Now that we have digital signatures, <u>Alice</u> ax <u>Bob</u>	י. י	of key exchange (u	sith active security)
		the oble to a active	
2 93	Completely V		
grö	r networ	k adverbary that can in	tercept and inject packets
gxy	9~3		
In addition, should guarantee that o	ne compromised session	should not affect of	ter <u>houst</u> sessions
- Alice <-> Eve should not c			
Authoritanted kay exchange (AVE):	ponsido serviche annu	+ active advocance	
Authenticated key exchange (AKE):	(anti-		happen have a l'idante
- Requires a "root of trust"	(contracte authority)	we need some bind	ing howard keys and identifies
Alice, pkalies			
cert _{Akee} CA	(one-time setup, at le	ast for duration of valid	ity period)
· · · · · · · · · · · · · · · · · · ·			
- the certificate bind	s Alice's public key pk	Alice to Alice's identify	
- Certificates typically have the fol	owing format (X509):		
- Subject (entity being authentic		\mathbf{h}	
- Public key (public key for sub			
- CA: identity of the CA issui	ng the certificate		
- Validity dates for certificate			
- CA's signature on certification		the browser and open	cating system have a set of hard-coded
0			and their respective public keys
Basic flow of Diffie-Hellman based f	KE:	(usually several hundred	
	Baak		
Alice x & Zp g X	y & Zp	Epublic- kay infra	
0 7			
		(%)	
< 3°, C← Encar (k', (cart Bunk, o)) k,k' ← H (9,9 [*] ,9 [*] ,9		
)) k,k' ← H (9,9 [*] ,9 [*] ,9		
< g ^y , c← EncAE(k', (certebook, σ) k, k' \leftarrow H(g, g ^x , g ^y , g'') $\sigma \leftarrow$ Sign (sk _{Bark} , (g		
$e^{g^{\vartheta}, C \leftarrow Enc_{AE}(k', (cent_{Buck}, \sigma))}$ derive $k, k' \leftarrow H(g, g^{\chi}, g^{\vartheta}, g^{\chi \vartheta})$)) k, k' ← H (g, g [*] , g [*] , g [*] , g [*])	,,9 ^x , 9 ⁸ , pk _{Bock}))	identifies many on Real (all of .)
$\langle g^{\vartheta}, C \leftarrow Enc_{AE}(k', (cert_{Buck}, \sigma')) \rangle$ derive k,k' \leftarrow H(g, g ^X , g ³ , g ^X) check σ is signature on (g, g ^X , g ³ , g ³ , p)) k, k' ← H (g, g ^x , g ⁸ , g ⁹	, g ^x , g ⁸ , pk _{Bock})) } <u>intuition</u> : Cert _{Book}	identifies server as Bonk (with pkBank)
$e^{g^{\vartheta}, C \leftarrow Enc_{AE}(k', (cent_{Buck}, \sigma))}$ derive $k, k' \leftarrow H(g, g^{\chi}, g^{\vartheta}, g^{\chi \vartheta})$)) k, k' ← H (g, g ^x , g ⁸ , g ⁹	,g ^x ,g ⁸ , pk _{Book})) } <u>intuition</u> : Cert _{Book} or binds	the session parameters (g, g ^x , g ³) to
$\langle g^{\vartheta}, C \leftarrow Enc_{AE}(k', Coert_{Back}, \sigma')$ derive $k, k' \leftarrow H(g, g^{X}, g^{\vartheta}, g^{X\vartheta})$ check σ is signature on $(g, g^{X}, g^{\vartheta}, p)$ under pk_{Back} is the public key)) k, k' ← H(g, g ^X , g ^Y , g ^Y , g ^Y	,,g ^x , g ⁸ , pk _{Book})) } <u>intuition</u> : Cert _{Book} o⊤ binds the pu	.
$\langle g^{\flat}, C \leftarrow Enc_{AE}(k', (cert_{Back}, \sigma')) \rangle$ derive $k, k' \leftarrow H(g, g^{X}, g^{\vartheta}, g^{X\vartheta})$ check σ is signature on $(g, g^{X}, g^{\vartheta}, p)$ under pk_{Back} is the public key)) k, k' ← H(g, g ^X , g ^Y , g ^Y , g ^Y	,,g ^x , g ⁸ , pk _{Book})) } <u>intuition</u> : Cert _{Book} o⊤ binds the pu	the session parameters (g, g ^x , g ³) to
derive k,k' ← H(g, g ^X , g ³ , g ^{X3}) derive k,k' ← H(g, g ^X , g ^X , g ³ , g ^{X3}) derive k,k' ← H(g, g ^X , g ^X , g ^X , g ^{X3}) derive k,k' ← H(g, g ^X , g)) k, k' ← H(g, g ^x , g ^y , g ^y , g ^y)	,,g ^x , g ⁸ , pk _{Bock})) } <u>intuition</u> : Cert _{Boak} or binds the pu not vice versa!)	the session parameters (g, g ^x , g ³) to
derive k,k' ← H(g, g ^X , g ³ , g ^X) derive k,k' ← H(g, g ^X , g ³ , g ^X) check of is signature on (g, g ^X , g ³ , p under pkBoak is the public key <u>End of protocol</u> : Alice kaows she is ² "one-sided AKE")) k, k' ← H(g, g ^X , g ^B , g ^Y) J ← Sign (sk _{Baak} , (g J Secsion key k' kBoak) identified by centBank identified by centBank holking to Boank (but g — most common mode	, g ^x , g ^u , pk _{Bock}))] <u>intuition</u> : Cert _{Boak} o binds the pu <u>tot</u> vice versa?) 2 on the web	the session parameters (g, g ^x , g ^y) to blic key identified by cert _{Bank}
derive k,k' ← H(g, g ^X , g ³ , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X) derive k,k' ← H(g, g ^X) derive k,k' ← H(g,)) k, k' ← H(g, g ^X , g ^B , g ^Y) J ← Sign (sk _{Baak} , (g J Secsion key k' kBoak) identified by centBank identified by centBank holking to Boank (but g — most common mode	, g ^x , g ^u , pk _{Bock}))] <u>intuition</u> : Cert _{Boak} o binds the pu <u>tot</u> vice versa?) 2 on the web	the session parameters (g, g ^x , g ^y) to blic key identified by cert _{Bank}
derive k,k' ← H(g, g ^X , g ³ , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k' ← H(g, g ^X) derive k,k' ← H(g, g ^X) derive k,k' ←)) k, k' ← H(g, g ^X , g	1,9 ^x , 3 ⁸ , pk _{Bock}))	the session parameters (g, g ^X , g ³) to blic key identified by cert _{Bank} n't invent your own AKE postoco)!
derive k,k' ← H(g, g ^X , g ³ , g ^{X3}) derive k,k' ← H(g, g ^X , g ³ , g ^X , g)) k, k' ← H(g, g ^X , g	,,g ^x ,g ³ , pk _{Bock})) } <u>intuition</u> : Cert Book or binds the pu not vice versa!) 2 on the web WAYS USE TLS 1.3 - Do parted ciphersuites	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n ⁴ invent your own AKE protocol! older systems / foreign systems may prefer different
derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³) derive k,k' ← H(g, g ^X , g ³ , g ^X , g ³ , g ^X , g ³ , p under pk_{bask} is the public key <u>End of protocol</u> : Alice knows she is ~ <u>``one-sided AKE' Sossis of TLS 1.3 hand shake (</u>)) k, k' ← H(g, g ^X , g	1,9 ^x , 3 ⁸ , pk _{Bock}))	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n't invent your own AKE protoco)! older systems / foreign systems may prefer different -256) cipturs
G ⁸ , C← EncAE(k', (cert _{Back} , or derive k,k' ← H(g, g ^x , g ³ , g ^x 3) derive k,k' ← H(g, g ^x , g ³ , g ^x 3) check of is signature on (g, g ^x , g ³ , p under pkBoak is the public key End of protocol: Alice kaows she is)) k, k' ← H(g, g ^X , g	,,g ^x ,g ³ , pk _{Bock})) } <u>intuition</u> : Cert Book or binds the pu not vice versa!) 2 on the web WAYS USE TLS 1.3 - Do parted ciphersuites	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n ⁴ invent your own AKE protocol! older systems / foreign systems may prefer different
)) k, k' ← H(g, g ^X , g	1,9 ^x , 3 ⁸ , pk _{Bock}))	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n't invent your own AKE protoco)! older systems / foreign systems may prefer different -256) cipturs
G ⁸ , C← EncAE(k', (cert _{Buk} , or derive k,k' ← H(g, g ^X , g ³ , g ^X) derive k,k' ← H(g, g ^X , g ^X) derive k,k')) k, k' ← H(g, g ^X , g	1,9 ^x , 3 ⁸ , pk _{Bock}))	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n't invent your own AKE protocol! Older systems / foreign systems may prefer different -256) cipturs Older versione of TLS vulnerable to
G ⁸ , C← EncAE(k', (cert _{Buk} , of derive k, k' ← H(g, g ^X , g ³ , g ^X) derive k, k' ← H(g, g ^X , g ^X , g ³ , g ^X , g ³ , g ^X , g ³ , g ^X , g ^X , g ³ , g ^X)) k, k' ← H (g, g ^X ,	1,9 ^x , 3 th , pk _{Back}))	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n ⁴ invent your own AKE postoco)! Older systems / foreign systems may prefer different (-256) cipturs older versions of
G ⁸ , C← EncAE(k', (cert _{Back} , or derive k,k' ← H(g, g ^x , g ³ , g ^x 3) check of is signature on (g, g ^x , g ³ , p under pkBaak is the public key End of protocol: Alice kaows she is - "one-sided AKE Sasis of TLS 1.3 hand shake Client Hello DH Key-Share ChertHello DH Key-Share Certificate (encrypted))) k, k' ← H(g, g ^X , g	1,9 ^x , 3 th , pk _{Back}))	the session parameters (g, g ^x , g ^y) to blic key identified by cert Bank n't invent your own AKE protocol! Older systems / foreign systems may prefer different -256) cipturs Older versione of TLS vulnerable to

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)	\implies	vulnercuble to	- (1	olata)
1		replay attack		derived from preshared key
			, server (esponse

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 form random (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 ^e K	Alice	<r ,="" certbank<="" th=""><th>Bank</th><th>skBank</th><th>) Yields statically-secure AKE</th><th></th></r>	Bank	skBank) Yields statically-secure AKE	
	H « CC	$C \leftarrow Enc(pk_{Beak}, (r, k))$	Day	Cert Bank	(no forward secrecy)	
	¥		ł	(r,k) - Decrypt (skeark, C) check that r'=r	Compromise of skeant compromises all pass	r
	k, Bank		k,⊥	check that r'=r	Sessions	
			Ĺ	no client autentication		

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

Mutual	authentic	ation:	Bank has certificate identifying Allice has certificate identifyin		
	k ^e K	Alice	C = Enc(pkBenk, (k, "Alize"))		
		L k, Bank	σ ← Sign (sk _{Alice,} (r,c, *Baak cert _{Alice}	r)) k, Alice	(k, Alice) - Dec(SkBank, c) Check Alice matches id in certificate
				- Set lines	e check Alice's signature on (r,c, "Bank") under platice 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:

σ ← Sign(sk Ere, (r, c, 'Bank")) ⇒ Bank thinks it's talking to Eve

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

Bank deposits it into Eve's account!

to observe that Eve did not break secrecy (she does not know k), but revertheless, broke authenticity

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

To get for	word secrecy,	use <u>epheneral keys</u> : Stresh public key	totally broken without signation adversary can replace with pk an	
k ♣ K		pk, certBook, O < Sign (skeank, pk) for signature scheme C < Brc(pk, k) Book CertBook	Provides one-sided authentication flear f (signature binds pk to Bank) Forward secure since each pk used only once	
	L k, Bank	$ \begin{array}{c} \downarrow \qquad k \leftarrow Dec(sk, c), \\ k, \bot \qquad delete sk \end{array} $	ound long-term secret is signing bey	

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 (sk Bank, 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 Can decrypt keys for all clients that responds!

	ok			
Alice		KEN Provides HS	M 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
↓ ↓ ₽.\			CertBank	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)