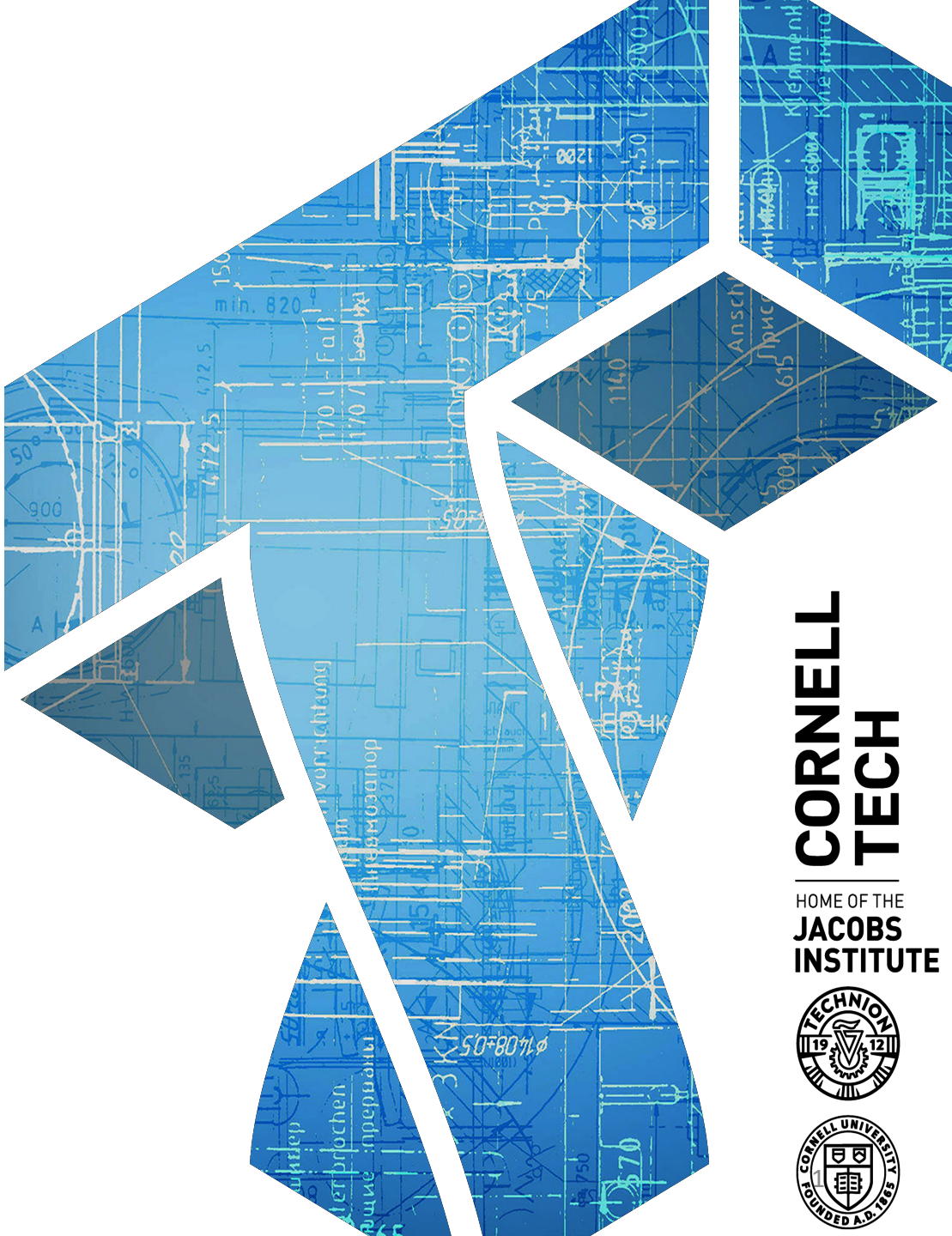
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Backdrop: Nation-state surveillance

Governmental intelligence agencies tasked with spying

- Mostly via end-point compromise, hacking operations
  - Government agency examples:
    - Tailored Access Operations (TAO) & FOXACID (USA)
    - Unit 8200 (Israel)
    - PLA Units 61398, 61486 (China)
    - Fancy Bear / GRU (Russia)
  - Commercial examples:
    - FinSpy by FinFisher Gmbh (Germany)
    - NSO Group's Pegasus (Israel)
    - DarkMatter (UAE)
- Also via communications intercept
  - Strong commercial and public-sector cryptography impedes

https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html

Targeted attacks

- Dissidents, journalists, activists targeted by nation-states
  - Phishing attacks, botnet-style C&C servers to collect data
  - Remote Access Trojans (RATs)
- Small industry of companies providing “lawful access” tools


From: Melissa Chan <melissa.aljazeera@gmail.com>
To:
Sent: Tuesday, 8 May 2012, 8:52
Subject: Torture reports on Nabeel Rajab

Acting president Zainab Al Khawaja for Human Rights Bahrain reports of torture on Mr. Nabeel Rajab after his recent arrest.

Please check the attached detailed report along with torture images.

1 attachment: Rajab.rar 1.4 MB Save

Figure 1: E-mail containing FinSpy.

https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-blond.pdf
https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-marczak.pdf
https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-hardy.pdf

NSO Pegasus

- Ahmed Mansoor (UAE) human rights activist
  - "On August 10 and 11, 2016, Mansoor received SMS text messages on his iPhone promising "new secrets" about detainees tortured in UAE jails if he clicked on an included link."
- Analysis indicated that link connects to exploit chain to remotely jailbreak iPhone
  - Three zero-days (WebKit browser vuln, ASLR bypass, kernel vuln)
  - Remotely installs implant (spyware)

https://citizenlab.ca/2016/08/million-dollar-dissident-iphone-zero-day-nso-group-uae/

NSO Pegasus

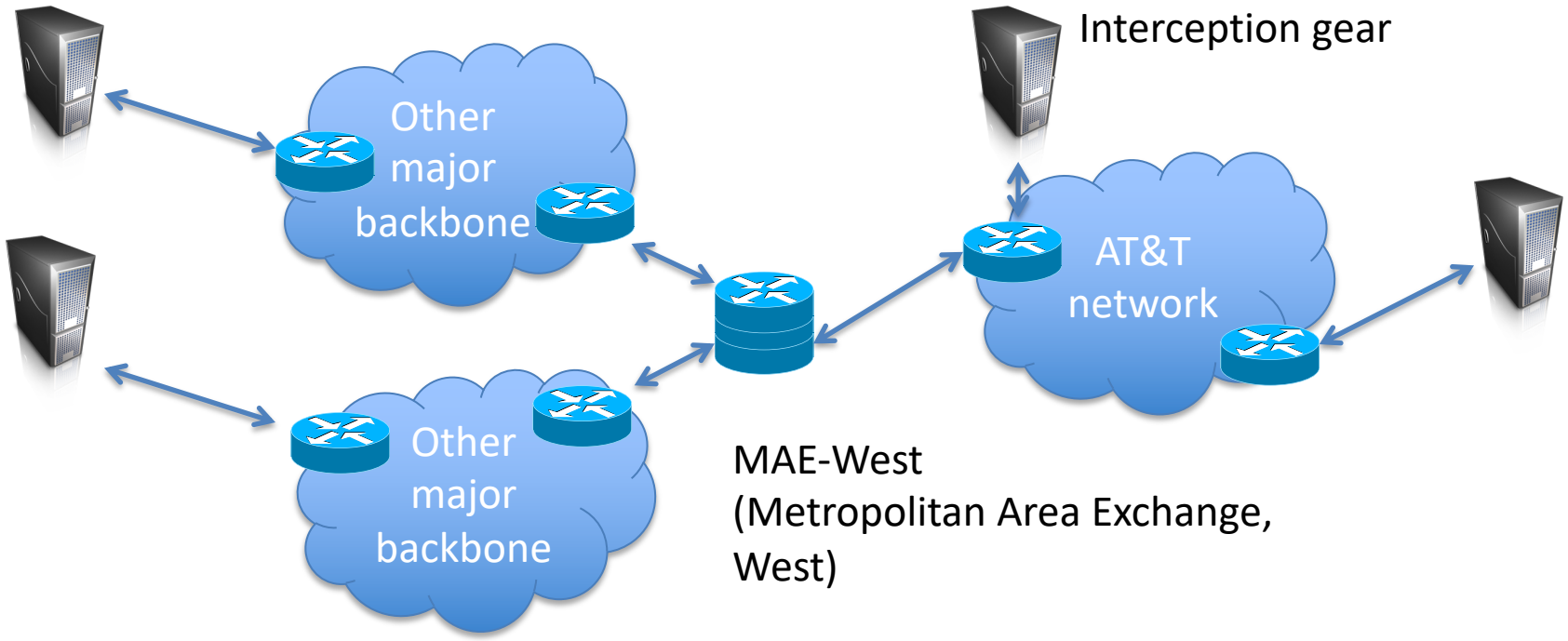
- Very recently CitizenLab uncovered evidence of UK government infections
- https://citizenlab.ca/2022/04/uk-government-officials-targeted-pegasus/
- Allege the infections linked to operations by UAE, India, Cyprus, and Jordan

https://citizenlab.ca/2016/08/million-dollar-dissident-iphone-zero-day-nso-group-uae/

Nation-state surveillance

- Governmental intelligence agencies spy on each other
  - Mostly via end-point compromise, hacking operations
    - Government agency examples:
      - Tailored Access Operations (TAO) & FOXACID (USA)
      - Unit 8200 (Israel)
      - PLA Units 61398, 61486 (China)
      - Fancy Bear / GRU (Russia)
    - Commercial examples:
      - FinSpy by FinFisher Gmbh (Germany)
      - NSO Group's Pegasus (Israel)
      - DarkMatter (UAE)
  - **Also via communications intercept**
    - **Strong commercial and public-sector cryptography impedes**

Wiretap surveillance

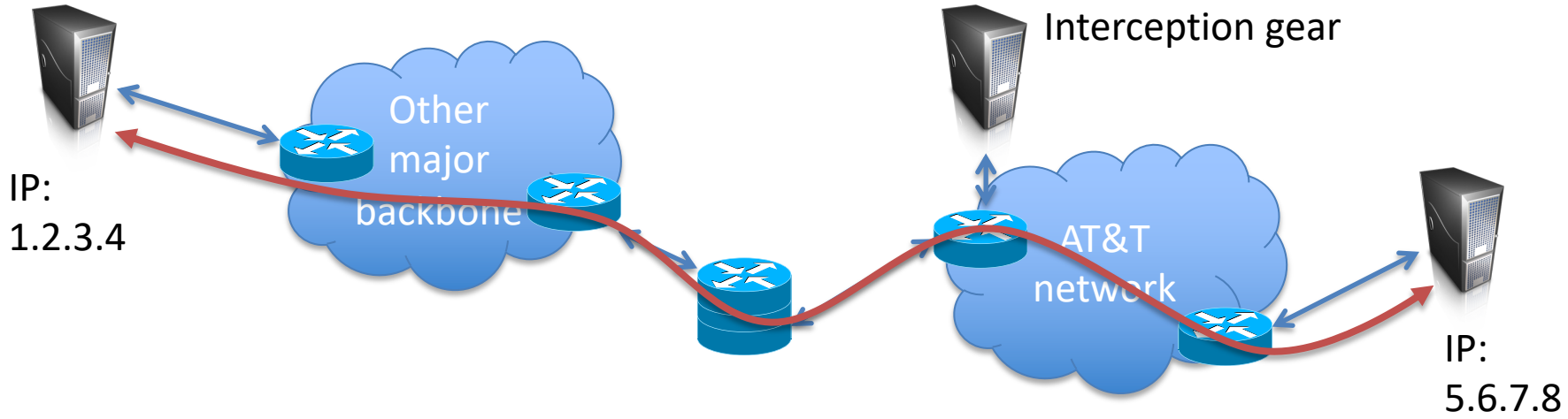


Large amounts of Internet traffic cross relatively few key points

AT&T Wiretap case

- Mark Klein discloses potential wiretapping activities by NSA at San Francisco AT&T office
- Fiber optic splitter on major trunk line for Internet communications
  - Electronic voice and data communications copied to “secret room”
  - Narus STA 6400 device

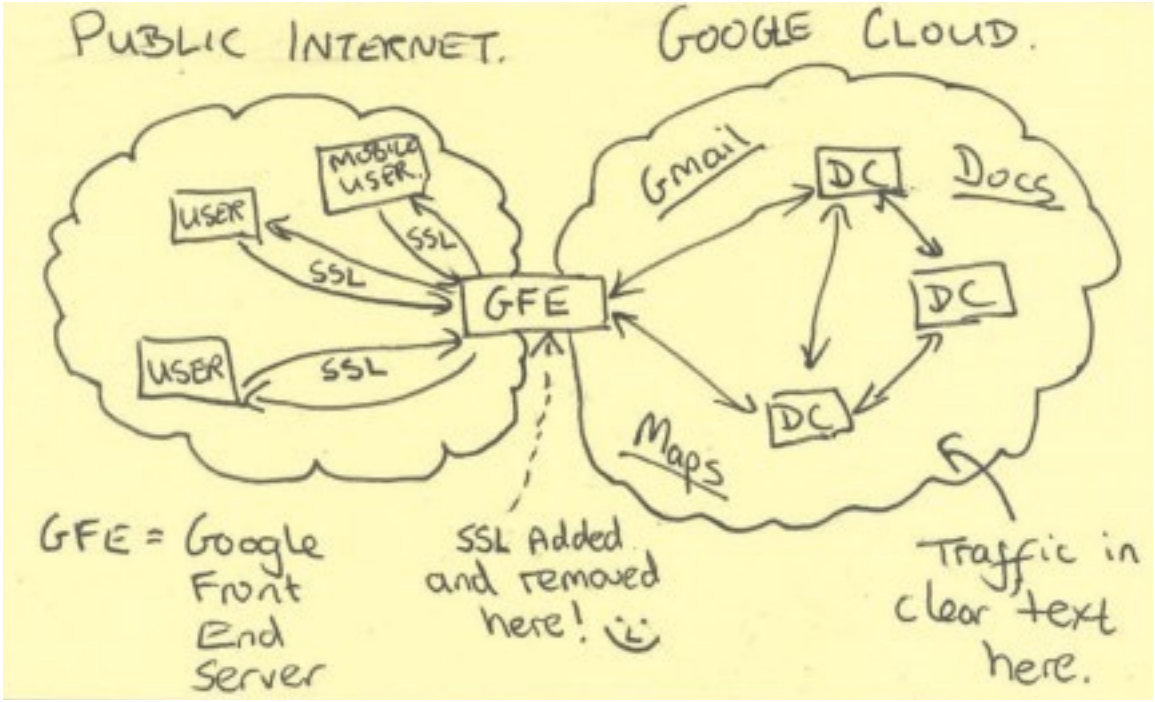
Encryption and intercept



- TLS, SSH, IPSec all prevent interception of plaintext data
- Prevents wiretapping for content

Snowden revelations

- HTTPS terminated at edge of Google networks
- Internal data center-to-data center communications on privately leased lines
  - No encryption up until summer 2013



History of overt weakening

Efforts to weaken public/commercial cryptography:

- DES key length reduced at NSA's bequest
- Clipper Chip in 1990s
  - NSA-designed encryption chip
  - Secret keys given at factory, escrowed with NSA
  - Campaign by cryptographic experts, others to prevent use
- Export controls (1990s and on)

Covert sabotage of crypto

- Purposefully inserting weaknesses into cryptographic protocols and/or implementations
- United States' NSA has history of doing so
  - Dual EC PRNG case probably most well known

Desiderata for good sabotage:

- Allow decryption, ideally in real time
- Decryption should be private
  - Only saboteur should be able to exploit
- Undetectability
- Others?

See https://eprint.iacr.org/2015/097.pdf
for taxonomy and easy-to-read summary

TLS 1.2 handshake for RSA transport

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

Check CERT using CA public verification key

← CERT = (pk of Amazon, signature over it by CA)

Pick random PMS
C <- Enc(pk,PMS)

C →

PMS <- D(sk,C)

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, “master secret” || Nc || Ns)

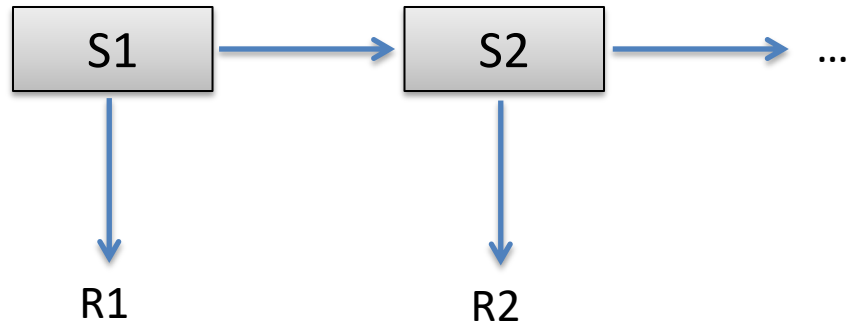
Sabotaging PRNGs

- Say we can sabotage client's random numbers to make them predictable
- Where do random numbers come from?
  - Use system service like /dev/urandom to generate initial seed S1
  - Use S1 with a pseudorandom number generator (PRNG)

(S2, R1) <- PRG(S1)

(S3, R2) <- PRG(S2)

⋮



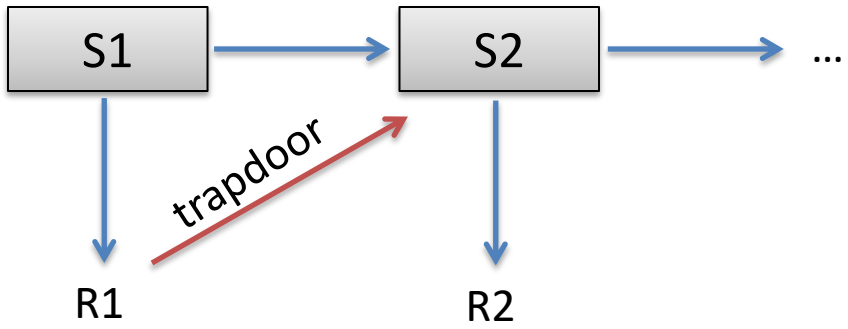
Sabotaging PRNGs

- Arrange that given R1, attacker with a trapdoor can compute S2
- This allows predicting all subsequent values

(S2, R1) <- PRG(S1)

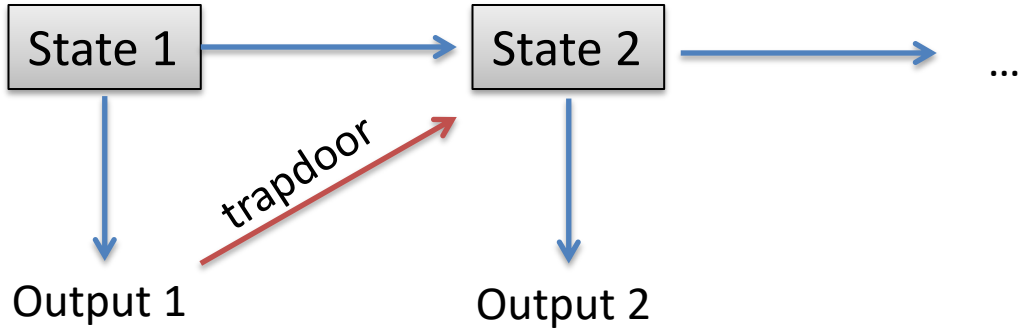
(S3, R2) <- PRG(S2)

⋮



Sabotaging PRNGs

- NIST's Dual EC pseudorandom number generator (PRNG) apparently backdoored
  - Mandated public parameters are public key
  - There exists a secret key, the trapdoor (known since 2005 Shumow, Ferguson)
- One output of PRNG + trapdoor reveals next state of PRNG, and prediction of future outputs



A Simple Diffie-Hellman Trapdoor

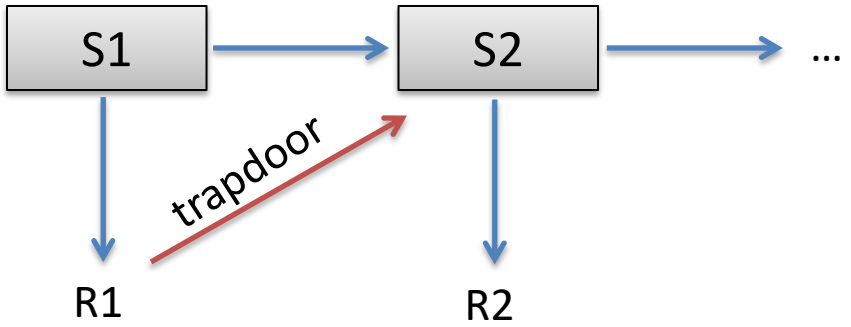
Let G be a cryptographically strong group with generator g

Let P in G be chosen parameter. Choose to be $P = g^p$

Let seed S1 be uniform value in $\mathbf{Z}_{|G|}$

$$PRG(S1) = (H(P^{S1}) , g^{S1}) = (S2,R1)$$

Given R1, p, compute $S2 = H(R1^p)$



Can view R1 as public-key encryption of next seed S2

Good PRNG to anyone without trapdoor p

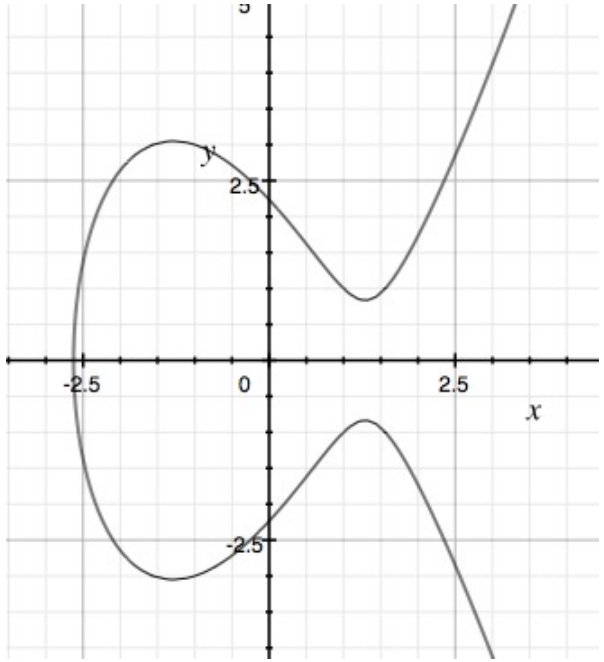
Dual EC is very similar

Parameters are curve points P, Q

If one knows value d such that P = d*Q,
then can predict future states

Again, informally view Dual EC output as public-key encryption of future state. Backdoor d is secret key that allows recovering future state

Some complicating details:
16 bits of “ciphertext” truncated



$y^2 = x^3 - 5x + 5$

(over the reals)

How easy is Dual EC to exploit in practice?

- PRNG may not be used in exploitable ways
  - May not be used in first place (many faster PRNGs out there)
  - Optional to truncate more than 16 bits
  - Dual EC supports *additional inputs* that could add entropy to each derivation, making attacks harder
  - May be implemented incorrectly
  - …

Checkoway et al. 2014 study

- Investigate implementations of TLS:

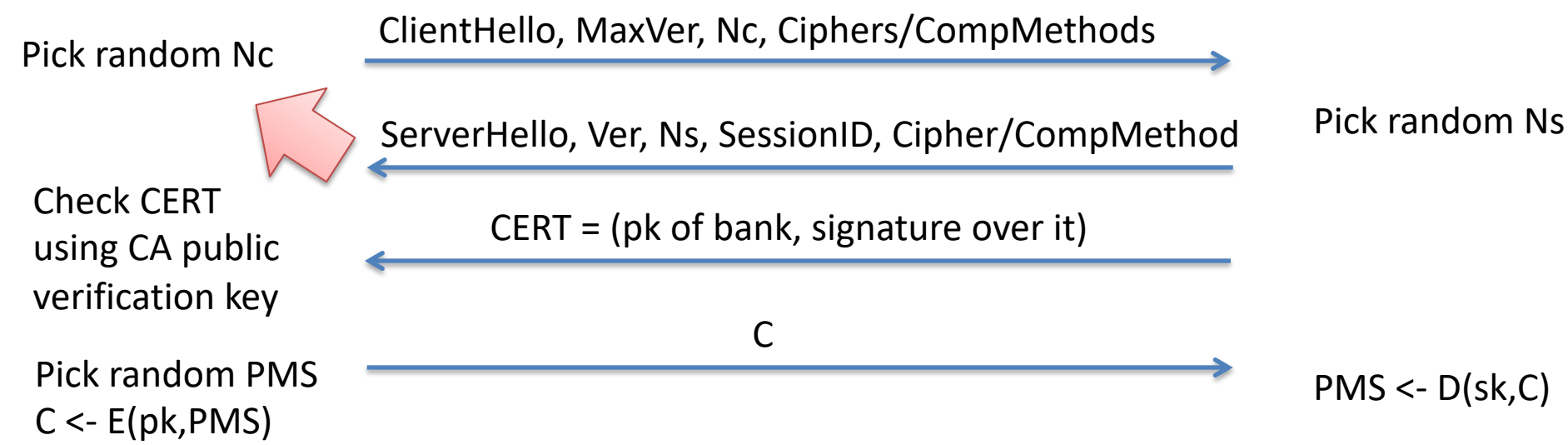
  openssl, Windows schannel, RSA BSAFE

- Conclude that some are more vulnerable than others:
  - Openssl bug prevents use of Dual EC (easy to fix)
  - Windows schannel uses additional input (deviates from Dual EC spec in ways that make attack faster)
  - RSA BSAFE very vulnerable

TLS handshake for RSA transport

Bank customer

Bank



Say client is using Dual EC for randomness generation

What is vulnerable?

RSA BSAFE library: 2.4 seconds to recover PMS

Windows: 60 minutes

OpenSSL: never (bug in code!)

Checkoway et al. 2014 study

Library	Default PRNG	Extended Random	Bytes per Session	Additional Entropy	Time (minutes)
BSAFE C	✓		31–60	—	0.04
BSAFE Java	✓	✓	28	—	63.96
SChannel I			28	—	62.97
SChannel II			30	—	182.64
OpenSSL-fixed I			32	20	0.02
OpenSSL-fixed II			32	35	83.32
OpenSSL-fixed III			32	$35+k$	$2^k \cdot 83.32$

ZMap scan of IPv4: only 720 servers using BSAFE Java

Juniper Dual EC Incident

[Checkoway et al. 2016]

- ScreenOS used in Juniper NetScreen firewall products. Used to perform VPN encryption (IPsec)
- Uses Dual EC, but supposedly wrapped within another PRNG. Shouldn’t be vulnerable, even to someone with trapdoor
- But it was. Worse, someone broke in and modified Q to a new value Q’
- Single 2008 patch modified P, introduced bug disabling secondary PRNG

Reversed ScreenOS PRNG

- Bunch of work to reverse engineer code
- Discover subtle bug in logic that disabled second PRNG step, thereby outputting Dual EC outputs directly to caller

- Really confusing history, summed up:
  - Juniper built backdoor into their code, Dual EC + subtle bugs + configuration of IKE
  - Someone broke in to Juniper and replaced Q with malicious Q' in 2012

Listing 1: The core ScreenOS 6.2 PRNG subroutines.

```
1  void prng_reseed(void) {
2    blocks_generated_since_reseed = 0;
3    if (dualec_generate(prng_temporary, 32) != 32)
4      error_handler("FIPS ERROR: PRNG failure, "
5                    "unable to reseed\n", 11);
6    memcpy(prng_seed, prng_temporary, 8);
7    prng_output_index = 8;
8    memcpy(prng_key,
9           &prng_temporary[prng_output_index], 24);
10   prng_output_index = 32;
11 }
12
13 void prng_generate(void) {
14   int time[2];
15   time[0] = 0;
16   time[1] = get_cycles();
17   prng_output_index = 0;
18   ++blocks_generated_since_reseed;
19   if (!one_stage_rng())
20     prng_reseed();
21   for (; prng_output_index <= 31;
22        prng_output_index += 8) {
23     // FIPS checks removed for clarity
24     x9_31_generate_block(time, prng_seed, prng_key,
25                          prng_block);
26     // FIPS checks removed for clarity
27     memcpy(&prng_temporary[prng_output_index],
28            prng_block, 8);
29   }
30 }
```

How do we prevent covert backdoors?

- Institutional safeguards
  - Laws and legal oversight
- Many treat intelligence agencies as bad-faith actors now
  - Separate defensive vs. offensive roles for government agencies?
- Secure the standardization processes
  - Dual EC should have been caught and excluded (it was known during process)
- Build cryptographic algorithms more robustly
  - “Nothing up our sleeve” parameters
  - E.g.: SHA-1 constant 0123456789ABCDEFFEDCBA9876543210F0E1D2C3

Backdrop 2: Lawful intercept

- Almost all national governments mandate some kind of intercept capabilities for law enforcement access
- CALEA
  - Communications Assistance for Law Enforcement Act (1995)
- FISA
  - Foreign Intelligence Surveillance Act (1978)
  - Demark boundaries of domestic vs. foreign intelligence gathering
  - Foreign Intelligence Surveillance Court (FISC) provides warrant oversight (good example of regulatory capture)
  - Executive order by President Bush suspend need for NSA to get warrants from FISC

Many proposals over the years

Exceptional access proposals:

- GCHQ virtual “crocodile clips” for encrypted messaging
- Apple’s CSAM client-side scanning for iMessage

GCHQ Virtual Crocodile Clips (“ghost user”)

https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate

“It’s relatively easy for a service provider to silently add a law enforcement participant to a group chat or call. The service provider usually controls the identity system and so really decides who’s who and which devices are involved - they’re usually involved in introducing the parties to a chat or call. You end up with everything still being end-to-end encrypted, but there’s an extra ‘end’ on this particular communication. This sort of solution seems to be no more intrusive than the virtual crocodile clips that our democratically elected representatives and judiciary authorise today in traditional voice intercept solutions and certainly doesn’t give any government power they shouldn’t have.”

Strong negative reactions. E.g.:

https://www.aclu.org/blog/privacy-technology/adding-ghost-user-our-encrypted-communications-no-crocodile-chip

CSAM Detection

https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf

- Announced in Summer 2021
- Mechanism to perform client-side detection of possible CSAM
  - Fuzzy match against database of known CSAM mechanisms
    - NeuralHash
  - Secure computation protocol to reveal to Apple servers if particular client has had some threshold number of matches
- Lots of pushback against this approach, it was sidelined

Discussion topic

Should governments mandate exceptional access mechanisms in encrypted messaging?

What are some other ways to do so?

What are the risks of suggested mechanisms?

Policy

- “Going dark” debate over last few years
  - Police and others argue encryption is preventing criminals from being caught
  - Push for building in backdoors into crypto & other systems
  - Manhattan DA have interesting report about smartphone unlocking
- Cryptographers & security folks arguing that mandated backdoors are really bad idea
  - Keys under doormats report

Academic computer security research

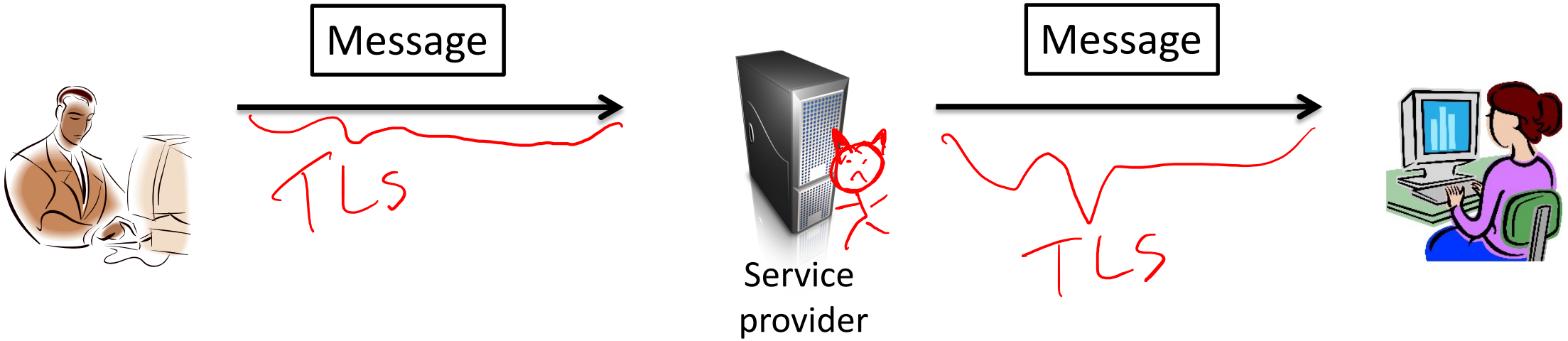
- Researchers play active role in uncovering state of computer security in the world, and understanding adversarial actions
- Useful in informing debates like encryption policy

Application-layer crypto

- So far focused on TLS as running example
  - Transport Layer Security
  - Provides network socket style stream interface

- What about if an application wants to encrypt discrete messages (as opposed to stream)?
  - Email
  - Text messages
  - Etc.

Messaging



What we want:

- Message may be large (body of email, PDF, image, etc.)
- Ideally: asynchronous communications (both parties need not be online)
- Desire confidentiality and authenticity (including replay resistance)

How would we design this given primitives we know?

Public-key encryption (RSA or Diffie-Hellman), digital signatures, symmetric encryption

ElGamal public-key encryption

g is generator for group of order p

Kg outputs pk = $(g, X = g^x)$ and sk = (g, x)

Enc((g,X), M)

$r \xleftarrow{\$} \mathbf{Z}_p$

$C1 = g^r$

$C2 = X^r * M$

Return C1 , C2

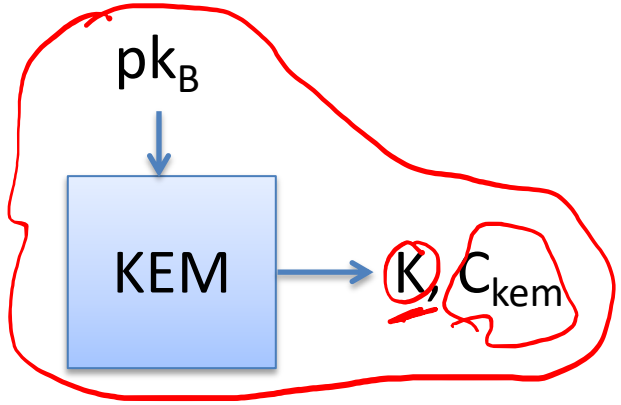
Dec((g,x), C1, C2):

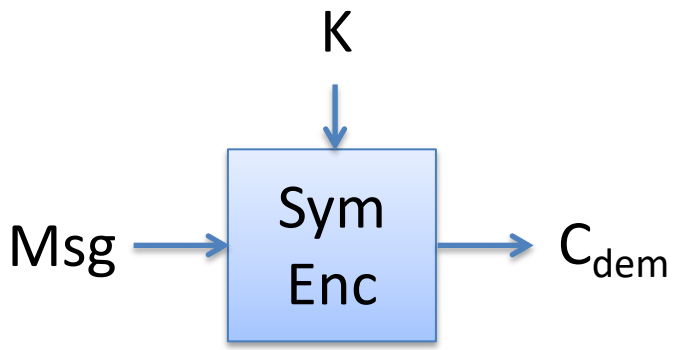
Return $C2 * C1^{-x}$

This is only at most chosen-plaintext attack secure. CCA attacks?

Only encrypts messages of size up to about log p bits

Hybrid encryption (KEM/DEM)





KEM = key encapsulation mechanism
Randomized public-key primitive

DEM = data encapsulation mechanism
One-time secure authenticated encryption

HybEnc(pk, M)
$K, C_{kem} \leftarrow\$ KEM(pk)$
$C_{dem} \leftarrow Enc(K,M)$
Return C_{kem}, C_{dem}

HybDec(sk, C_{kem} , C_{dem})
$K \leftarrow KEM^{-1}(sk, C_{kem})$
$M \leftarrow Dec(K, C_{dem})$
Return M

ElGamal KEM

Kg outputs pk = (g,X =g^x) and sk = (g,x)
g is generator for group of order prime p

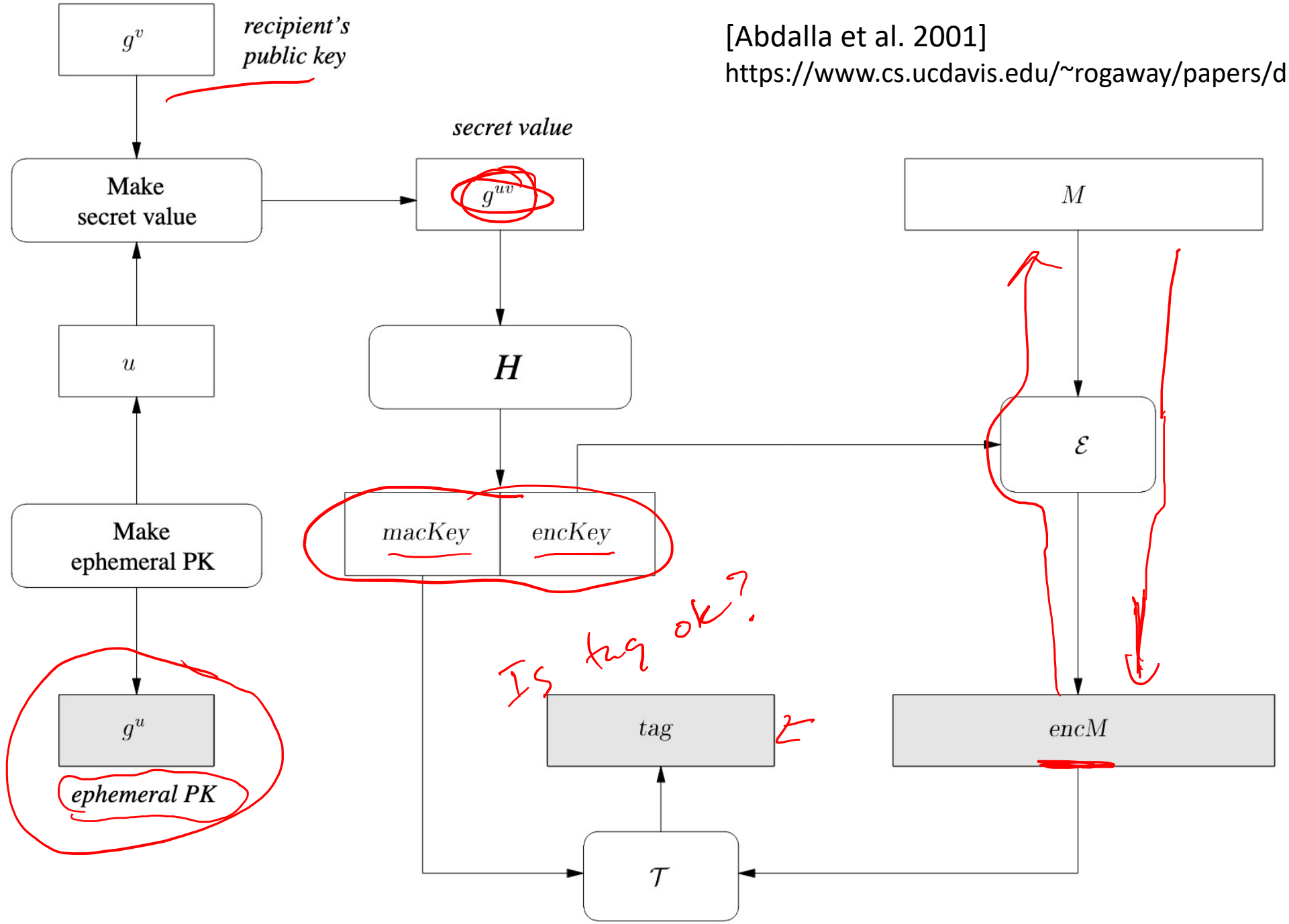
EG-KEM((g,X)):
r <-$ $\mathbf{Z}_p$
C_{kem} <- g^r
Return $H(X^r)$, C_{kem}

EG-KEM
Dec((g,x), C_{kem}):
Return $H(C_{kem}^x)$

Secure if computational Diffie-Hellman assumption holds in group

DHIES (Diffie-Hellman Integrated Encryption Scheme) is an example of public-key encryption using this type of KEM

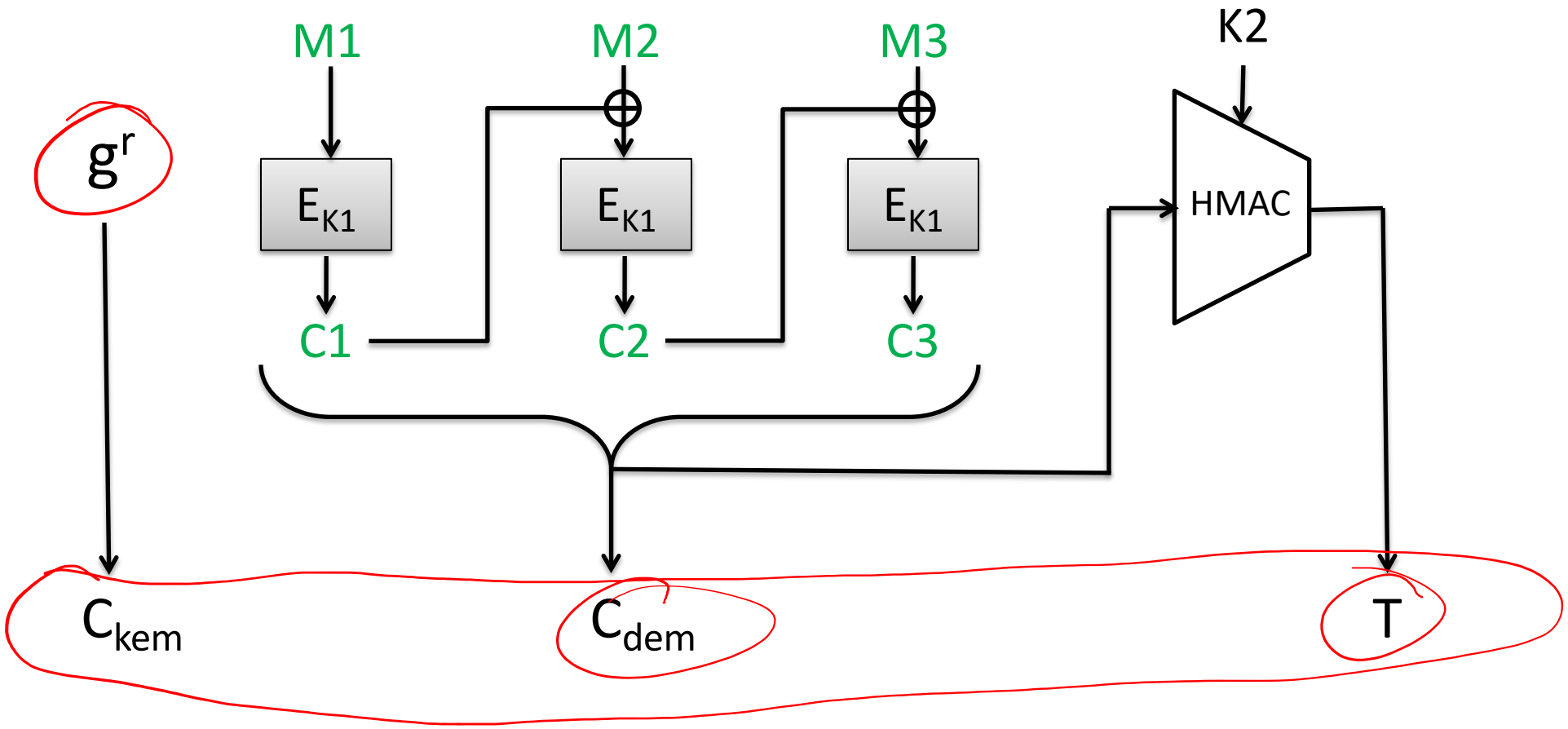
[Abdalla et al. 2001]
https://www.cs.ucdavis.edu/~rogaway/papers/dhies.pdf



DHIES with CBC and HMAC

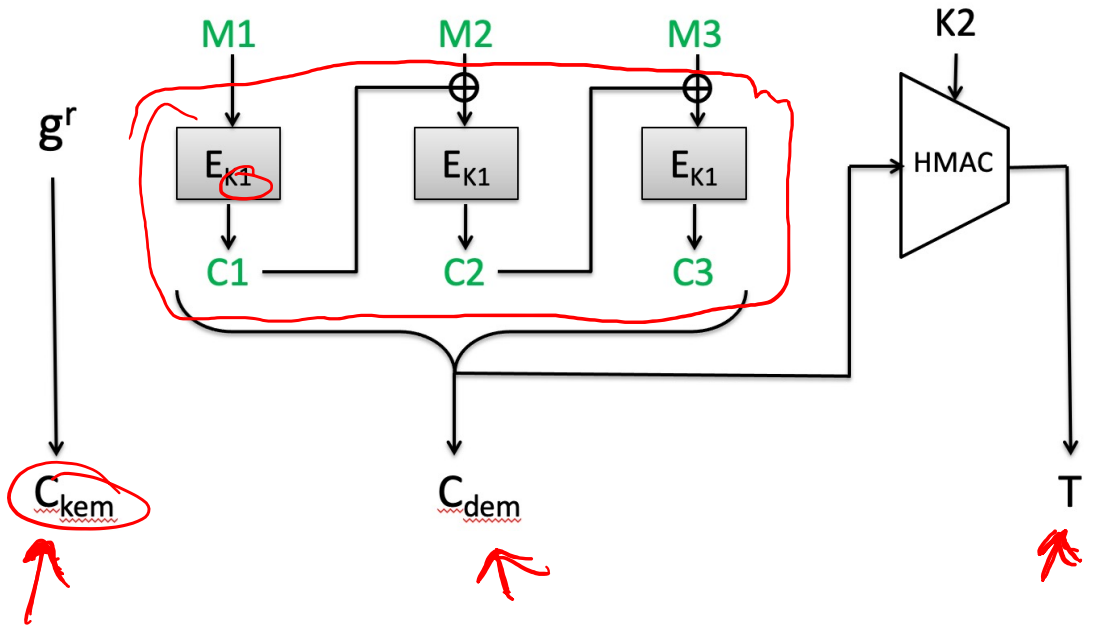
Recipient public key is g^x

K1 || K2 = SHA256(g^{xr})



DHIES security intuition

- Confidentiality for passive adversaries (IND-CPA)?
  - Symmetric encryption prevents

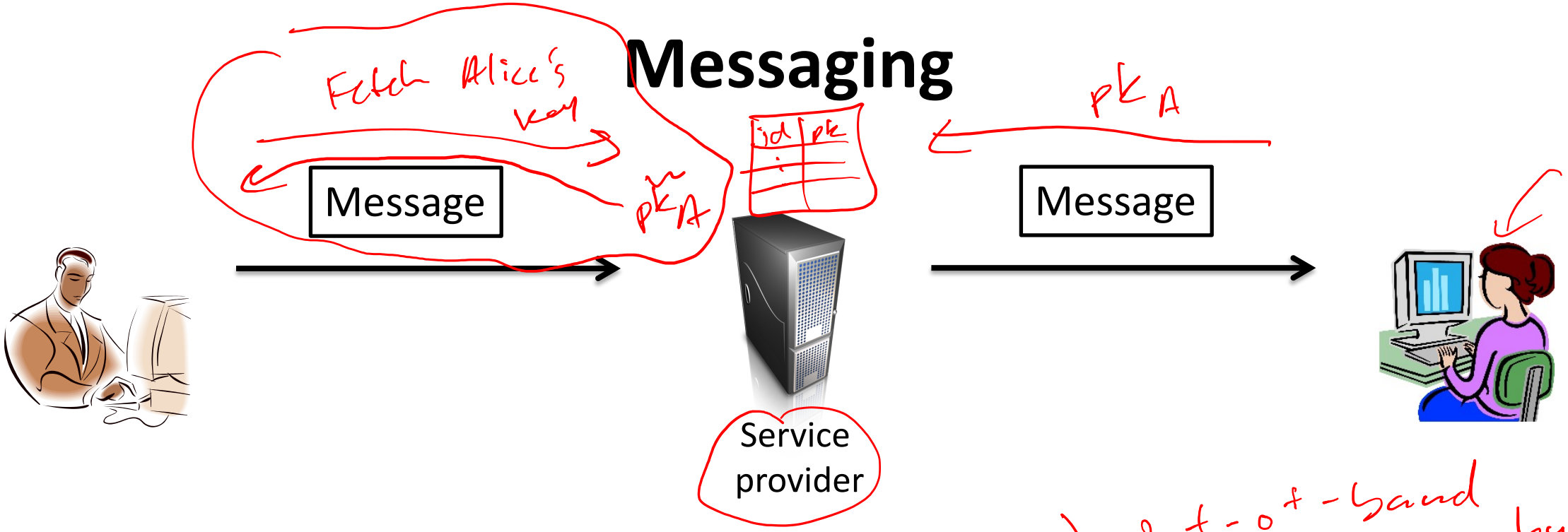


- Chosen-ciphertext attack security?
  - Change g^r => K1||K2 will be different
  - Change C_{dem} or T => tag check fails

- Can formally show security implied by:
  - IND-CPA security of symmetric encryption, message authenticity of tag, and (hashed) Diffie-Hellman assumption:

$(g^r, g^x, H(g^{rx}))$ indistinguishable from (g^r, g^x, U) (U uniform random bits)

Messaging



Want to use DHIES to encrypt message

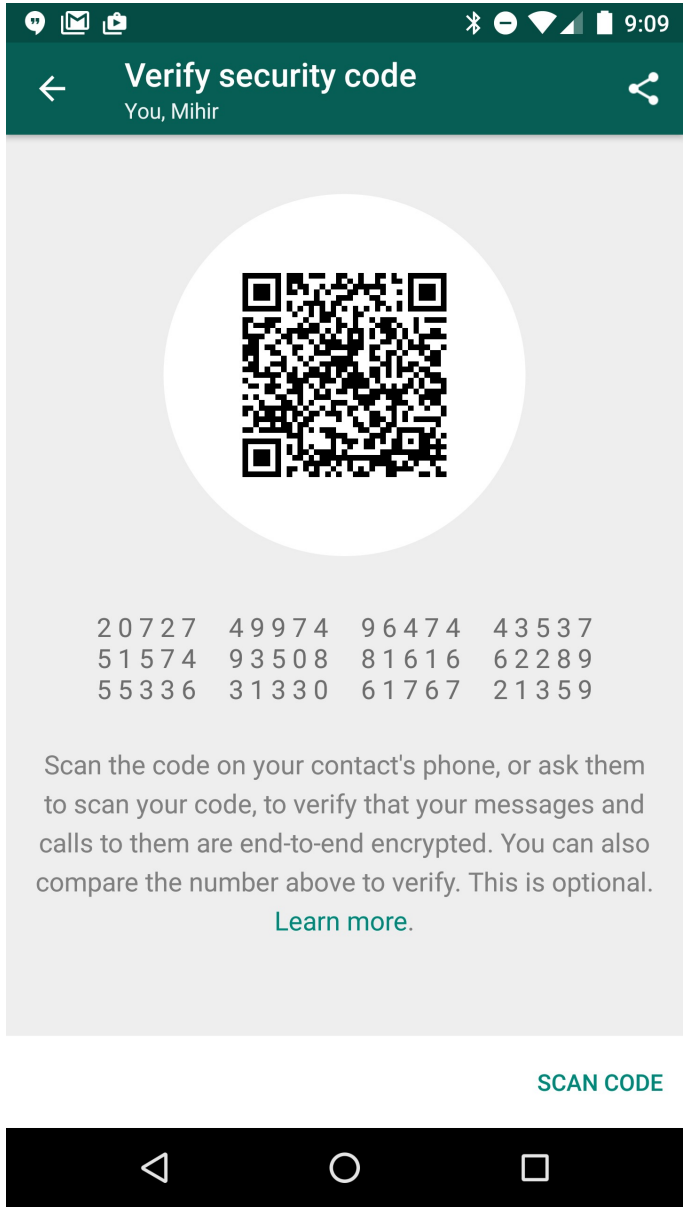
How do we get recipient's public key?

How do we trust recipient's public key?

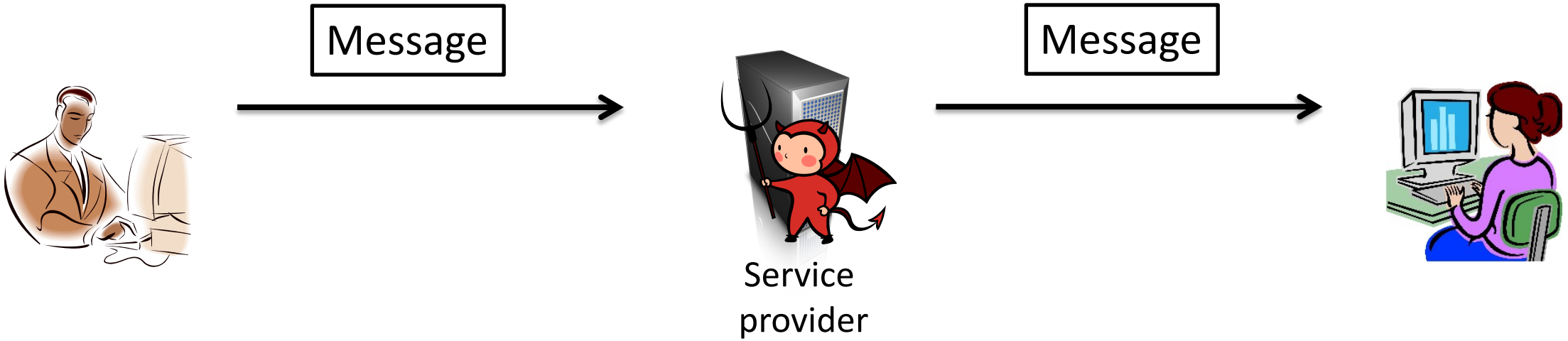
1) Out-of-band verification by users

2) Key transparency mechanisms

Verifying public keys



Messaging



Say we verify recipient public key

What attacks does using just DHIES still allow?

No authenticity! Service provider can inject arbitrary messages
No repeat protection! Service provider can replay ciphertexts

Need digital signatures and some replay resistance mechanisms

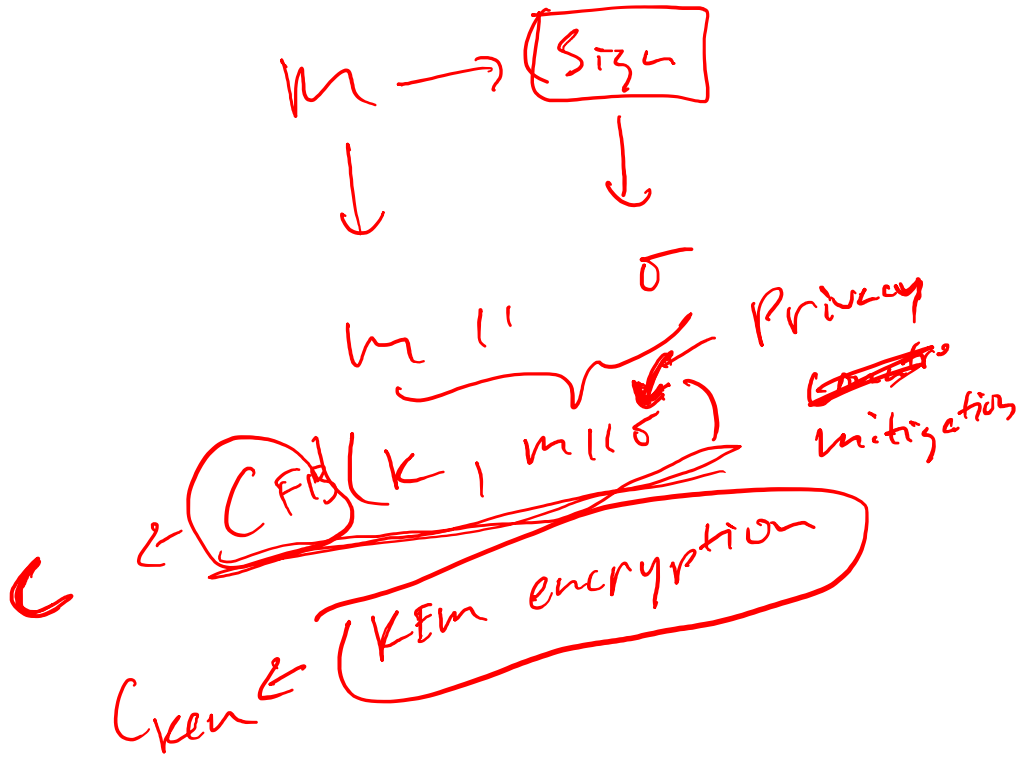
PGP history

- Phil Zimmerman released “Pretty Good Privacy” in 1991 on a USENET post marked as “US only”

- 1993: Criminal investigation by US government for munitions export without a license.
  - Printed PGP source code into a book. First amendment gambit

OpenPGP overview

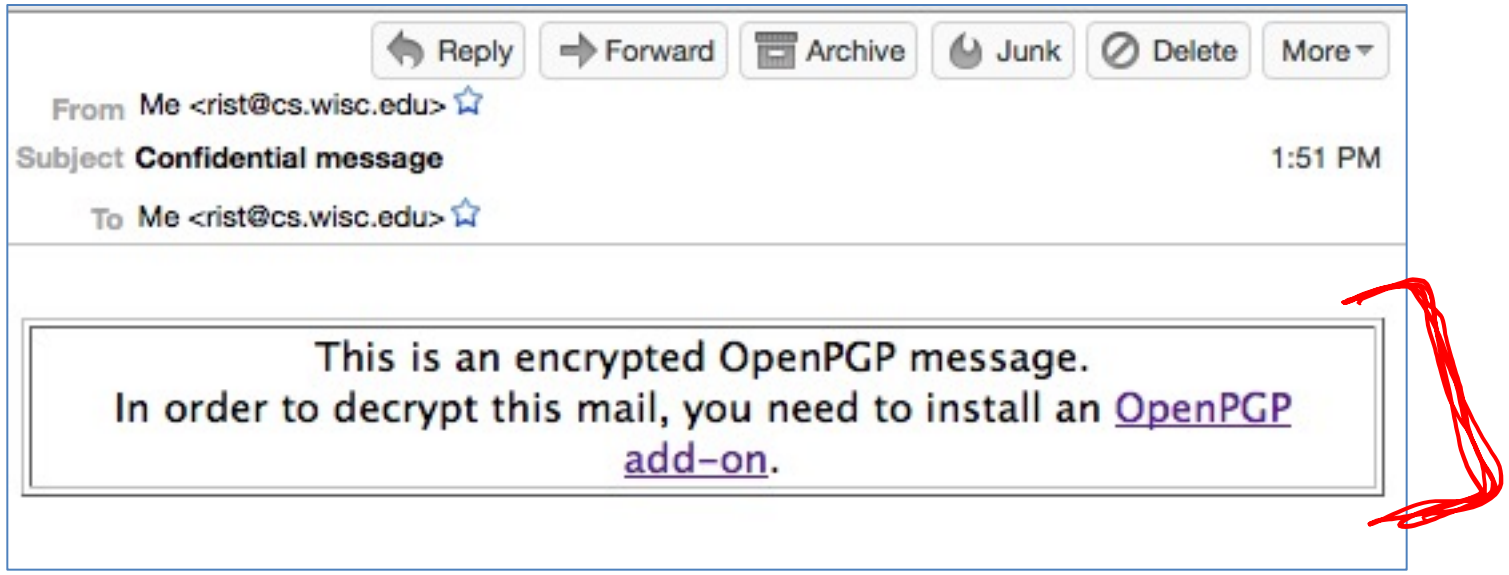
- Standard for PGP is RFC 4880
- Key encapsulation mechanism:
  - RSA PKCS#1 v1.5 encryption
  - ElGamal over finite field or elliptic curve
- Digital signatures:
  - RSA PKCS#1 v1.5 signatures
  - DSA
- Symmetric encryption:
  - Password-based key derivations using iterated hashing
  - CFB mode using block cipher (variant of CBC mode)



OpenPGP overview

- Security problems:
  - Padding oracle attacks against CFB & PKCS#1 v1.5
  - Attacks against home-brewed integrity checks (modification detection check, MDC)
  - Subject lines always in the clear
- Usability problems:
  - Users must manage their own keys
  - Copying private keys to each device
  - Checking validity of other recipient's public key

web of trust



The Switch

Yahoo's plan to get Mail users to encrypt their e-mail: Make it simple

A | 14 | Save for Later | Reading List

End-to-end (E2E) Encrypted messaging

- TextSecure by Open Whisper Systems (2013)
  - Trevor Perrin and Moxie Marlinspike (Levchin award winners)
  - Later became Signal
  - Developed Signal Protocol, which was adopted elsewhere (most notably: WhatsApp
- Protocol provides:
  - "… confidentiality, integrity, authentication, participant consistency, destination validation, forward secrecy, post-compromise security (aka future secrecy), causality preservation, message unlinkability, message repudiation, participation repudiation, and asynchronicity."

Signal Protocol (X3DH)

https://signal.org/docs/specifications/x3dh/

Identity key

Signed pre-key

Signature of SPK_B using secret key associated to IK_B

One-time pre-keys

IK_B, SPK_B, σ , $OPKi_B$

Service provider

IK_B, SPK_B, σ , $OPK1_B$, … , $OPKn_B$

IK_B, SPK_B, $OPKi_B$ all public DH keys, recipient retains secret keys

Verify σ

DH(PK1,PK2) is means output $g^{sk1*sk2}$

$r \leftarrow \$ \mathbf{Z}_p$; $EK_A \leftarrow g^r$

$DH1 = DH(IK_A, SPK_B)$
$DH2 = DH(EK_A, IK_B)$ — Mutual authentication
$DH3 = DH(EK_A, SPK_B)$
$DH4 = DH(EK_A, OPK_B)$ — Forward secrecy

$SK = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, DH4)$

$C \leftarrow AEAD(SK, message)$

IK_A , EK_A, key identifiers, C

IK_A , EK_A, key identifiers, C

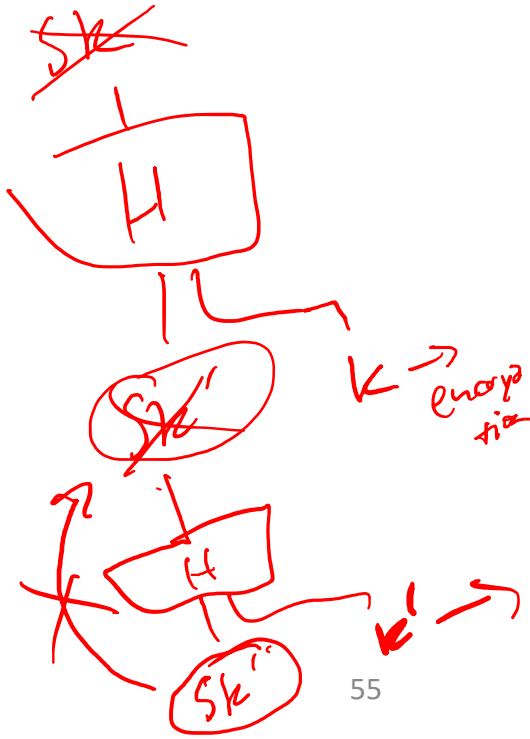
replay

delete secret key associated with $OPKi_B$

Signal Double Ratchet

https://signal.org/docs/specifications/doubleratchet/

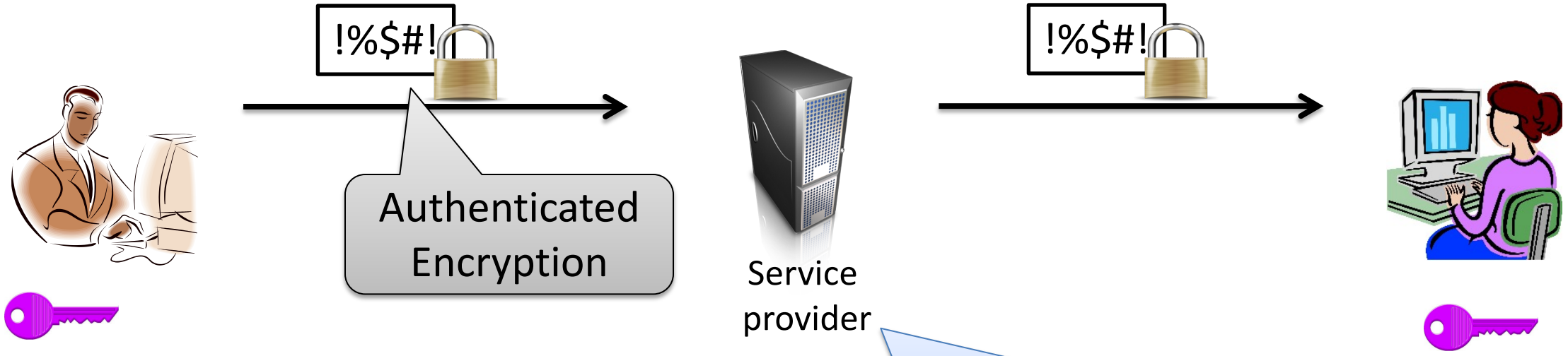
- Use X3DH to establish shared secret SK
- To improve forward secrecy, shouldn't use SK forever
- Ratcheting refers to deriving new key material to allow deleting old key material
- Asymmetric ratchet:
  - New ephemeral DH values sent along with messages
- Symmetric ratchet:
  - Use KDFs to generate new symmetric keys
- Key complexity: asynchronous communications



Encrypted messaging big topic

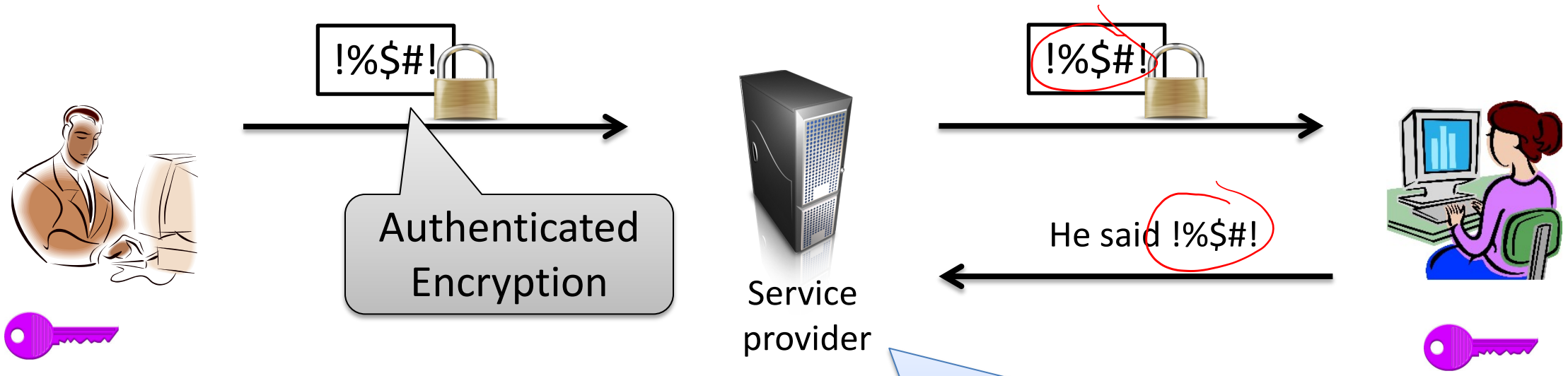
- Message Layer Security (MLS) in-progress standard
  - Helps handle group messaging, which has complexities
- Google, Facebook (Messenger & WhatsApp), Apple, Signal, Keybase, Skype, ... have implemented some E2E encryption
- Adds complexity to trust and safety features (e.g., abuse reporting)

End-to-end encrypted messaging and abuse



End-to-end security:
Provider cannot read or modify messages

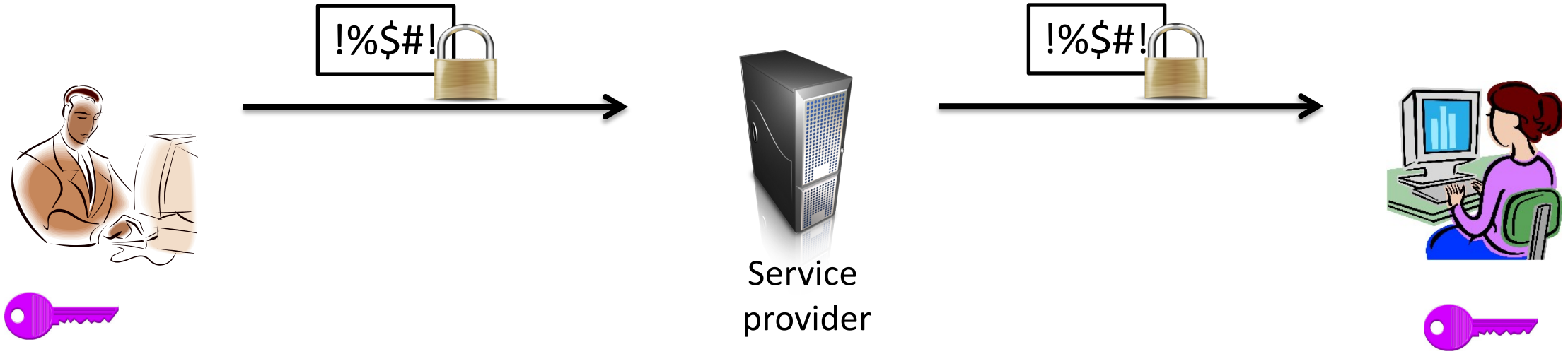
End-to-end encrypted messaging and abuse



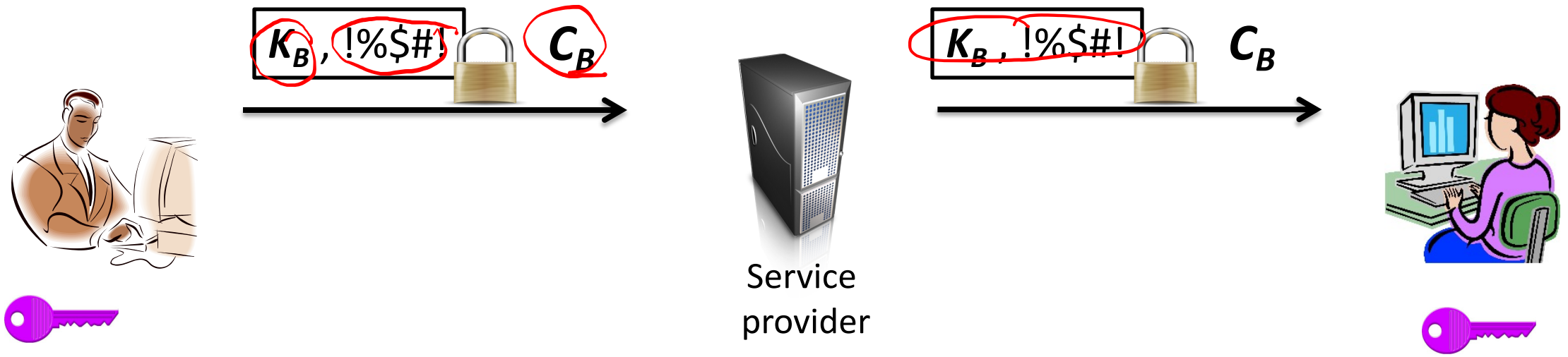
[Facebook 2016]:

- Provide cryptographic proof of message contents when reporting abuse
- Called technique ***message franking***

Facebook's message franking protocol



Facebook's message franking protocol

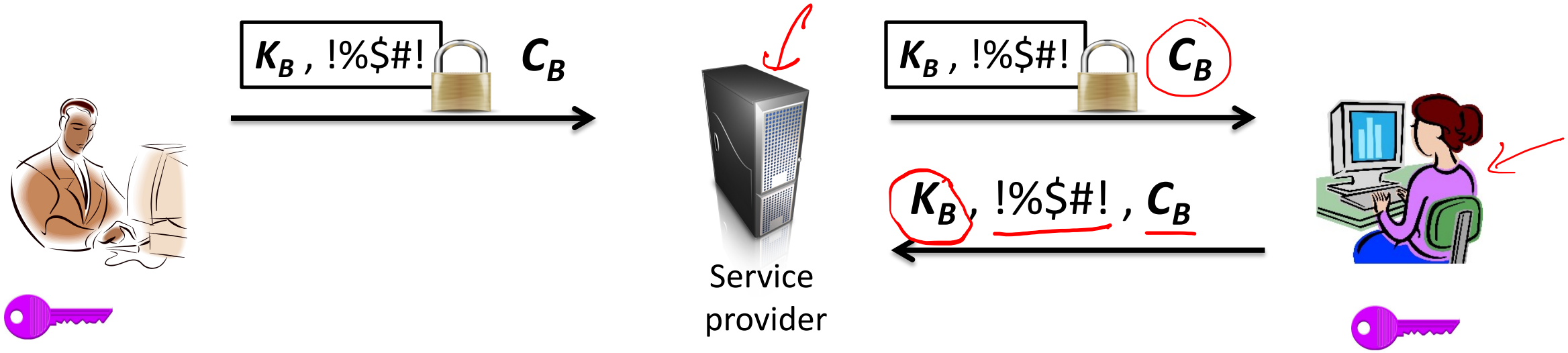


Sender ***cryptographically commits*** to message: $C_B = HMAC(K_B, \text{“!\%\$\#!”})$

Encrypt message along with K_B (called the opening)

Receiver decrypts, retrieves K_B, and verifies C_B

Facebook's message franking protocol

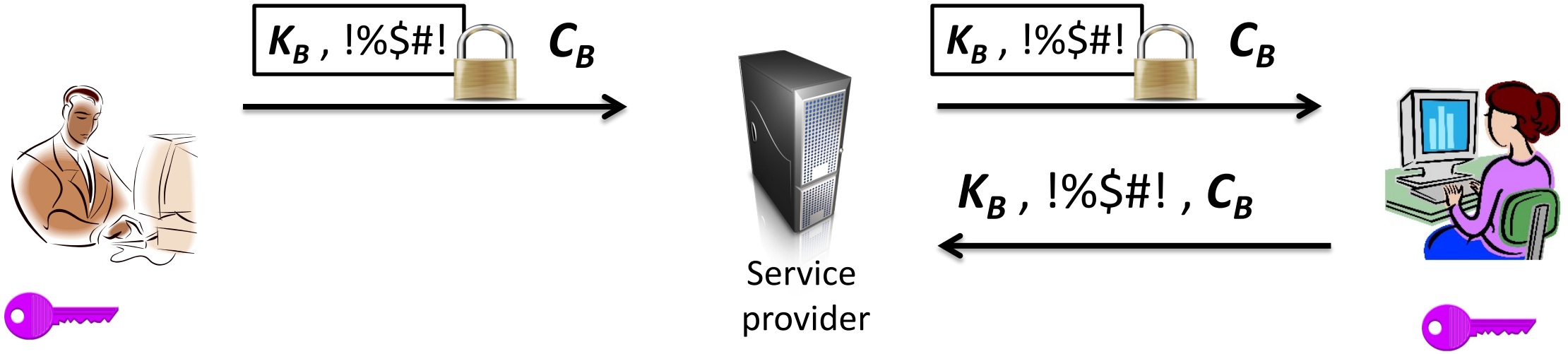


To report abuse, send message as well as K_B , C_B

Provider can verify that C_B was previously sent (and by whom)

Recompute $HMAC(K_B,$ "!%$#!"$)$ and check that it equals C_B

Security goals for message franking



1) ***Receiver binding***: receiver can't open a message not sent

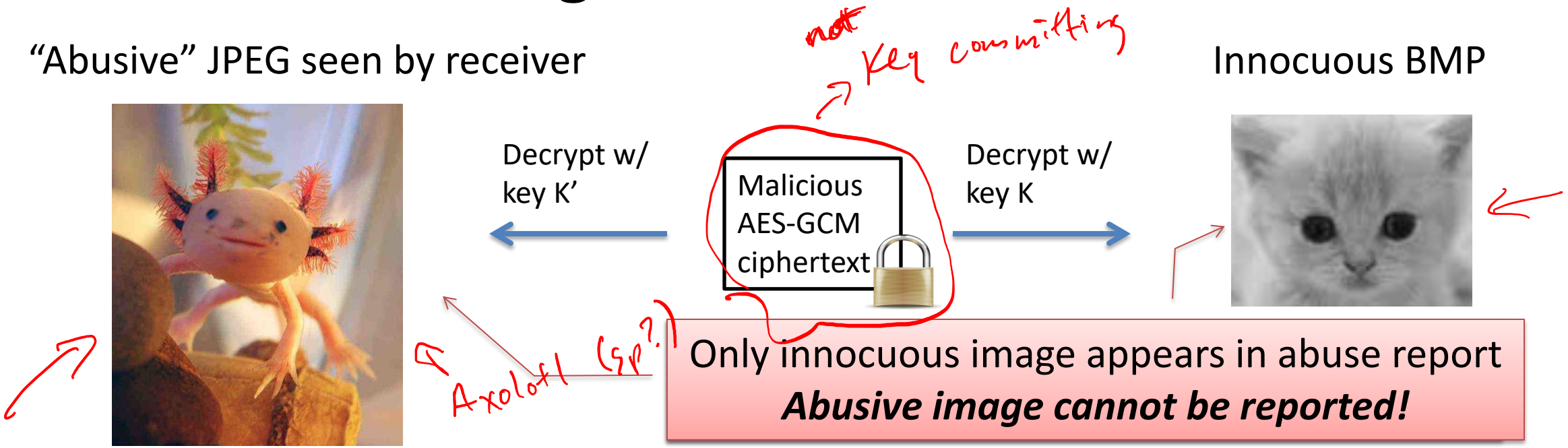
2) ***Sender binding***: can't send a message that can't be reported

3) End-to-end confidentiality/authenticity for messages not reported

~~Our~~ analysis:

Facebook's scheme achieves this for messages, but not for image/video attachments

An attack against Facebook moderation



Vulnerability due to weakness in standard authenticated encryption (AES-GCM)

- Find two keys K,K' that decrypt ***same*** ciphertext to ***distinct*** images

More recently: exploit this weakness to recover secrets [Len et al. Security 2021]

Abuse mitigation motivated finding fundamental weakness in encryption tools

- New cryptographic theory to guide design of better encryption
- New, secure algorithms

[Grubbs et al. Crypto 2017]
[Dodis et al. Crypto 2018]

Summary

- Hybrid encryption uses combination of asymmetric and symmetric cryptography
  - Key encapsulation mechanisms (KEM) based on secure PKE, (elliptic curve) Diffie-Hellman
  - Use an authenticated encryption scheme for data encapsulation mechanism (DEM)
- PGP is historical example (and still somewhat widely used)
- End-to-end messaging for IM, chat hotter topic, now widely deployed

Signal Protocol

Signed pre-key

Signature of SPK_B using secret key associated to IK_B

Identity key

One-time pre-keys

Service provider

$IK_B, SPK_B, \sigma, OPK1_B, \ldots, OPKn_B$

$IK_B, SPK_B, OPKi_B$ all public DH keys, recipient retains secret keys

$IK_B, SPK_B, \sigma, OPKi_B$

Verify σ

DH(PK1,PK2) is means output $g^{sk1*sk2}$

$r \leftarrow \$ \mathbf{Z}_p$; $EK_A \leftarrow g^r$

$DH1 = DH(IK_A, SPK_B)$

$DH2 = DH(EK_A, IK_B)$

Mutual authentication

$DH3 = DH(EK_A, SPK_B)$

$DH4 = DH(EK_A, OPK_B)$

Forward secrecy

$SK = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, DH4)$

$C \leftarrow AEAD(SK, message)$

IK_A, EK_A, key identifiers, C

IK_A, EK_A, key identifiers, C

Summary

- Hybrid encryption uses combination of asymmetric and symmetric cryptography
  - Key encapsulation mechanisms (KEM) based on secure PKE, (elliptic curve) Diffie-Hellman
  - Use an authenticated encryption scheme for data encapsulation mechanism (DEM)
- PGP is historical example (and still somewhat widely used)
- End-to-end messaging for IM, chat hotter topic, now widely deployed

Summary

- Elliptic curves are specially constructed groups where DLP is conjectured to be hard
- These are faster than RSA or DLP over $\mathbf{Z}_p^*$
- Being used widely in practice
  - EC-DSA (bitcoin)
  - TLS EC-DHE (elliptic curve ephemeral DH)
- Post-quantum crypto studies new asymmetric primitives conjectured to resist quantum computers