key confirmation key*, which helps the client and the access point confirm they've agreed on the same keys.



(a) Louis Reasoner proposes that we don't generate ANonce or SNonce, and instead derive the PTK directly from the SSID and PMK. What sort of attack does this fail to prevent?

(b) WPA2 has an interesting pattern which is common in cryptographic protocols. Both parties agree on a shared secret, which they use to derive keys. Which other protocol have we seen which follows this motif?

(c) Alyssa P. Hacker wants to compromise a WPA2 WiFi network. In order to do so, she performs the handshake many times. She bruteforces possible PMK against the Access Point many times, until the access point eventually accepts it. If the password has 28 bits of entropy[1] and the attacker can make 10 guesses a second, how long will it take to bruteforce the password?

(d) Ben Bitdiddle has an alternate idea. Ben waits until Louis attempts to connect to the network. While this happens, he records all of the messages that Louis sends over the network. How can Ben use this to bruteforce possible PMKs? Why do we expect this to be faster than Alyssa's method?

[1] As per this XKCD comic, a password which looks like `Tr0ub4dor&3` has roughly 28 bits of entropy.