no more downgrades: protecting TLS from legacy crypto

karthik bhargavan

INRIA

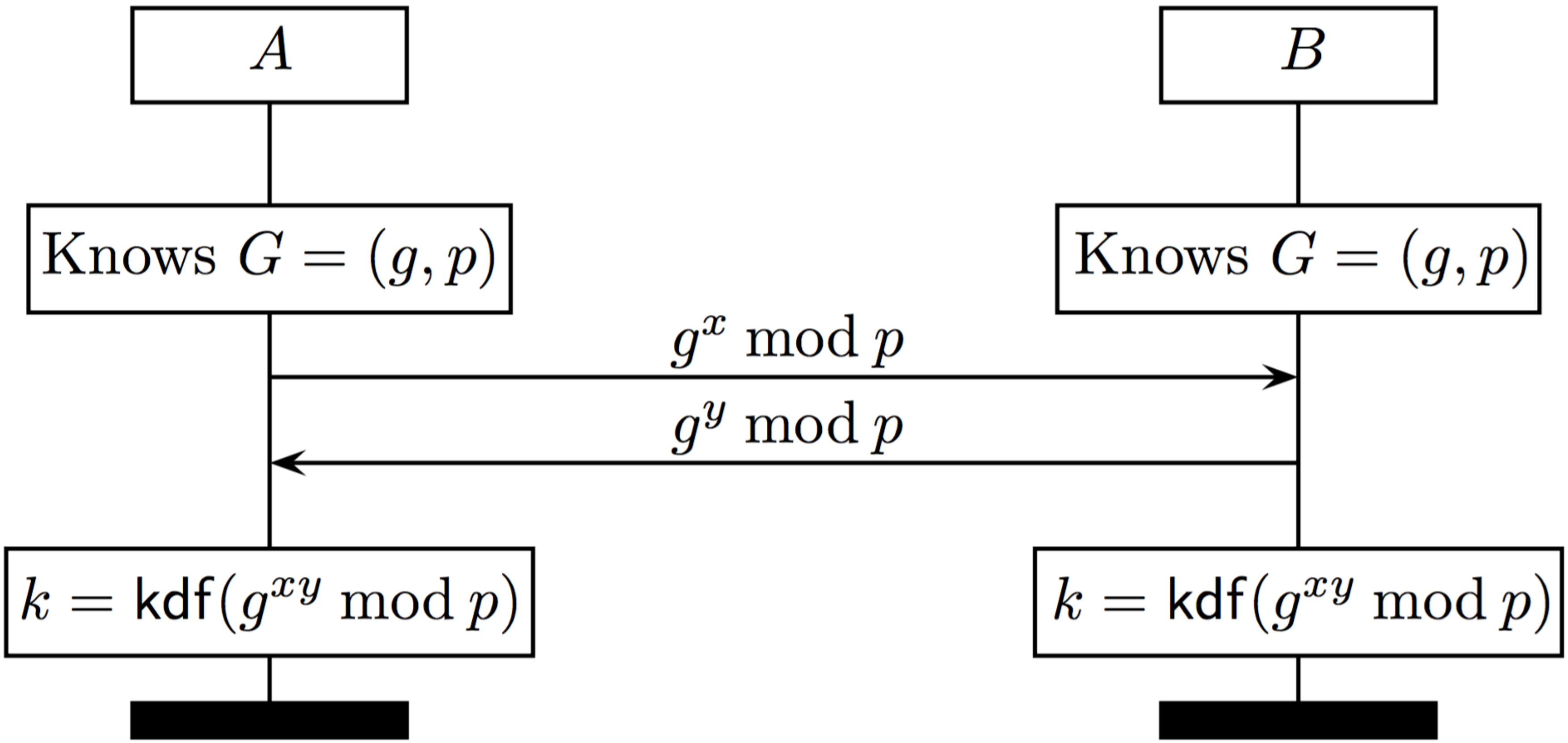
joint work with:
g. leurent, c. brzuska, c. fournet,
m. green, m. kohlweiss, s. zanella-beguelin

TLS: a long year of downgrade attacks

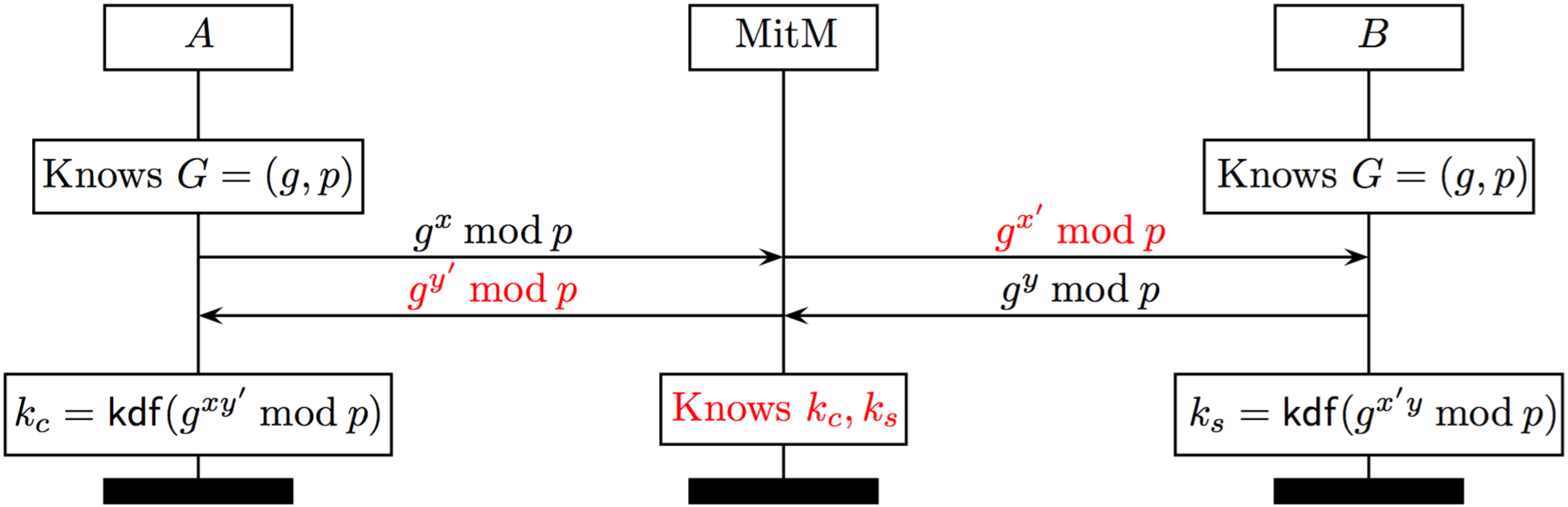
• POODLE	TLS 1.2 → SSLv3	[Dec'14]
• FREAK	RSA-2048 → RSA-512	[Mar'15]
• LOGJAM	DH-2048 → DH-512	[May'15]
• BLEICH?	RSA-Sign → RSA-Enc	[Aug'15]
• SLOTH	RSA-SHA256 → RSA-MD5	[Jan'16]

- What's going on?
- How do we fix it in TLS 1.3?
  - More details: mitls.org, sloth-attack.org

Anonymous Diffie-Hellman (ADH)

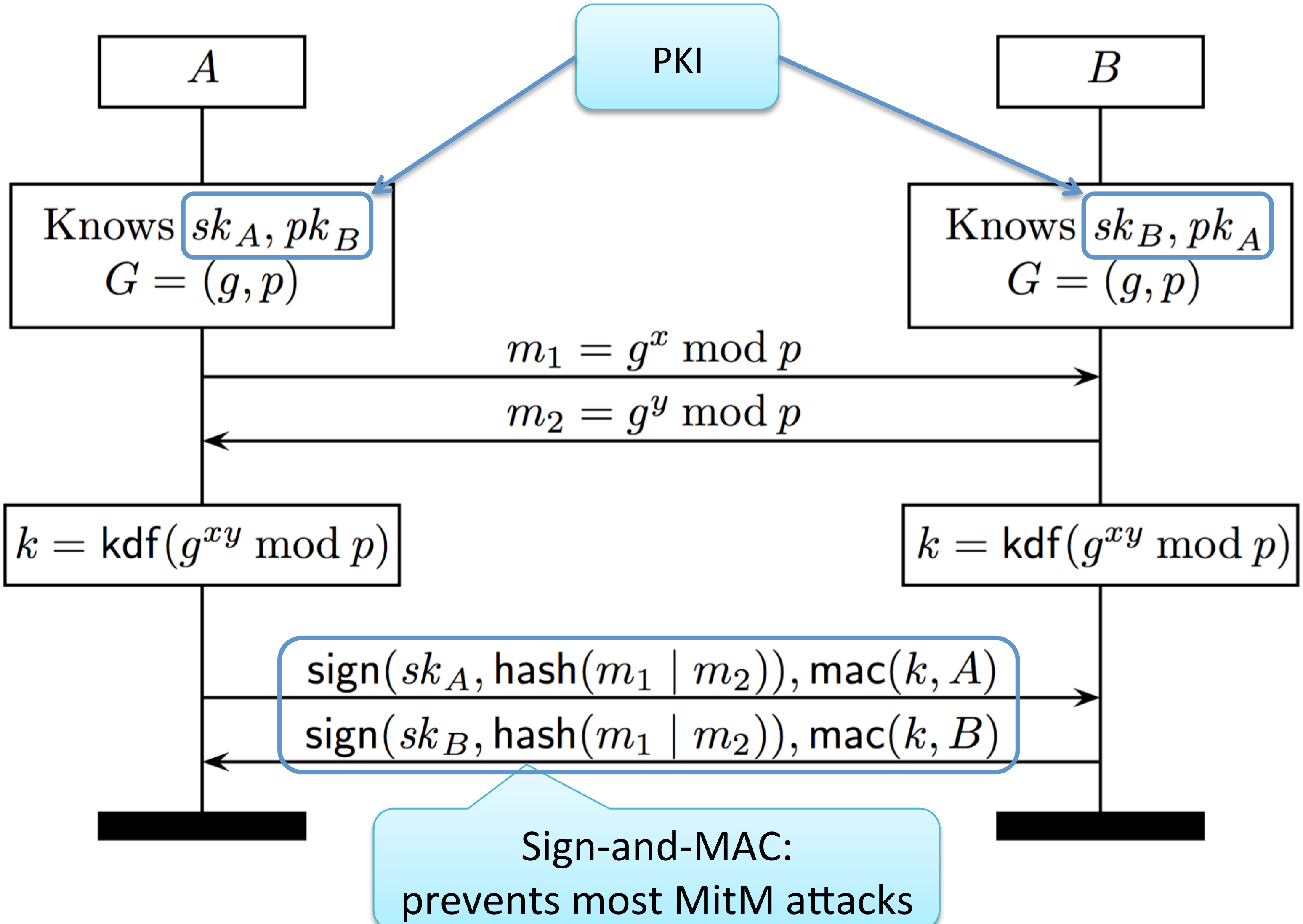


Man-in-the-Middle attack on ADH

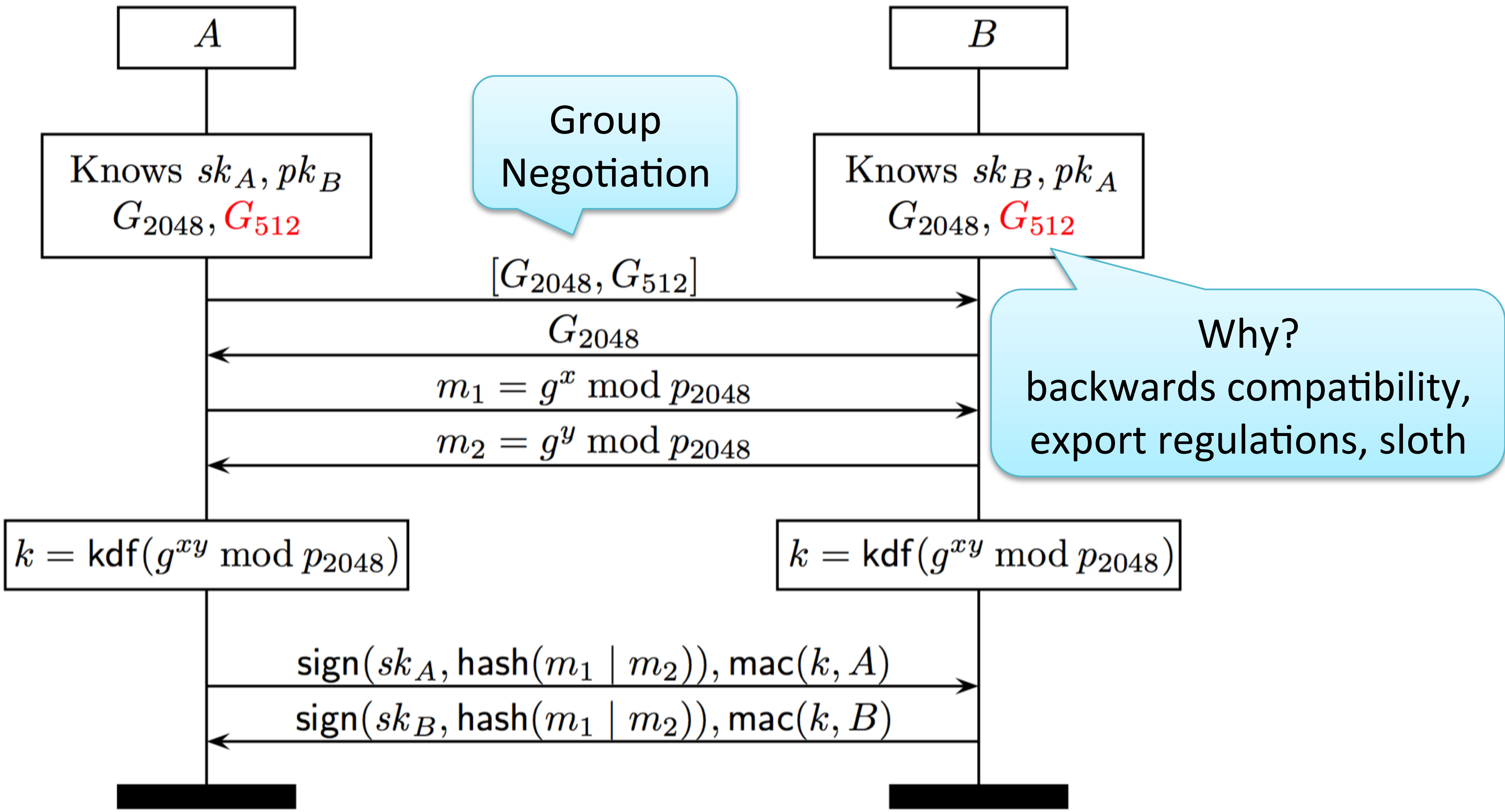


Active Network Attacker
or Malicious Peer

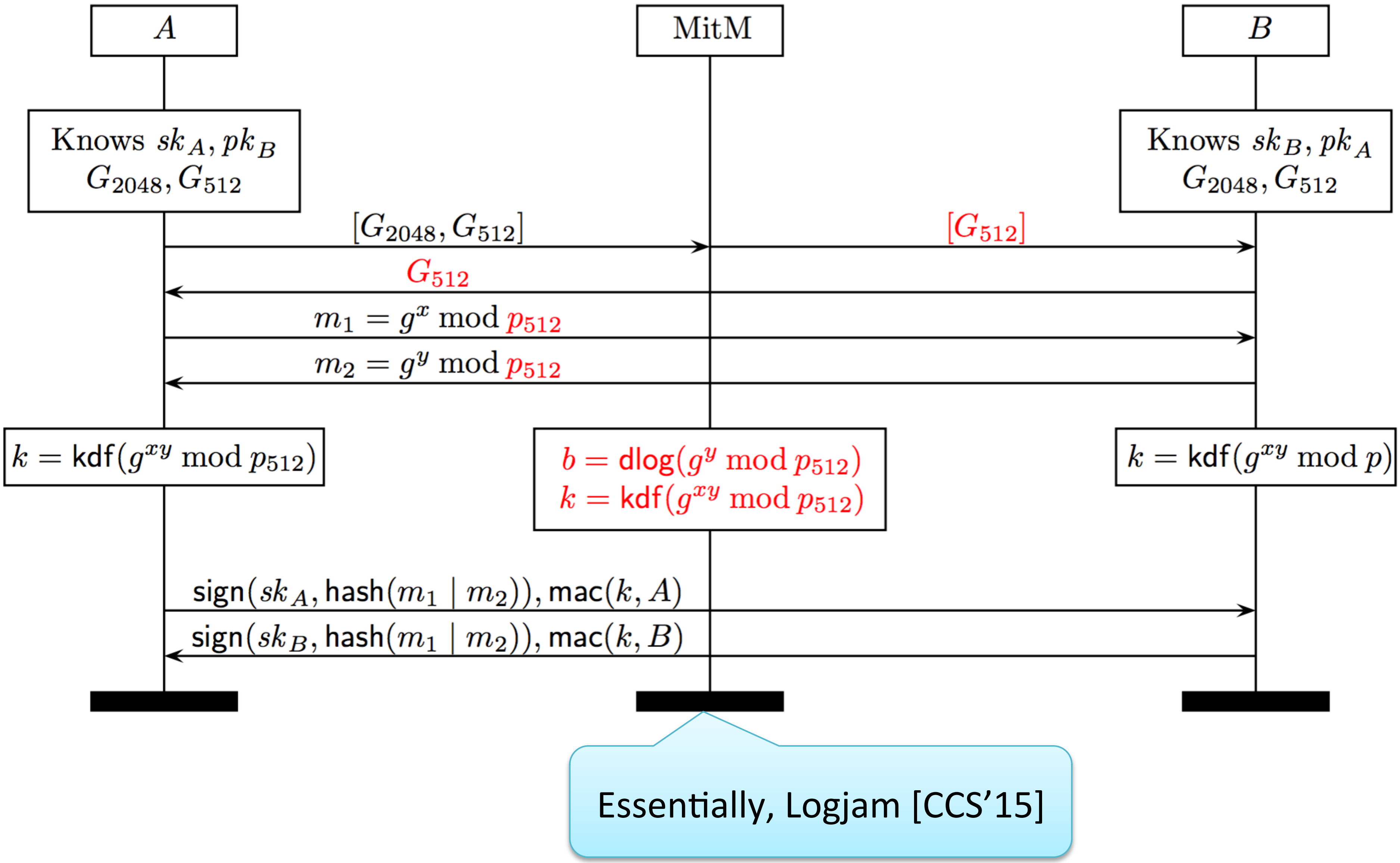
Authenticated DH (SIGMA)



Agility: Negotiating DH Groups



MitM Group Downgrade Attack



MACs for Downgrade Protection

- TLS: mac the full transcript to prevent tampering
  - **mac**(k, [G_{2048},G_{512}] | G_{512} | m_1 | m_2)
  - but it is too late, because we already used G_{512}

    k = **kdf(**g^{xy} mod p_{512})
  - so, the attacker can forge the **mac**

- *The TLS downgrade protection mechanism itself depends on downgradeable parameters.*
  - hence, the only fix is to find and disable all weak parameters: groups, curves, mac algorithms,…

Signing Handshake Transcripts

- IKEv1: both A and B sign the offered groups
  - **sign**(sk_B, **hash**($[G_{2048}, G_{512}] \mid m_1 \mid m_2$))
  - no agreement on chosen group!
- IKEv2: each signs its own messages
  - **sign**(sk_A, **hash**($[G_{2048}, G_{512}] \mid m_1$))
  - **sign**(sk_B, **hash**($G_{512} \mid m_2$))
  - no agreement on offered groups!
- SSH-2 and TLS 1.3: sign everything
  - **sign**(k, **hash**($[G_{2048}, G_{512}] \mid G_{512} \mid m_1 \mid m_2$))
  - works! (.... or does it?)

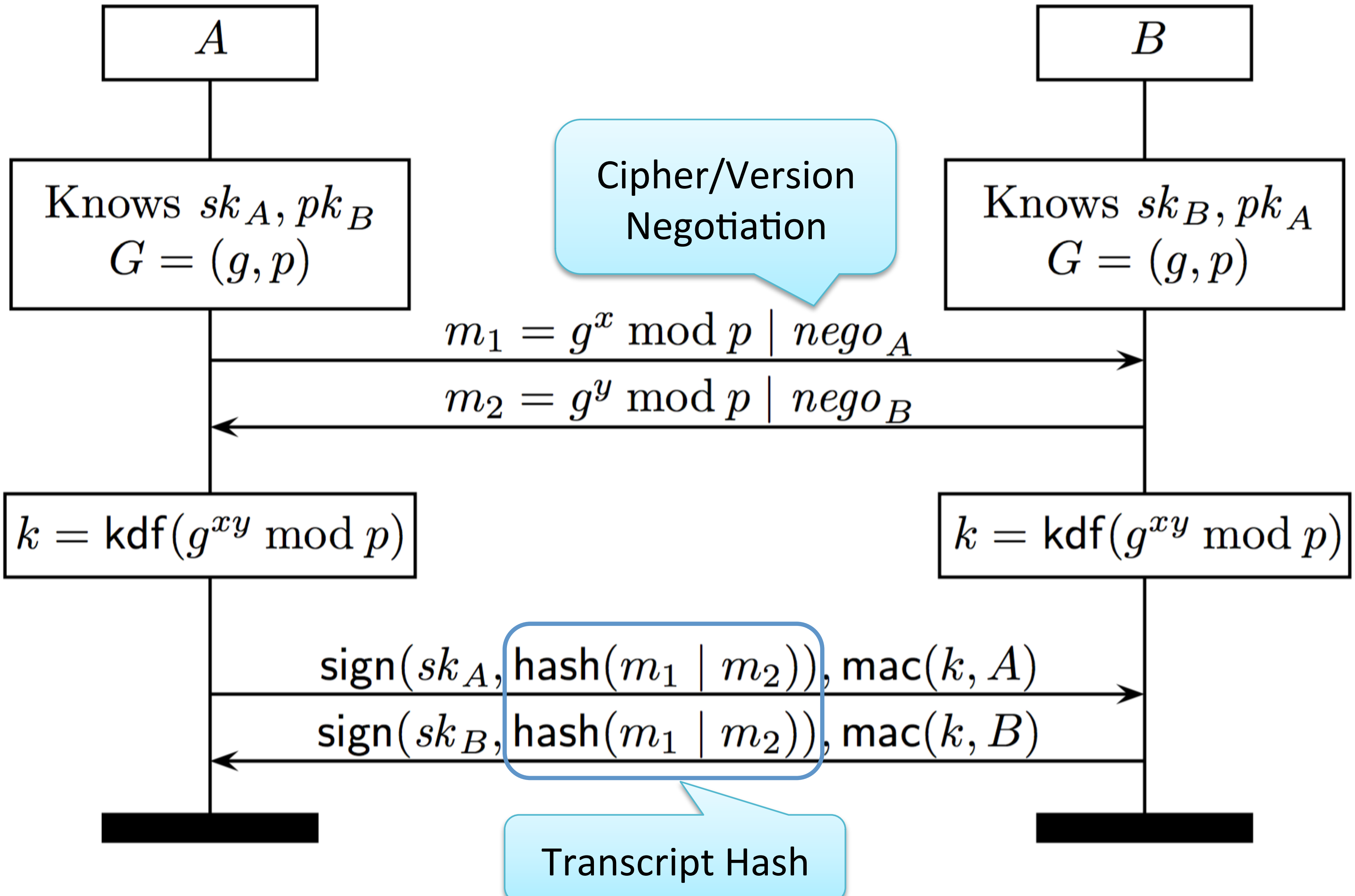
SLOTH: Transcript Collision Attacks

- SSH-2 and TLS 1.3: sign the full transcript
  - **sign**(k, **hash**($[G_{2048}, G_{512}] \mid G_{512} \mid m_1 \mid m_2$))
  - what if **hash** were a weak hash function?

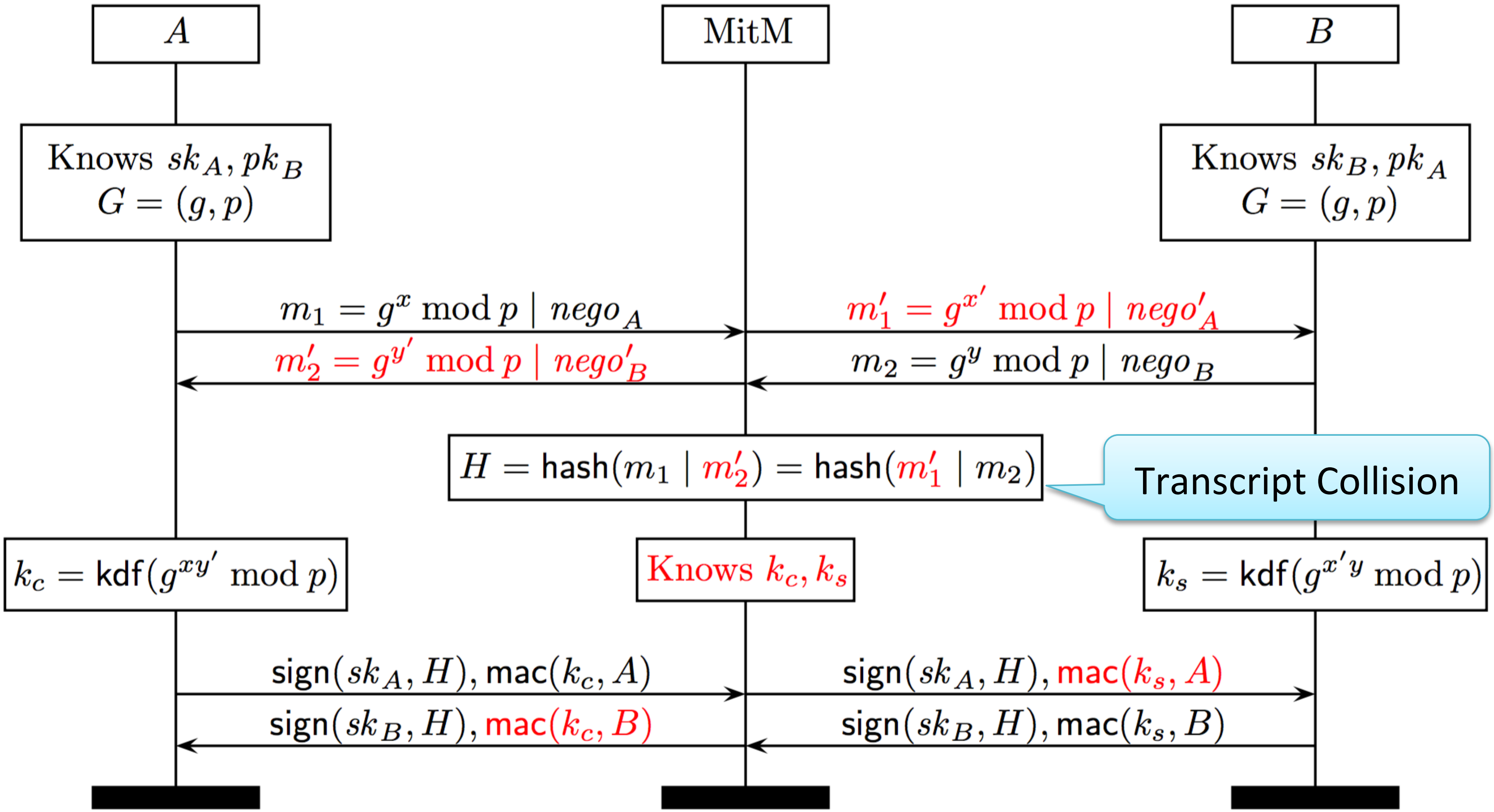
- How weak can **hash** be?
  - do we need collision resistance?
  - do we only need 2nd preimage resistance?

- SLOTH: *transcript collision* attacks break key protocol guarantees in TLS, IKE, SSH
  - so yes, we do need collision resistance

Authenticated DH with Negotiation



A Transcript Collision Attack



Computing a Transcript Collision

$$\textbf{hash}(m_1 \mid m'_2) = \textbf{hash}(m'_1 \mid m_2)$$

- We need to compute a collision, *not a preimage*
  - Assume we know or control the black bits, how easy would it be to compute the red bits?
  - This is usually called a **generic collision**

- If we're lucky, we can set up a **shortcut** collision
  - **Common-prefix**: collision after a shared prefix
  - **Chosen-prefix:** collision after attacker-controlled prefixes

Primer on Hash Collision Complexity

- MD5: known hash collision complexities
  - **MD5** generic collision: 2^{64} hashes (birthday)
  - **MD5** chosen-prefix collision: 2^{39} hashes (1 hour)
  - **MD5** common-prefix collision: 2^{16} hashes (seconds)
- SHA1: estimated hash collision complexities
  - **SHA1** generic collision: 2^{80} hashes (birthday)
  - **SHA1** chosen-prefix collision: 2^{77} hashes (?)
  - **SHA1** common-prefix collision: 2^{61} hashes (?)
- Collision resistance for other hash constructions
  - **MD5**(x) | **SHA1**(x) not much better than SHA1
  - **HMAC-MD5**(k,x) not much better than MD5
  - **HMAC-SHA256**(k,x) truncated to 2N bits takes 2^{N} hashes

A Generic Transcript Collision

$$\textbf{hash}(m_1 \mid m'_2) = \textbf{hash}(m'_1 \mid m_2)$$

- Suppose **hash = MD5**

- *Problem*: attacker must compute m'_1 before seeing m_2
- So, suppose *B* uses predictable m_2 (no freshness)

- We can break the protocol with 2^{64} **MD5** hashes
  - Still impractical for academics, but almost feasible

A Common-Prefix Transcript Collision

$$\textbf{hash}(m_1 \mid m'_2) = \textbf{hash}(m'_1 \mid m_2)$$

$$\textbf{hash}([len_1 \mid g^x \mid nego_A] \mid [len'_2 \mid g^{y'} \mid nego'_B) =$$
$$\textbf{hash}([len'_1 \mid g^{x'} \mid nego'_A] \mid [len_2 \mid g^y \mid nego_B])$$

- Suppose len_2 is predictable but m_2 is not
- *Problem*: need to compute m'_1 after m_1 but before m_2
- But suppose $nego_A$, $nego_B$ allow arbitrary data

- We can break the protocol with 2^{39} **MD5** hashes
  - About 1 hour on a powerful workstation

A Common-Prefix Transcript Collision

$$\textbf{hash}(m_1 \mid m'_2) = \textbf{hash}(m'_1 \mid m_2)$$

- Compute a *chosen-prefix* MD5 collision C_1, C_2:

  $\textbf{hash}([len_1 \mid g^x \mid nego_A] \mid [len'_2 \mid g^{y'} \mid C_1) =$

  $\textbf{hash}([len'_1 \mid g^{x'} \mid$ filler bytes $\mid C_2])$

- Then, by carefully choosing m'_1, m'_2, we get

  $\textbf{hash}(m_1 \mid m'_2) =$

  $\textbf{hash}([len_1 \mid g^x \mid nego_A] \mid [len'_2 \mid g^{y'} \mid C_1 \mid m_2]) =$

  $\textbf{hash}([len'_1 \mid g^{x'} \mid$ <filler bytes> $\mid C_2] \mid m_2) =$

  $\textbf{hash}(m'_1 \mid m_2)$

SLOTH in TLS 1.2

- TLS 1.2 supports MD5-based signatures!
  - Surprising, because TLS <= 1.1 only supported **MD5 | SHA1**
  - Even if the client and server prefer **RSA-SHA256**, the connection can be downgraded to RSA-MD5!
- We can break TLS 1.2 client signatures
  - Takes 1 hour/connection on a 48-core workstation
  - Practical-ish: we can always throw more cores/ASICs at it
- TLS 1.2 server signatures are harder to break
  - Irony: the same flaw that enables Logjam blocks SLOTH
  - Needs 2^X prior connections + 2^{128-X} hashes/connection
  - Not practical for academics, maybe doable by govt?

Other SLOTH Attacks

- Reduced security for TLS 1.*, IKEv1, IKEv2, SSH
  - via downgrades + transcript collisions
  - these are protocol flaws, not implementation bugs
  - *Mitigation*: fully disable MD5 (and SHA1?)

http://sloth-attack.org

Protocol	Property	Mechanism	Attack	Collision Type	Precomp.	Work/conn.	Preimage	Wall-clock time
TLS 1.2	Client Auth	RSA-MD5	Impersonation	Chosen Prefix		2^{39}	2^{128}	48 core hours
TLS 1.3	Server Auth	RSA-MD5	Impersonation	Chosen Prefix		2^{39}	2^{128}	48 core hours
TLS 1.0-1.2	Channel Binding	HMAC (96 bits)	Impersonation	Generic		2^{48}	2^{96}	80 GPU days
TLS 1.2	Server Auth	RSA-MD5	Impersonation	Generic	2^X conn.	2^{128-X}	2^{128}	
TLS 1.0-1.1	Handshake Integrity	MD5 | SHA-1	Downgrade	Chosen Prefix		2^{77}	2^{160}	
IKE v1	Initiator Auth	HMAC-MD5	Impersonation	Generic		2^{65}	2^{128}	
IKE v2	Initiator Auth	RSA-SHA-1	Impersonation	Chosen Prefix	2^{77}	0	2^{160}	
SSH-2	Exchange Integrity	SHA-1	Downgrade	Chosen Prefix		2^{77}	2^{160}	

Downgrade Resilience in TLS 1.3

- Both client and server sign the full transcript with strong signature and hash algorithms
  - TLS 1.3 client/server authentication with RSA-MD5 is completely broken by SLOTH, so we got rid of MD5
- Good news: We can prove that the downgrade protection sub-protocol within TLS 1.3 works
  - New crypto definitions, proofs, in draft paper
- What do we do about version downgrade?
  - Can an attacker downgrade TLS 1.3 to TLS 1.2 and remount Logjam, SLOTH etc?

Version Downgrade Resilience

- To detect downgrades, clients need to check that the server chose the highest common version
  - TLS 1.3 server signatures do cover client+server versions
  - But TLS <= 1.2 server signatures **do not** cover the version

- How do we patch TLS <= 1.2 to prevent downgrades?
  - Protocol extensions or SCSVs cannot help; the attacker will delete them
  - Look away: we put the max server version in the server nonce because it is signed in all versions of TLS

- Good news: we can now prove version downgrade resilience for clients and servers that support TLS 1.0-1.3
  - only for signature ciphersuites, not if they support RSA

Final Thoughts

- Legacy crypto is strangely hard to get rid of, but we have to keep trying to kill them
- Key exchanges in Internet protocols do rely on collision resistance, don't let anyone tell you otherwise!
- We can and should design downgrade resilient protocols
- Implementation bugs can undermine all protections; so we need to verify protocol code

- More details, papers, demos are at:
  - http://mitls.org
  - http://sloth-attack.org