Cryptography
CS 555

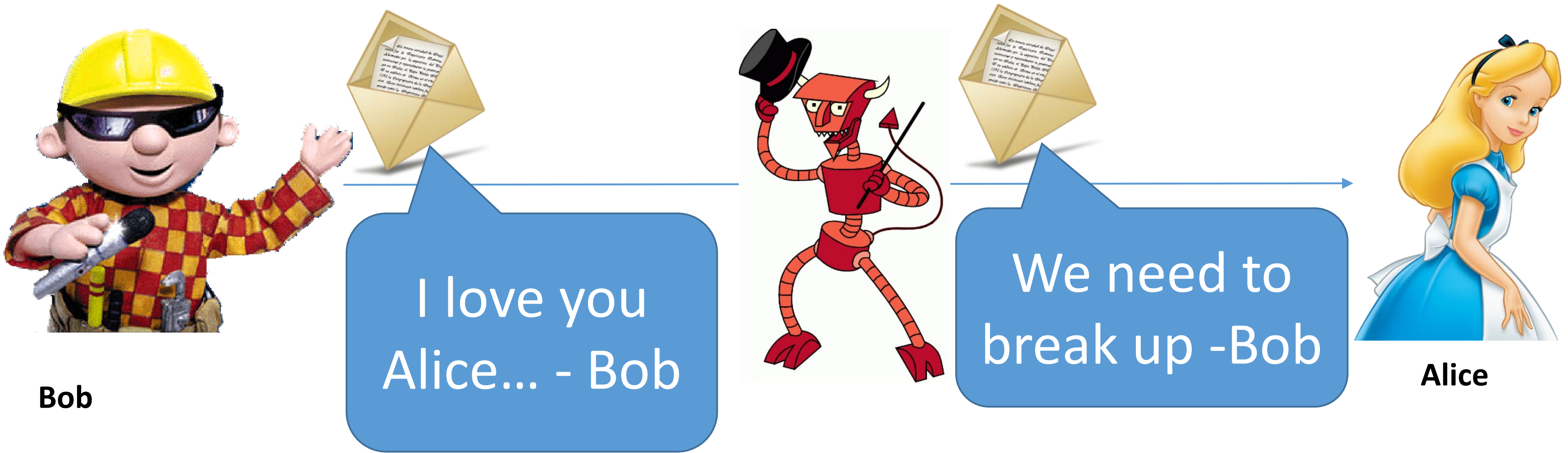
Topic 34: SSL/TLS

Recap

- Digital Signatures
- Attacks on Plain RSA Signatures
- RSA-FDH
- Secure Identification Scheme + Fiat Shamir Transform
- Digital Signature Standard

What Does It Mean to “Secure Information”

- Confidentiality (Security/Privacy)
  - Only intended recipient can see the communication
- Integrity (Authenticity)
  - The message was actually sent by the alleged sender



Signcryption: Authenticity + Confidentiality

- Public Key: **pk=(vk,ek**)
  - **vk** is used to **verify** messages
  - **ek** is used to **encrypt** messages

- Secret Key: **sk=(dk,sk)**
  - **dk** is used to **decrypt** messages
  - **sk** is used to **sign** messages

- **Goal**: Design a mechanism that allows a sender S to send a message m to a receiver R
  - Integrity
  - Secrecy

Attempt 1: Encrypt then Authenticate

- Sender S computes $\mathrm{c} = \mathrm{Enc}_{\mathbf{ek_R}}(m)$ and sends R

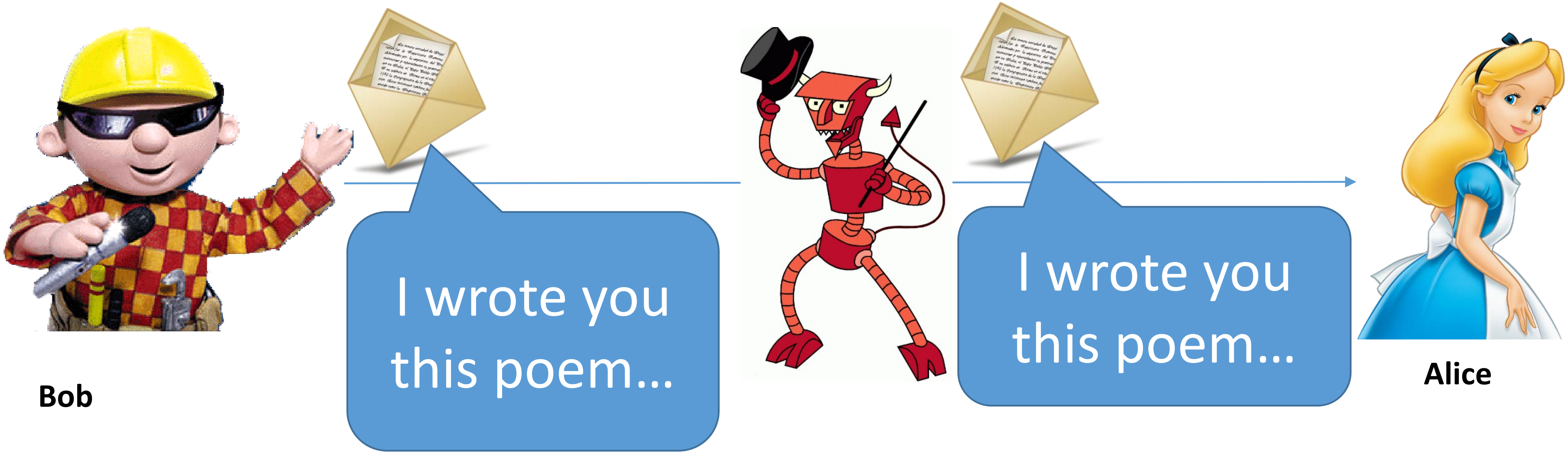
$$\langle S, c, \mathrm{Sign}_{\mathbf{sk_S}}(c) \rangle$$

- Receiver R decrypts c and then validates the signature

- This is the approach we used to build Authenticated Encryption with MACs
- Any problems here?

Attempt 1: Encrypt then Authenticate

$\langle Bob, c, \text{Sign}_{\mathbf{sk_B}}(c) \rangle$ $\langle \text{Devil}, c, \text{Sign}_{\mathbf{sk_{Devil}}}(c) \rangle$



Attempt 1: Encrypt then Authenticate

- Sender S computes $\mathrm{c} = \mathrm{Enc}_{\mathbf{ek_R}}(m)$ and sends R

$$\langle S, c, \mathrm{Sign}_{\mathbf{sk_S}}(c) \rangle$$

- Receiver R decrypts c and then validates the signature

- This is the approach we used to build Authenticated Encryption with MACs
- Another Issue:
  - How can R convince judge that sender S signed the message m?
  - Judge can verify that S signed the ciphertext, but needs R's key to decrypt c.

Attempt 2: Authenticate then Encrypt

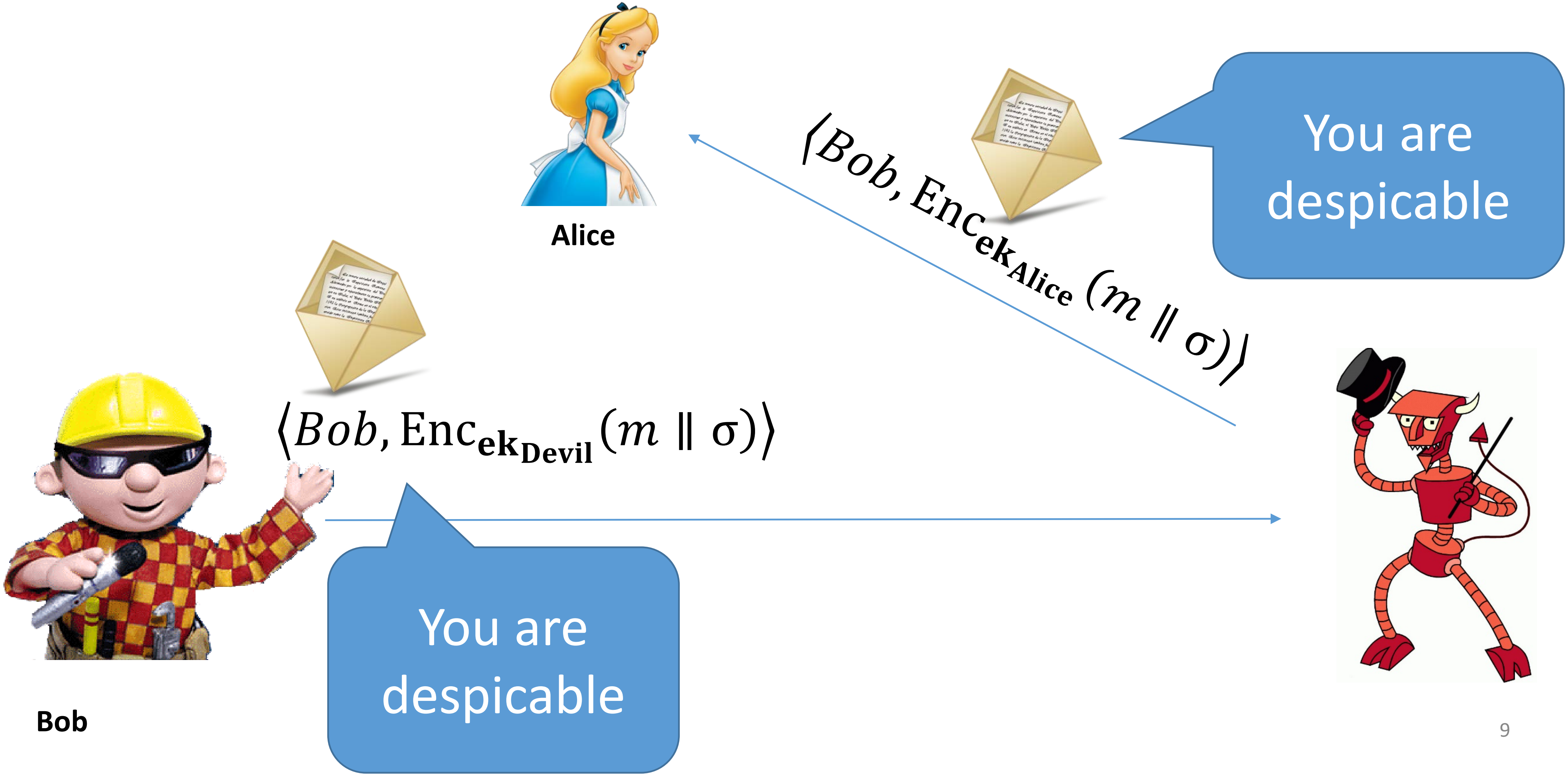
- Sender S computes $\sigma = \mathrm{Sign}_{\mathbf{sk_S}}(m)$ and sends R

$$\langle S, \mathrm{Enc}_{\mathbf{ek_R}}(m \parallel \sigma) \rangle$$

- Receiver R decrypts ciphertext to obtain m and then validates the signature σ

- Solve the issue of non-repudiation. Receiver obtains a signature σ for m
- Any other Issues?

Attempt 2: Authenticate then Encrypt



Attempt 3:

- Sender S computes $\sigma = \mathrm{Sign}_{\mathbf{sk_S}}(m \parallel R)$ and sends R

$$\left\langle S, \mathrm{Enc}_{\mathbf{ek_R}}(S \parallel m \parallel \sigma)\right\rangle$$

- This works ☺
- So does encrypt then authenticate with $\mathrm{c} = \mathrm{Enc}_{\mathbf{ek_R}}(S \parallel m)$

$$\left\langle S, c, \mathrm{Sign}_{\mathbf{sk_S}}(c \parallel R)\right\rangle$$

- Rule of Thumb:
  - When signing a message with your secret key include identity of receiver
  - When encrypting message with someone’s public key include your identity in message

Transport Security Layer (TLS)

- Standardized protocol based on processor SSL (Secure Socket Layer)
- Used for **https** connections by your browser
- Multiple Versions
  - TLS 1.0, 1.1, 1.2
  - (version 1.3 in progress https://tools.ietf.org/html/draft-ietf-tls-tls13-18)
- We will focus only on high level details

Transport Security Layer (TLS)

- First Goal: Agree on a set of keys
  - For Confidentiality
  - Also Authentication

- Handshake Precondition:
  - Client has a subset of $\{pk_1,\ldots pk_n\}$ --- public keys for several Certificate Authorities
  - Server has a key-pair (pk_s,sk_s) for a KEM

1. Client C begins by sending S a message indicating
   1. Protocol Versions + Ciphertext suites that he can run
   2. A random “nonce” N_C

Transport Security Layer (TLS)

1. Client C begins by sending S a message indicating
    1. Protocol Versions + Ciphertext suites that he can run
    2. A random “nonce” N_C
2. S responds by selecting the most recent version of the protocol it supports as well as an appropriate ciphersuite
    1. Also sends pk_S and certificate $cert_{i \to S}$ (signed message form certificate authority i validating pk_S)
    2. A nonce N_S
3. C checks to see if it has pk_i for CA_i.
    1. Yes? Verify the certificate and ensure that it is not expired/revoked
    2. No? Abort/Ask Again

Transport Security Layer (TLS)

1. Client C begins by sending S a message indicating
   1. Protocol Versions + Ciphertext suites that he can run
   2. A random “nonce” N_C
2. S responds by selecting the most recent version of the protocol it supports as well as an appropriate ciphersuite
   1. Also sends pk_S and certificate $cert_{i \to S}$ (signed message form certificate authority i validating pk_S)
   2. A nonce N_S
3. C checks to see if it has pk_i for CA_i.
   1. Assuming pk_S is validated…
   2. C runs $(c, pmk) \leftarrow \mathrm{Encaps}_{pk_S}(1^n)$ (pmk is *pre-master key*)
   3. C sends c to S (who will later use c and sk_S to recover pmk)
   4. C computes mk=KDF(pmk,N_C,N_S) (mk is master key)
   5. C computes four keys k_C,k_C’,k_S,k_S’= PRG(mk)
   6. C computes $\tau_C \leftarrow \mathrm{MAC}_{mk}(transcript)$ and sends $\left\langle Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right\rangle$ to S

Transport Security Layer (TLS)

1. Client C begins by se
   1. Protocol Versions
   2. A random "nonce"
2. S responds by select
   an appropriate ciph
   1. Also sends pk_S and
   2. A nonce N_S
3. C checks to see if it has pk_i for CA_i.
   1. Assuming pk_S is validated…
   2. C runs $(c, pmk) \leftarrow \mathrm{Encaps}_{pk_S}(1^n)$ (pmk is *pre-master key*)
   3. C sends c to S who recovers pmk
   4. C computes mk=KDF(pmk,NC,NS) (mk is master key)
   5. C computes four keys k_C, k_C', k_S, k_S' = PRG(mk)
   6. C computes $\tau_C \leftarrow \mathrm{MAC}_{mk}(transcript)$ and sends $\left\langle Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right\rangle$ to S

	Client Sends Message	Sever Sends Message
Encryption	k_C	k_S
MAC	k_C'	k_S'

Transport Security Layer (TLS)

3. C checks to see if it has pk_i for CA_i.
   1. Assuming pk_S is validated…
   2. C runs $(c, pmk) \leftarrow \mathrm{Encaps}_{pk_S}(1^n)$ (pmk is *pre-master key*)
   3. C sends c to S who recovers pmk
   4. C computes mk=KDF(pmk,NC,NS) (mk is master key)
   5. C computes four keys k_C, k_C', k_S, k_S'= PRG(mk)
   6. C computes $\tau_C \leftarrow \mathrm{MAC}_{mk}(transcript)$ and sends $\left\langle Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right\rangle$ to S
4. Sever
   1. Computes $pmk \leftarrow \mathrm{Decaps}_{sk_S}(c)$
   2. Computes mk=KDF(pmk,NC,NS) (mk is master key)
   3. Computes four keys k_C, k_C', k_S, k_S'= PRG(mk)
   4. Validates $\left\langle Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right\rangle$ by
      1. Decrypt $Enc_{k_c}(\tau_C)$ with to obtain τ_C
      2. If $\mathrm{Vrfy}_{k'_c}\left(Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right) \neq 1$ or $\mathrm{Vrfy}_{mk}(transcript, \tau_C) \neq 1$ then abort
      3. Otherwise server and client agree so far on communication

Transport Security Layer (TLS)

4. Sever
   1. Computes $pmk \leftarrow \mathrm{Decaps}_{sk_S}(c)$
   2. Computes mk=KDF(pmk,NC,NS)   (mk is master key)
   3. Computes four keys k_C,k_C’,k_S,k_S’= PRG(mk)
   4. Validates $\left\langle Enc_{k_c}(\tau_C), \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right\rangle$ by
      1. Decrypt $Enc_{k_c}(\tau_C)$ with to obtain τ_C
      2. If $\mathrm{Vrfy}_{k'_c}\left(\tau_C, \mathrm{MAC}_{k'_c}\left(Enc_{k_c}(\tau_C)\right)\right) \neq 1$ or $\mathrm{Vrfy}_{mk}(transcript, \tau_C) \neq 1$ then abort
      3. Otherwise server and client agree so far on communication
   5. S computes $\tau_S \leftarrow \mathrm{MAC}_{mk}(transcript')$ and sends $\left\langle Enc_{k_S}(\tau_S), \mathrm{MAC}_{k'_S}\left(Enc_{k_S}(\tau_C)\right)\right\rangle$ to C

5. Client validates τ_S; otherwise aborts

Security Intuition

- C verifies certificate so it knows it is talking to S
- Knows that only legitimate S can learn pmk and mk
- If protocol finishes successfully then C knows that it shares four keys k_C, k_C', k_S, k_S' with S
- MAC on transcript?
  - Ensures consistency
  - Man-in-the-Middle attacker may attempt to modify ciphersuite
  - E.g., force C and S to use old version of cipher with security bugs etc…

Transport Security Layer (TLS)

- Record Layer Protocol once C and S share keys they start communication

	Client Sends Message	Sever Sends Message
Encryption	k_C	k_S
MAC	k_C'	k_S'

- Sequence numbers prevent replay attacks
- TLS 1.2 used authenticate-then-encrypt (can be problematic)

Building Authenticated Encryption

Attempt 3: (Authenticate-then-encrypt) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \left\langle \mathrm{Enc}'_{K_E}(m \parallel t), \right\rangle \text{ where } \mathrm{t} = \mathrm{Mac}'_{K_M}(m)$$

Can be problematic for some CPA-Secure schemes…

Building Authenticated Encryption

Attempt 3: (Authenticate-then-encrypt) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \left\langle \mathrm{Enc}'_{K_E}(m \parallel t), \right\rangle \text{ where } \mathrm{t} = \mathrm{Mac}'_{K_M}(m)$$

$$Dec_K(c) =$$

1. $\widetilde{m} = \mathrm{Dec}'_{K_E}(c)$. If $\widetilde{m}$ is not padded correctly return "bad padding"
2. Parse as $m \parallel t$. If $\mathrm{Vrfy}'_{K_M}(m, t) = 1$ return m. otherwise output "authentication failure"

Building Authenticated Encryption

$$Dec_K(c) =$$

1. $\tilde{m} = \mathrm{Dec}'_{K_E}(c)$. If is not padded correctly return "bad padding"
2. Parse as $m \parallel t$. If $\mathrm{Vrfy}'_{K_M}(m,t) = 1$ return m. otherwise output "authentication failure"

It is hard to ensure that the error messages cannot be distinguished!

- Timing Attacks
- Debugging
- Generic Integration of MAC scheme with Encryption scheme?

Next Class: Multiparty Computation

- Finished with Katz and Lindell!
- Read Wikipedia entry on Secure Multi-party computation
- Read Katz and Lindell page 187-188 (commitment schemes)
  - OK, almost done ☺