TLS supports session setup using a "pre-shared key" (so full handshake not needed):

client server		client		Server
< full handshake >		first message (ClientHello +	Pre Shared Key (id)
< New Session Ticket (none, id)	\Rightarrow	vulnercable to	Encae (k,	olata)
Ţ		replay attack	Ľ	- derived from preshared key
			, Server	response

preshared key derived from session secrets, nonce, and il

fresh key KA-3B, KB-3A derived for rest of session (based on initial messages)

hegotiated ______identity of peer

Output of AKE protocol: (key, id)

<u>Authenticity</u>: Only party that knows key is id (i.e., the party identified by id) <u>Secrecy</u>: All parties other than client and id cannot distinguish key from candom (i.e., key is hidden) <u>Consistency</u>: If id also completes protoco), then it outputs (key, id client)

if we do not have client authentication, then

idetent is empty

Often also require <u>forward secrecy</u>: compromise of server in the future <u>cannot</u> affect secrecy of sessions in the past In TLS, server secret is a signing key - fresh Diffie-Hellman secret used for each session is fresh ("epheneral") Compromising signing key allows impersonation of server, but does not break secrecy of past sessions As we will see, not all AKE protocols provide forward secrecy

Very tricky to get right as we will see ... Just use TLS!

<u>AKE from PKE</u>: suppose server has certificate authenticating a public key for a PKE scheme (CCA-secure):

k₽K	Alice	<r ,="" certbank<="" th=""><th>Bank</th><th>skBaak</th><th>) Xields statically-secure AKE</th><th></th></r>	Bank	skBaak) Xields statically-secure AKE	
	11	$\underline{c \leftarrow Enc(pk_{Beak,}(r,k))}$		CertBank	(no forward secrecy)	
	Ť		ł	(r,k) - Decrypt (skeark, c) check that r'=r	Comprensive of skeank compromises all past	
	k, Bank	-	k,⊥	check that 'r'=r	Sessions	
			Ĺ	no client authentication		

If we do not encrypt the nonce r: replay attack possible (adversary replays messages from past session - e.g., "send Eve \$10") C nonce ensures <u>freshness</u>

Mutual	authentics	<u>stion</u> :	Bank has certificate identifying Allice has certificate identifyin		
	k ^e K	Alice	C == Enc(pkBenk, (k, "Alize"))		
			σ < Sign (skalice, (r,c, "Book certAlice	\downarrow	(k, Alice) - Dec(skoonk, c) check Alice motches is in certificate
		k, Bank		k, Alice	e check Alice's signature on (r, c, "Bank") under plate in certAlice

Above protocol provides static (no forward secrecy) mutual authentication

Most variants to this protocol are broken! AKE very delicate:

- Example: Suppose Alice encrypts (k, r) instead of (k, "Alice") like in the server-auth protocol above
 - Vulnerable to "identity misbinding" attack where Alice thinks she's talking to Bank but Bank Ahinks it's talking to Exc:

$$k \stackrel{\text{R}}{=} k \xrightarrow{\text{Alice}} (c \leftarrow \text{Enc}(pk_{\text{Bank}}, (k, r))) \xrightarrow{\text{Bank}} (k, r) \xrightarrow{$$

or ← Sign (Sk Ere, (r. c, "Bank")) => Bank thinks it's talking to Eve cert Eve

if Alice now sends "deposit this check into my account" to Bank,

Bank duposits it into Eve's account!

to observe that Eve did not break secrecy (she does not know k), but revertheless broke <u>consistency</u>

Above protocols supported by TLS 1.2, but deprecated in TLS 1.3 due to lack of forward secrecy

To get from	a) =8145 (4	use epheneral keys:	totally broken without signation	
		fresh public key	with pk an	'Y -
		pk, certionek, or a Sign (skeane, pk) for signature scheme	Provides one-sided authentication learn F	
k∉K		$C \leftarrow Enc(pk,k)$ Bonk Cert Bonk	((signature birds pk to Bank)	
	-		Forward secure since each pk used only once	
	L	$\int k \leftarrow Dec(sk, c),$	ound long-term secret is signing bey	
	k, Bank	k, 1 delete sk		

hardware security module (used to protect cryptographic secrets)

Problem: Does not provide "HSM security"

> Suppose adversory breaks into the bank and learns a single (pk', sk') poir with or < Sign (skBonk, pk')

L> Adversary can now impersonate the bank to any client:

adversary always use the message (pk', cert Bank, o) (defending against this requires freshness from client Scan decrypt keys for all clients that responds!

	ok			
Alice		KEK Provides Hi	im security: client chooses fresh	pk each time, so signature
Huce				as a "proof" that the other
	certBank o	J		signing key for id identified by
↓ ↓ ₽.\			Cert Bank	5 8 1 7
k, Banl	к k, ⊥		Bonk	

In many cases, also want to hide the endpoint (the id identified by cert) Possible by encrypting two keys (k, k') and using k' to encrypt certBunk

Diffie-Hellman key-exchange: suboriture Diffie-Hellman handshake for the PKE scheme (simpler) (TLS 1.2, 1.3)