

The background of the slide is a collage of architectural blueprints in various shades of blue and white. The blueprints feature technical drawings, lines, and text in multiple languages, including English, German, and Russian. The text is oriented vertically on the right side of the slide. The Cornell Tech logo is prominently displayed in the center-right, with the words 'CORNELL' and 'TECH' stacked vertically. Below the logo, the text 'HOME OF THE JACOBS INSTITUTE' is written in a smaller font. At the bottom right, there are two circular logos: the Technion logo on the left and the Cornell University logo on the right. The Technion logo is a circular seal with the word 'TECHNION' at the top, a central emblem, and the years '1912' and '1912' on the sides. The Cornell University logo is a circular seal with the words 'CORNELL UNIVERSITY' at the top and 'FOUNDED A.D. 1826' at the bottom.

HOME OF THE
JACOBS
INSTITUTE



CCA Attacks against PKE

- Ad-hoc fix to Bleichanbacher:
 - Don't leak whether padding was wrong or not
 - This is harder than it looks (timing attacks, control-flow side channel attacks, etc.)
- Better:
 - use chosen-ciphertext secure encryption
 - OAEP is common choice

Formalizing security: IND-CCA for PKE

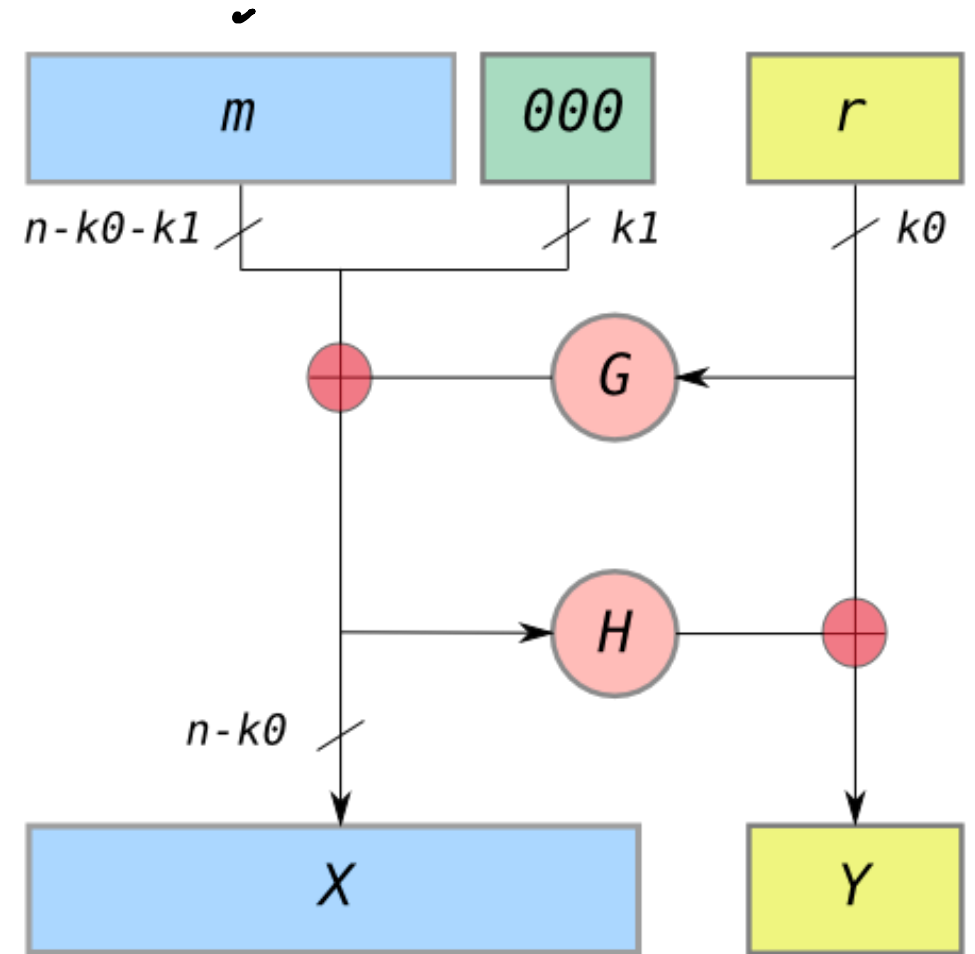
- Can formalize chosen-ciphertext attack (CCA) security for PKE: IND-CCA
- This is different than authenticity in AEAD. Why?
 - Anyone can encrypt: ciphertext forgeries trivial!
 - Combine digital signatures (stay tuned) with PKE to achieve authenticity in asymmetric setting
- Reduction showing RSA one-wayness \Rightarrow OAEP is IND-CCA [Fujisaki et al. 2001]

IND-CCA(PKE, \mathcal{A}):
 $(M_0, M_1) \leftarrow \$ \mathcal{A}$
 $(pk, sk) \leftarrow \$ Kg$; $b \leftarrow \$ \{0, 1\}$
 $C \leftarrow \$ Enc(pk, M_b)$
 $b' \leftarrow \$ \mathcal{A}^{Dec}(pk, C)$
Return $(b = b')$

Dec(C')
If $C' = C$ then Return \perp
 $M \leftarrow Dec(sk, C)$
Return M

RSA-OAEP (optimal asymmetric encryption padding)

- Provide better padding scheme than PKCS#1v1.5
- OAEP is such a padding scheme
 - r chosen randomly
 - G, H hash functions
 - $C = (X || Y)^e \bmod N$
- RSA one-wayness implies CCA security



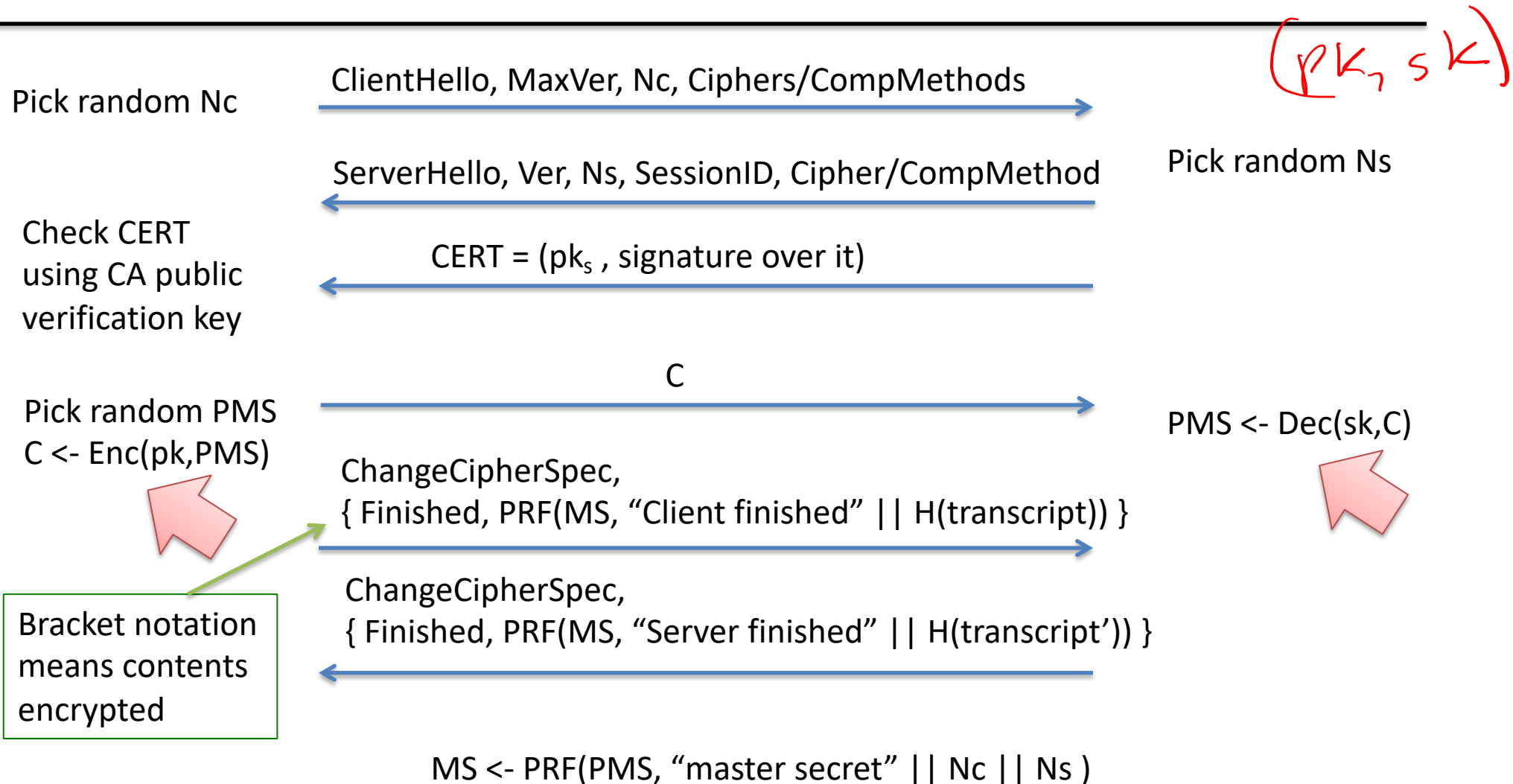


Client

TLS 1.2 handshake for RSA transport

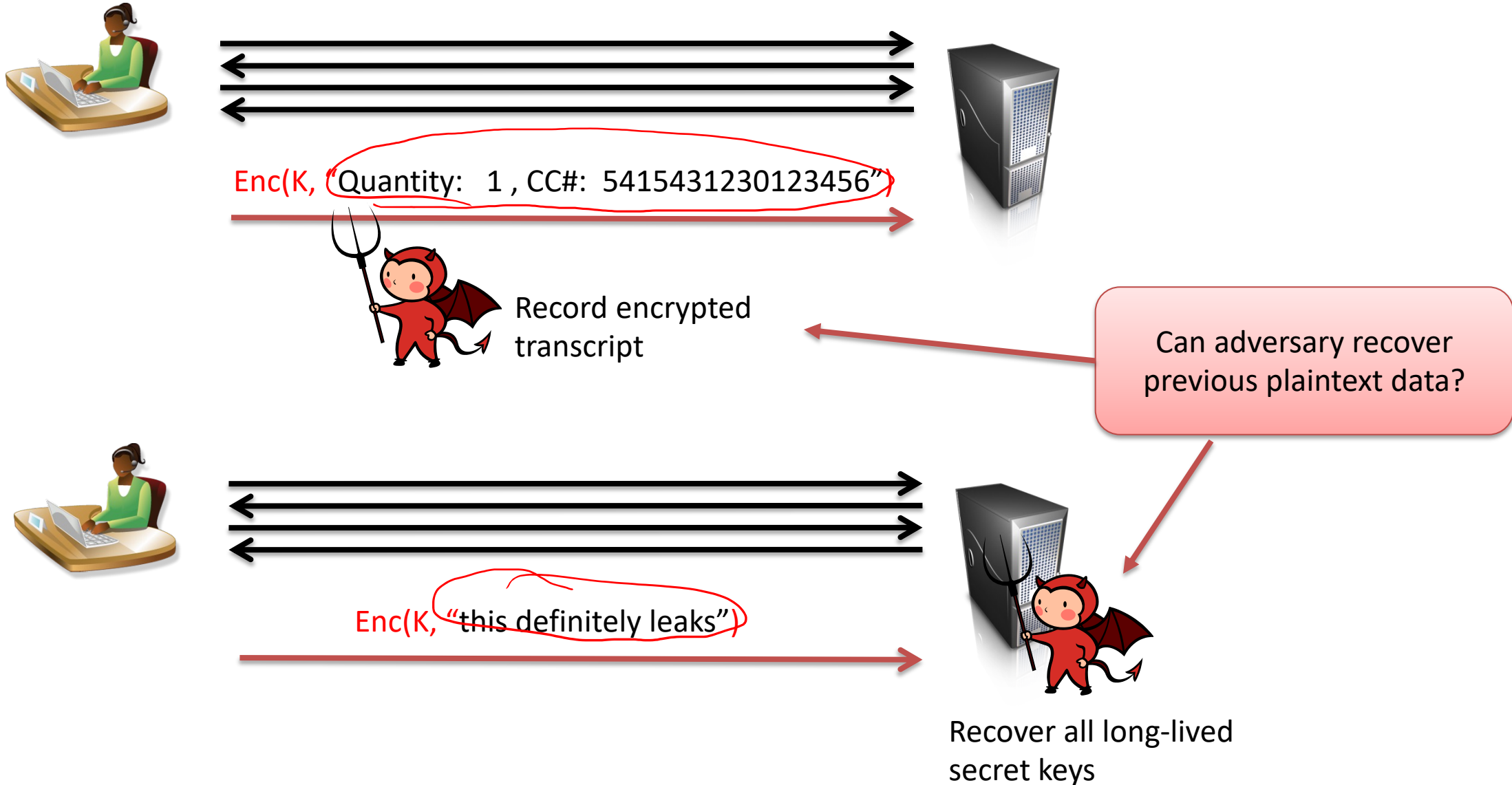


Server



Forward-secrecy

Future compromises of long-lived secrets should not enable decryption of past communications



Towards achieving forward secrecy

- Can't encrypt secret session key material under long-lived encryption key
- Need *ephemeral* secrets that can be deleted after key exchange
 - Basic recipe: generate new asymmetric key for each key exchange
 - Could use new RSA keys, but this is pretty slow (prime generation)
- Instead: Diffie-Hellman key exchange

Diffie-Hellman using integers modulo prime

Let p be a large prime number

Fix the group $G = \mathbf{Z}_p^* = \{1, 2, 3, \dots, p-1\}$

Multiplicative subgroup of finite field $\text{GF}(p)$ (which includes 0)
So often called **finite field DH**

Then G is *cyclic*. This means one can give a member $g \in G$, called the generator, such that

$$G = \{ g^0, g^1, g^2, \dots, g^{p-1} \}$$

General exponentiation notation for group; many of our protocols agnostic to exact group.
 g^i is $g^i \bmod p$ in this case

Example: $p = 7$. Is 2 or 3 a generator for \mathbf{Z}_7^* ?

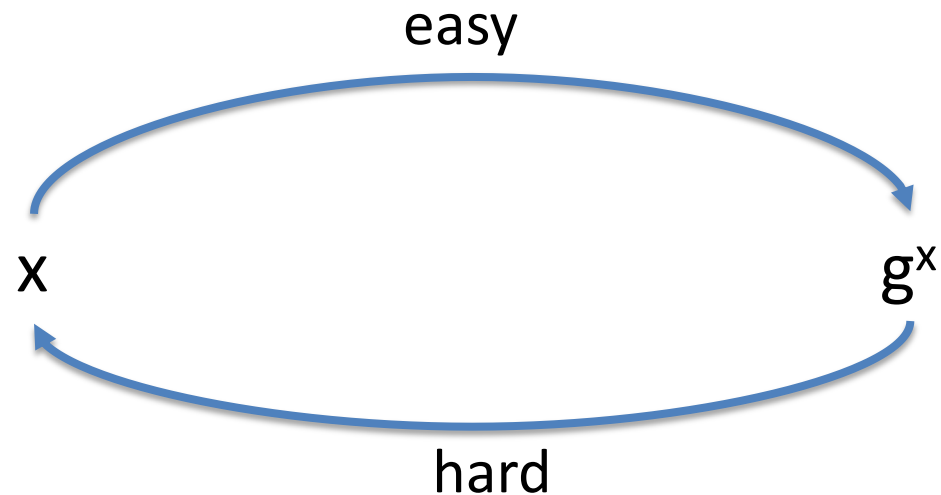
x	0	1	2	3	4	5	6
$2^x \bmod 7$	1	2	4	1	2	4	1
$3^x \bmod 7$	1	3	2	6	4	5	1

The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary g , $X = g^x$. Adversary's goal is to compute x



The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary $g, X = g^x$. Adversary's goal is to compute x

$\mathcal{A}(X)$:

```
for  $i = 2, \dots, |G|-1$  do  
    if  $X = g^i$  then  
        Return  $i$ 
```

Very slow for large groups!

$O(|G|)$

Baby-step giant-step is better:

$O(|G|^{0.5})$

Nothing faster is known for some groups.

Baby-Step Giant-Step algorithm

- DLP: Given g^x for random x , compute x

$$g^x = g^{az+b}$$
$$g^x g^{-az} = g^b$$

Rewrite x as $x = \underline{az} + \underline{b}$ with $z = \text{ceil}(p^{0.5})$

$$g^x g^{-az} = g^b$$

For $b = 1, \dots, z$

Store (b, g^b)

For $a = 1, \dots, z$

Check if $g^x g^{-az}$ equals one of precomputed g^b values

Return $az + b$

- Works in time $O(p^{0.5})$ and space $O(p^{0.5})$
- Pollard rho method: reduce space to constant

Better than Baby-Step, Giant Step?

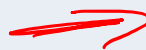
- If prime factorization of group order $p-1$ is “smooth” (all primes in factorization relatively small), can use Pohlig-Hellman algorithm:

$$\prod_i \underline{p_i^{e_i}} = p-1 \quad \text{All } p_i \text{ are primes}$$

– Run time is $\mathcal{O} \left(\sum_i \underline{e_i (\log (p - 1) + \sqrt{p_i})} \right)$

- Should choose **safe prime**: $p = 2q + 1$ for large prime q
- Even for safe primes: index calculus methods & NFS for DL in \mathbf{Z}_p^*
 - Run time is about $1.92 \cdot (\ln p)^{1/3} \cdot (\ln \ln p)^{2/3}$
 - Same as for RSA, so need pretty large p

Comparison

Security level	RSA (log N)	DLP in finite field (log p)	DLP subgroup size (log q)
80	1024	1024	1023
112	2048	2048	2047
128	3072	3072	3071
256	 15360	15360	15359

Exponentiation time performance scales with $O(n^3)$ for bit length n numbers
So RSA and finite field DL are basically same performance wise

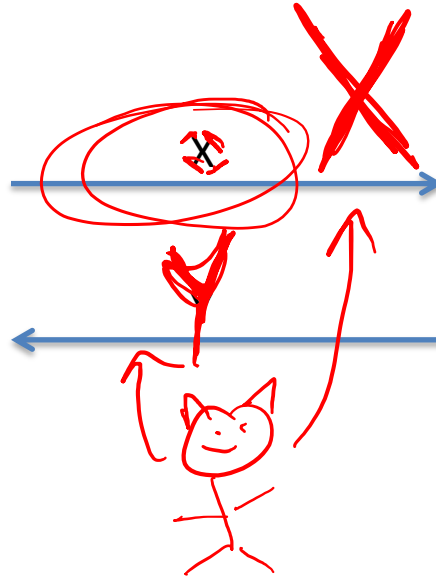
Finite field DL primes are specified in standards, see e.g.:
<https://datatracker.ietf.org/doc/html/rfc7919>

Unauthenticated Diffie-Hellman Key Exchange



Pick random x from $\mathbf{Z}_{|G|}$
 $X = g^x \bmod p$

$K = \text{HKDF}(Y^x \bmod p)$



Pick random y from $\mathbf{Z}_{|G|}$
 $Y = g^y \bmod p$

$K = \text{HKDF}(X^y \bmod p)$

Get the same key. Why? $Y^x = g^{yx} = g^{xy} = X^y$

What does adversary **need** to compute to break security (learn K)?

What **suffices** for adversary to compute to break security (learn K)?

Computational Diffie-Hellman Problem

Fix a cyclic group G with generator g

Pick x, y both at random $\mathbf{Z}_{|G|}$

Give adversary $g, X = g^x, Y = g^y$

Adversary must compute g^{xy}

For most groups, best known algorithm solves DL of X or Y

But we have no proof that this is best approach

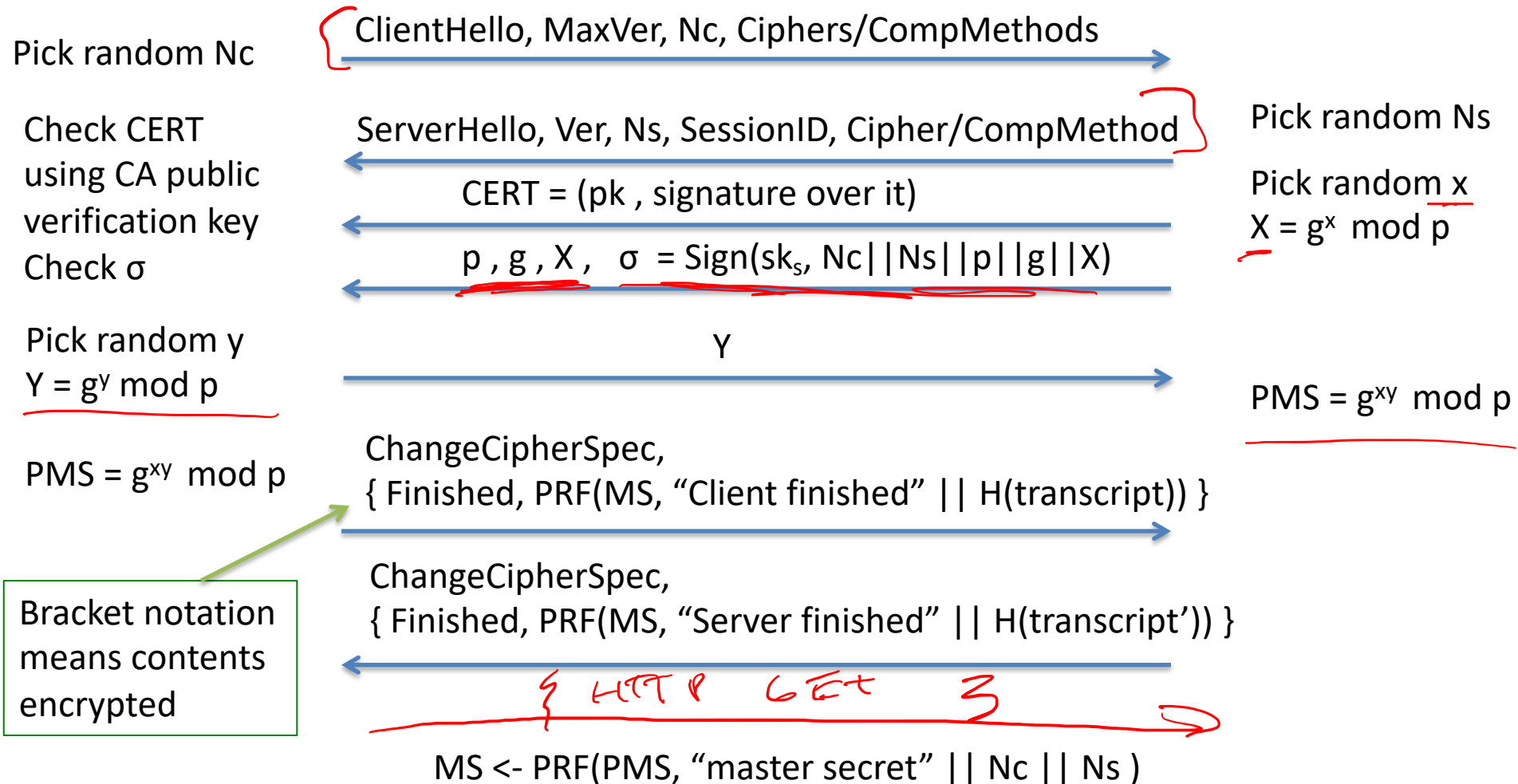


Client

TLS 1.2 handshake for Diffie-Hellman Key Exchange

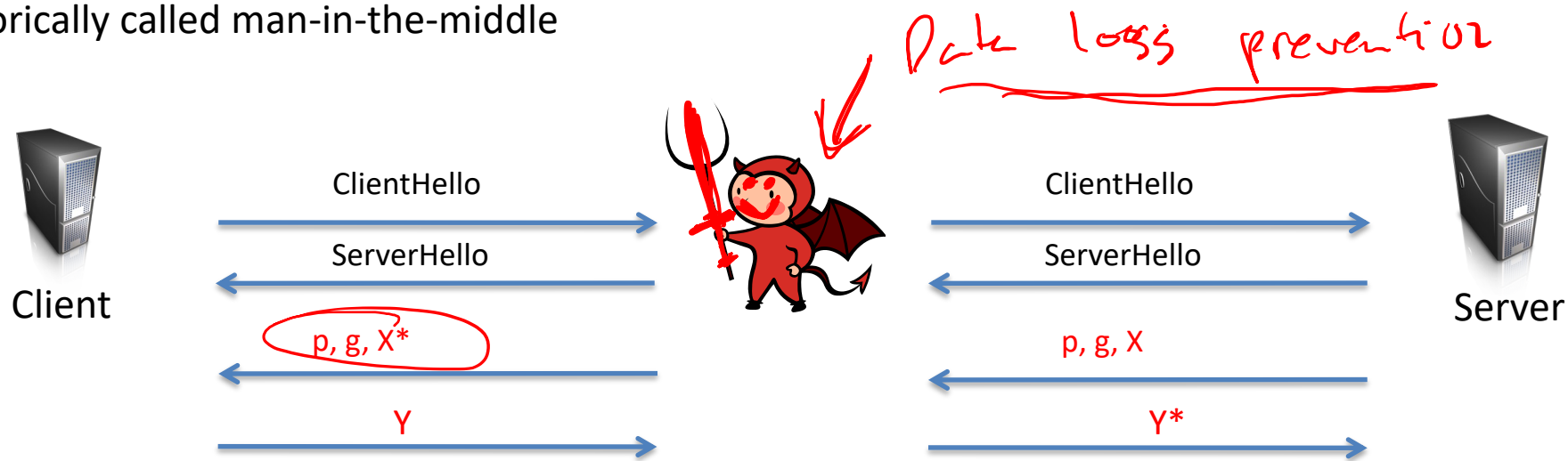


Server



Meddler-in-the-middle (MITM) attacks

Historically called man-in-the-middle



Attacker can choose X^* , Y^* , so it knows discrete logs

Completes handshake on both sides

Client thinks its talking to Server

All communications decrypted by adversary, re-encrypted and forwarded to server

MITM proxy implementation:

<https://docs.mitmproxy.org/stable/concepts-howmitmproxyworks/>

When we discuss digital signatures & PKI we will show how to prevent this

TLS 1.3 Design

- TLS 1.3 RFC 8446, August 2018
 - Eric Rescorla (now Mozilla CTO) wrote it, but dozens of contributors
 - Many years of work, discussions at conferences, IETF workshops, online. Research groups proposing suggestions, analyses
 - Design closely integrated with academic community to validate designs
- Targets improvements on many fronts:
 - Handshake latency. 1-RTT and 0-RTT modes (round trip time)
 - Improved privacy (encrypt more of handshake)
 - No more key transport, only forward-secret modes
 - Deprecate old ciphers, use only modern crypto schemes. Only AEADs



Client

TLS 1.3 handshake for Diffie-Hellman Key Exchange



Server

Pick random N_c

Pick random x
 $X = g^x \text{ mod } p$

$PMS = g^{xy} \text{ mod } p$

Bracket notation
means contents
encrypted

ClientHello, Ver, N_c , {Supported Versions},
{Supported Ciphers}, X

ServerHello, Ver, N_s , SessionID, Cipher, Y

{CERT = (pk, signature over it), PRF transcript so far}

{ ClientFinished, PRF of transcript so far }

{ application data (e.g., HTTP request) }

Has pk,sk pair

Pick random N_s

Pick random y
 $Y = g^y \text{ mod } p$

$PMS = g^{xy} \text{ mod } p$

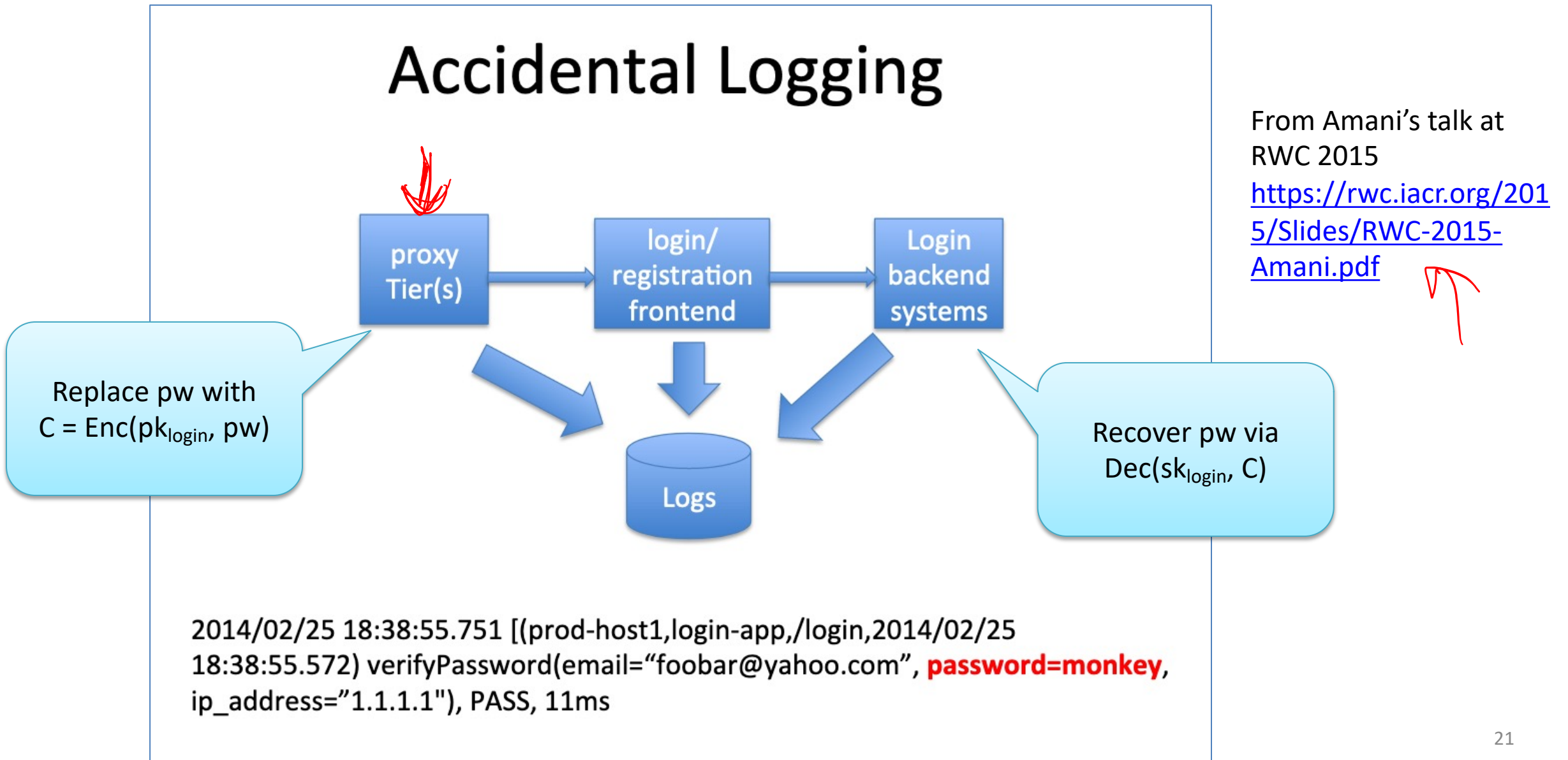
Interactive handshake inspector: <https://tls13.xargs.org/>

Public-key encryption beyond TLS

- Many other places where we want to use public-key encryption
 - Password encryption at TLS endpoint (LinkedIn example)
<https://rwc.iacr.org/2015/Slides/RWC-2015-Amani.pdf>

The logging problem and complex infrastructure

Accidental Logging



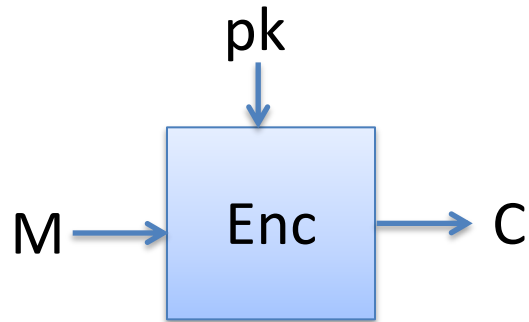
Public-key encryption beyond TLS

- Many other places where we want to use public-key encryption
 - Password encryption at TLS endpoint (LinkedIn example)
<https://rwc.iacr.org/2015/Slides/RWC-2015-Amani.pdf>
 - PGP and encrypted email
 - Encrypted messaging
 - ...

Public key encryption from CDH

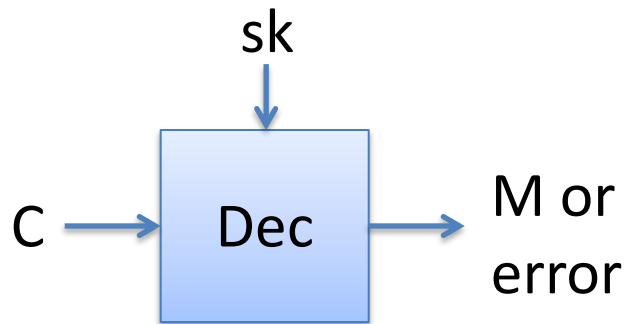
Fix cyclic group G with generator g

Kg outputs random sk from $\mathbf{Z}_{|G|}$ and $pk = \underline{g^{sk}}$



```
Enc(pk, M)  
r <- $  $\mathbf{Z}_{|G|}$  ; K <- KDF(  $pk^r$  )  
 $C_{kem} <- g^r$   
 $C_{dem} <- AEnc(K, M)$   
Return (  $C_{kem}$ ,  $C_{dem}$  )
```

AEnc, ADec are
authenticated encryption
& decryption, respectively
such as AES-GCM



```
Dec(sk,  $C_{kem}$ ,  $C_{dem}$ )  
K <- KDF(  $(C_{kem})^{sk}$  )  
M <- ADec(K,  $C_{dem}$  )  
Return M
```

Can optionally bind KEM
ciphertext to DEM
ciphertext by using it as
associated data with AEnc
& ADec

Example hybrid encryption schemes in practice

- ECIES (Elliptic curve with included encryption scheme), also called DHIES (Diffie-Hellman Integrated Encryption Scheme)
- libsodium / NaCL library cryptobox primitive
- Many standards: ANSI X9.63, IEEE 1363a, ISO/IEC 18033-2 and SECG SEC 1
- HPKE new standard to try to have interoperable, widely supported hybrid encryption scheme
 - <https://tools.ietf.org/id/draft-barnes-cfrg-hpke-01.html>

Hybrid encryption sender is *not authenticated*

- Use of AEAD does not provide *sender authenticity*
 - Anyone can generate valid ciphertext
- Does provide *non-malleability*
 - Can't modify ciphertext on unknown message M to some related M'

```
Enc(pk, M)
r ← $ Z|G| ; K ← KDF( pkr )
Ckem ← gr
Cdem ← AEnc(K, Ckem, M)
Return (Ckem, Cdem)
```

Sender authenticity requires digital signatures,
or pre-shared symmetric key (PSK)

Summary

- Diffie-Hellman can be built from integers modulo large prime
 - Sometimes called finite field Diffie-Helman
 - Should use safe primes due to Pollard-Hellman algorithm, size and efficiency about same as RSA
- Used to generate ephemeral keys for TLS key exchange, providing forward secrecy
- Used for hybrid encryption schemes, such as HPKE