Recall

Classical cryptography

One-time pad

Block ciphers

Cipher modes and MACs

Symmetric-key cryptography

Can demonstrate semantic security for:

- One-time pad (perfectly secure **iff** key symbols unpredictable)
- Block ciphers (with appropriate modes)
- Stream ciphers (inspired by one-time pad)

but... how to share the key?

- *key exchange problem* meant crypto only really for global orgs

Up until the 1970s, in order to really make use of cryptography, you needed to have a mechanism to reliably and confidentially distribute keying material to all of the places it might be used. This kept the tool of full-strength cryptography out of the hands of all but large and powerful organizations with global networks, i.e., governments and some multi-national corporations. And, of course, multinationals don't have diplomatic pouches!

Diffie and Hellman*

How to solve key exchange problem?

Focused on *discrete logarithm* problem:

$$Y = \alpha^X \pmod{q}$$
$$X = \log_\alpha Y \pmod{q}$$

"Normal" logarithms are easy... *discrete* log $\pmod{q}$ tougher!

Example of mathematical one-way function

* Diffie and Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory* 22(6), 1976.
DOI: 10.1109/TIT.1976.1055638

Solving this problem could make cryptography much more useful, both for "ordinary" people and for traditional users of cryptography who wouldn't have to expend so much effort or limit its use quite so much.

A one-way function is one that is relatively easy to compute in one direction (in this case, raising a value to an exponent in a finite field) and extraordinarily difficult to compute in the other direction (in this case, finding the logarithm of α^X). How can we use these kinds of one-way functions for cryptographic ends?

Diffie-Hellman operations

1. Raise α to random numbers $X_A, X_B \pmod{q}$:

$$Y_A = \alpha^{X_A} \pmod{q}$$
$$Y_B = \alpha^{X_B} \pmod{q}$$

2. Raise results to $X_A, X_B \pmod{q}$:

$$Y_A{}^{X_B} \mod q = \left(\alpha^{X_A} \bmod q\right)^{X_B} \mod q = \alpha^{X_A X_B} \mod q$$

$$Y_B{}^{X_A} \mod q = \left(\alpha^{X_B} \bmod q\right)^{X_A} \mod q = \alpha^{X_A X_B} \mod q$$

Diffie and Hellman proposed a cryptographic scheme in which two subjects — conventionally referred to as ____________ and ___________ — would exchange messages _________ ________, i.e., assuming that ______________________________________.

Diffie-Hellman key exchange

$$Y_A{}^{X_B} \mod q = \alpha^{X_A X_B} \mod q$$

$$Y_B{}^{X_A} \mod q = \alpha^{X_A X_B} \mod q$$

So what?

- What if $\alpha^{X_A X_B} \mod q$ were called... K_{AB}?
- Exchange of α^{X_A} and α^{X_B} **in the clear** establishes a key
- As long as the discrete log problem is "hard", the established key can be used for symmetric-key cryptography (with **one** caveat)

In the brave new world of quantum computers, an algorithm called ________________________ can be used to efficiently compute discrete logarithms. So, straightforward Diffie-Hellman key exchange as originally proposed won't work forever — we need another hard mathematical problem. However, there are lots of _________________________ out there, some of which are more amenable than others to an environment where the adversary possesses quantum computers. Whether or not we use discrete log as our one-way function, the *idea* of Diffie-Hellman key exchange is still useful. That's why it turns up all over the place, e.g., ECDH in a TLS cipher suite.

Significance

Created a new era of *public-key cryptography*

- well, at least as far we anyone knew at the time*
- a.k.a., *asymmetric-key cryptography*
- first possiblity of cryptography for everyone

---

* P. Wayner, "British Document Outlines Early Encryption Discovery", *The New York Times*, 24 Dec 1997.

This is an example of a funny thing that often happens: pure mathematicians play around with number theory and find a result that is elegant, pure and also completely useless in practical terms. That result then sits on the shelf for years and years until someone discovers a way to ______________________ to a new problem that had nothing to do with the original mathematicians' objectives (which was do discover something elegant). This kind of progression (pure research leads to applied research which leads to practical products) happens all the time, leading to ____, and it ____________________ if you only look for things you can already think of. That's why the "little R, big D" style of R&D doesn't offer much of a future by itself.

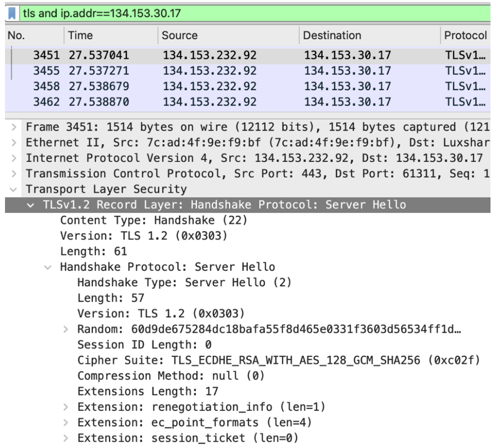
Diffie and Hellman were, unbeknownst by them, beaten to the punch by six years by cryptographers working for the UK government. C.E.S.G. Report No. 3006 was declassified by GCHQ in 1997, but annoyingly, the report doesn't include any declassification markings: it still looks like a classified document! So, depending on where you work, don't print this out and leave it lying around. 🙂

Diffie-Hellman today

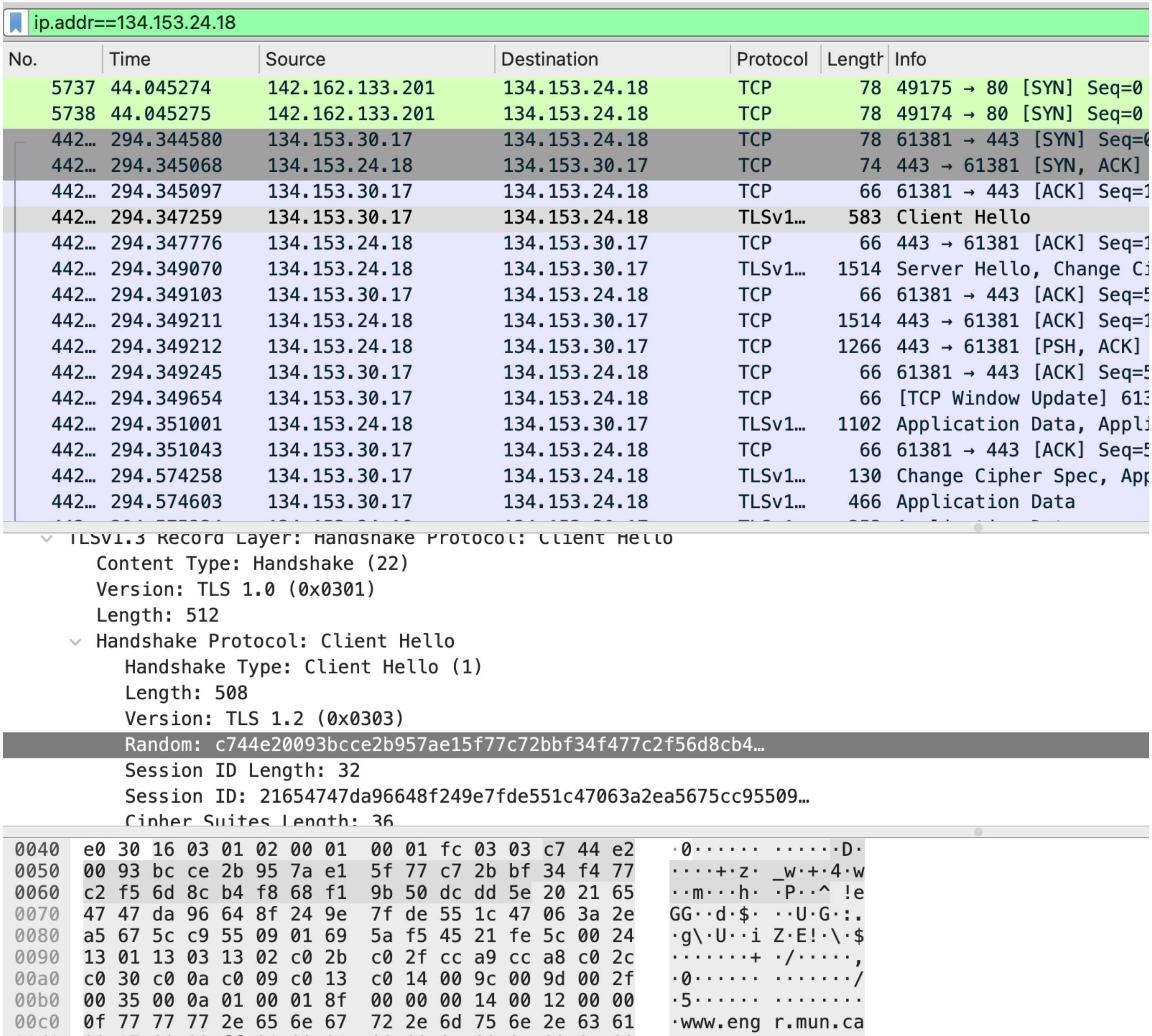
Foundation of TLS

- Used to establish symmetric keys between parties
- Inspiration for later ECDHE (will discuss in a few slides)

More TLS later...



Demo:



We'll talk more about TLS and its importance when we get to the ________________________ portion of the course next week.

RSA cryptosystem

1. Choose large primes p and q, compute $n = p \cdot q$
2. Choose b, compute a from $a \cdot b \mod \phi(n) = 1$
   - $\phi(n) = (p-1)(q-1)$ is the *Euler totient function*
   - b should be co-prime with $\phi(n)$; compute a using *extended Euclidean algorithm*
3. *Public key* $K_P = \{b, n\}$, *secret key* $K_S = \{a, p, q\}$

Rivest, Shamir, Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*. 21 (2): 120–126, 1978. DOI: 10.1145/359340.359342.

Although Diffie-Hellman came first, the most famous public-key cryptographic algorithm is RSA. Unlike Diffie-Hellman key agreement, which just lays the groundwork for subsequent use of a symmetric-key cipher, RSA really is an encryption algorithm all by itself.

RSA encryption/decryption

Given large (e.g., 2048b) plaintext P:

$$C = P^b \mod n$$

$$\begin{aligned} P &= C^a \pmod{n} \\ &= \left(P^b\right)^a \pmod{n} \\ &= P^{a \cdot b} \pmod{n} \\ &= P^{k \cdot \phi(n)+1} \pmod{n} \\ &= P \pmod{n} \text{ if } P \text{ co-prime to } n \end{aligned}$$

This is an example of a ____________________________, which adds an additional layer on top of a one-way function. Like a one-way function, a ___________________ function should be easy to compute in one direction and hard in the other. What's new here is that some __________ _______________ (in this case, the value a) can make even the hard direction easy. Whoever possesses the ____________________ can encrypt or decrypt, whereas you can only encrypt without it.

Significance of RSA encryption

Uses *asymmetric key pair*

- Can encrypt using public key
- Decryption requires knowledge of private/secret key

Much slower than symmetric-key encryption!

- Can be used to encrypt a symmetric key for bulk crypto

Security of RSA

Given b, n and $C = P^b \mod n$, find P

Difficult to invert exponentiation

- Easy if we know a
- Easy if we known p or q (which we can use to find a)

Difficult to factor n

- ... at least until quantum computing becomes "real"!

Difficult to invert exponentiation

We already know that the discrete log problem is hard: we can't just find the logarithm of $P^b \mod n$. If you find a way to do that easily, you will break both Diffie-Hellman key exchange *and* RSA encryption. And become very famous... or very rich!

Finding a multiplicative inverse in real numbers is easy: if $ab = 1$, $a = \frac{1}{b}$. However, in a finite, field, finding the inverse of $ab \mod n$ is not so easy! In fact, it's another known "hard problem" in mathematics and computing.

Difficult to factor n

If we could factor n into p and q, we could find a in the same way that the private key owner did when they generated it. However, factoring large almost-prime numbers is another known hard problem. Well, at least with conventional computers...

This is one instance in which some of the hype around quantum computing is justified. If people ever manage to build a quantum computer large enough and powerful enough, Shor's algorithm provides a method of factoring large integers in bounded-error quantum polynomial time (BQP). So, will that "break all cryptography"? ______________________________ ____________________ quantum-resistant algorithms:

- algorithm standardization
- practical experimentation

RSA factoring

RSA Factoring Challenge

- Historic contest funded by RSA Security, Inc.
- Now defunct, people still attacking challenges

Current recommendations

- 2048b RSA keys until... 2030?
- Beyond that... not RSA-2048!

Bits	Factored
330	1991
426	1994
512	1999
640	2005
768	2009
795	2019
829	2020

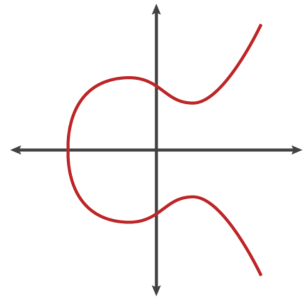
Once we get beyond 2048b RSA keys, we're not going to keep using larger and larger RSA moduli. As these numbers get bigger and bigger, it becomes more and more expensive to perform even legitimate RSA operations. Instead, the world is already moving away from RSA and towards...

Elliptic-curve cryptography

- Curve over finite field that satisfies:

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

- Relies on *point addition* (and many additions makes for multiplication)
- Can represent keys with *hundreds* of bits instead of *thousands*
- ECDH, ECES, ECDSA...



Again, this is all in a ________________, not general (real) numbers. Here, we can replace the standard discrete log problem with the __________________________________ and re-constitute the same sorts of public-key cryptographic algorithm that we've built with the regular discrete log.

One major advantage of ECC is that we can use smaller keys (hundreds of bits providing equivalent security to thousands of bits for RSA) and do less computational work.

Using elliptic curves, we can do key negotation (Elliptic Curve Diffie-Hellman), encryption (Elliptic Curve Encryption Standard) and other things (e.g., the Elliptic Curve Digital Signature Algorithm).

How about the reverse?

We know that:

- $c = E_P(p)$ requires knowledge of a public key
- $p = D_S(c)$ requires knowledge of a private key

What about:

- $s = D_S(m), m$?
- **Anyone** can check that $m = E_P(s)$!

If we have public-key cryptographic algorithms that are easy in one direction and hard in the other direction, could we reverse them so that the opposite is true?

Yes! We can apply the "decryption" operation (using the private key) in order to produce a message.

If we do this, the message will be such that anyone can check it using the public key. If it checks out, this must mean that ____. This is a bit like a Message Authentication Code, but now we don't need to have access to a secret key in order to verify the message, ________________________!

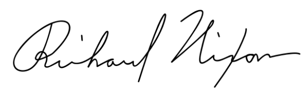
Digital signatures

- Like encrypting large plaintexts, too slow to be practical
- Instead, "decrypt" a *cryptographic hash* of a message

Used for:

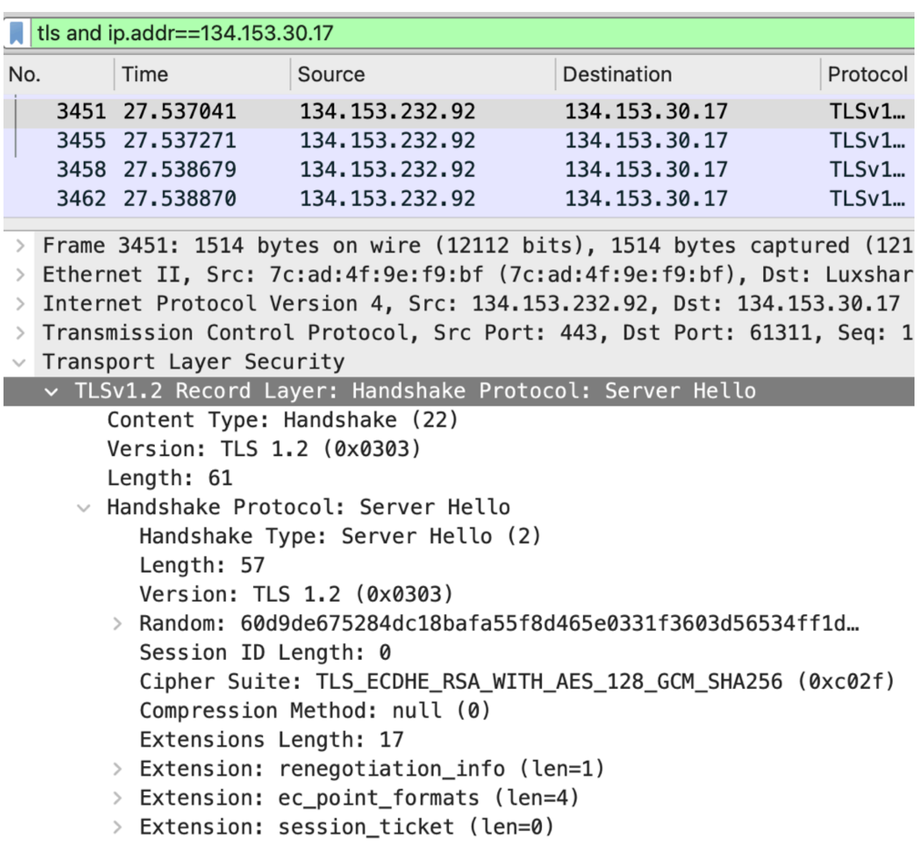
- Signing documents (a bit)
- Signing server certificates (later)
- Signing code (next time)

Digitally signed by Jonathan Anderson
Reason: I am approving this document
Date: 2021.06.16 08:02:42 -02'30'



When we use public-key cryptography to produce such a verification token, we call it a ________________________.

We never digitally sign a message directly. Instead, we compute the hash of a message (which computes a fixed-length value from an arbitrary-length message), then produce a digital signature of that hash value. Now we can understand all of the elements of a *TLS Cipher Suite*, e.g.:



- ECDHE (Elliptic Curve Diffie-Hellman Exchange) for key negotation

- RSA for digital signature of server information
- AES-128 in GCM (Galois Counter Mode) for encryption
- SHA-256 for message hashing

Summary

Diffie-Hellman

RSA

Elliptic curves

Digital signatures