Downgrade Resilience in Key-Exchange Protocols

Ruth Ng

TLS Crypto Seminar, Winter 2019, UC San Diego

Overview

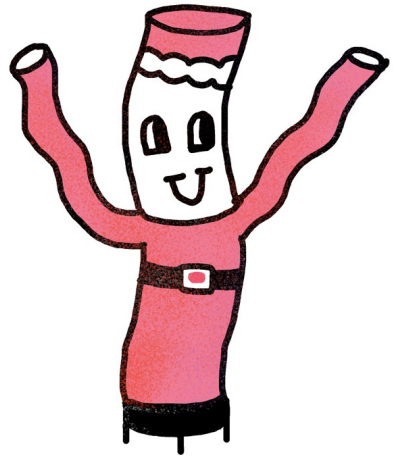
- High-level summary of the whole paper:
  - Motivate the study of downgrade resilience
  - (Very) brief discussion on modelling downgrade resilience
  - Touch on all the authors' results
- Case Study: Downgrade attacks in TLS
  - Logjam as a downgrade attack on TLS 1.2 and TLS 1.3 (Draft 10)
  - Mitigation of this attack in TLS 1.3
- Broader discussion on crypto standards in practice
  - Inspired by talks at RSAC 2019

Overview of [BBF+16]: Motivation

- A key feature of real-world key exchange protocols is that can run in different *modes*. This gives the protocol flexibility in handling different types of network devices.
  - E.g. different DH-groups, different primitives, up-to-date vs legacy versions
- This presents a challenge when trying to model it as a key exchange protocol and prove its security.
- So far, we know how to:
  - Prove that TLS running in *one* particular mode is secure (e.g. [JKSS12])
  - Prove that TLS is secure, as long as *any* secure mode is chosen (e.g. [BPK+14])
- But none of these models discuss *how* a secure mode is chosen
  - We want to guarantee that the “preferred common mode” is being used.

Overview of [BBF+16]: Model

- A (very) simplified version of the [BBF+16] model of key-exchange protocols
  - Two parties with different roles: initiator *I* and responder *R*
  - Their goal is to set up two "partnered" sessions, π_I and π_R
  - In the absence of an adversary, the sessions will send each other messages until they complete and agree on some set of (protocol specific) session variables *X* (e.g. mode, version number).
  - An adversary interacts with the sessions via oracles. He can (1) initialize sessions (2) send messages to sessions and observe their response and (3) corrupt sessions and look at key material



Either a description of how flexible this model is, or how hand-wavy my explanation of this model is going to be. You pick.

Overview of [BBF+16]: Model

- In place of “correctness”, we define the following security goals for key-exchange protocols in the presence of an adversary:
  - Goal #1: (Uniqueness) A session can be partnered with at most one other session, with an opposite role
  - Goal #2: (Partnering security) A completed session cannot be unpartnered
  - Goal #3: (Multi-mode authentication) A completed session must have a partner session which agrees on X
  - Goal #4: (Key indistinguishability) An adversary cannot distinguish real session keys from random ones in uncorrupted sessions.
- Finally, we define our main goal: (downgrade resilience) A completed session must have a partnered session who share a mode which is not downgraded

Overview of [BBF+16]: Downgrades

- But what is a “downgraded” session mode?
- Each key exchange protocol defined in a multi-mode setting defines an algorithm *Nego* which takes as input the client and server configurations and returns a set of “preferred” modes.
  - For now, let’s assume that a configuration is a set of modes.
- A downgrade occurs when an adversary can alter the session mode to be inconsistent with *Nego*.
  - Here, $Mode/Mode_A$ are the modes that the algorithm can end up using without/ with adversarial involvement.

Cfg_{Client}	Cfg_{Server}	$Nego(Cfg_{Client}, Cfg_{Server})$	$Mode$	$Mode_A$	Downgrade?
{A,B}	{A,B}	{A}	A	B	Yes
{A,C,D}	{B,C,D}	{C,D}	C	D	No
{A,C,D}	{B,C,D}	{C,D}	{C,D}	D	No
{A,C,D}	{B,C,D}	{C}	C	D	Yes

Overview of [BBF+16]: Results

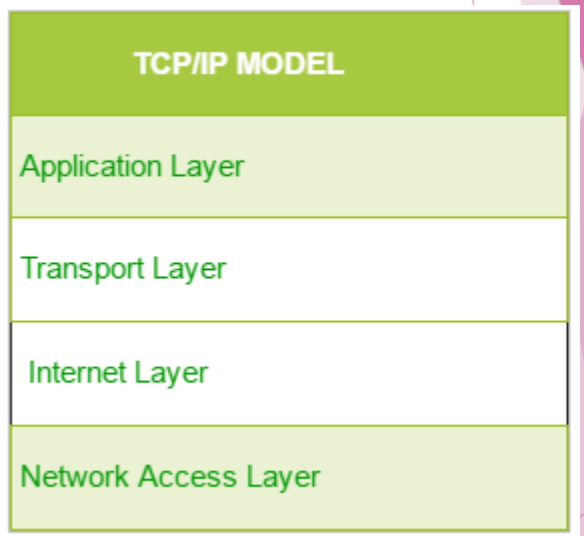
- Define a model to formalize downgrade resilience
- Formalize the following “folklore” result:

“*To prevent an attack on a particular protocol mode, it is sufficient to deactivate the configurations that lead to its negotiation*”

  - More specifically, “when downgrade security holds, only the security of modes which can be negotiated in the absence of an adversary matters. That is, if peers support insecure modes, but with such low priority that they never negotiate them on their own, then these modes do not affect security in the presence of an adversary.”

Overview of [BBF+16]: Results

- Model real-world protocols at all levels of the protocol stack:
  - SSHv2: Application layer protocol
  - zRTP: Transport layer protocol used to secure phone calls via UDP
  - IKEv2: Internet layer protocol used in IPSec
  - TLS: Between application and transport layers
- Analyze the security of real-world protocols:
  - Stronger result for SSHv2
  - Vulnerability/ Patch for zRTP and IKEv2
  - Confirm that TLS 1.0-1.2 and TLS 1.3 (draft 10) is not secure, provide simple patch used in TLS 1.3 (draft 11)



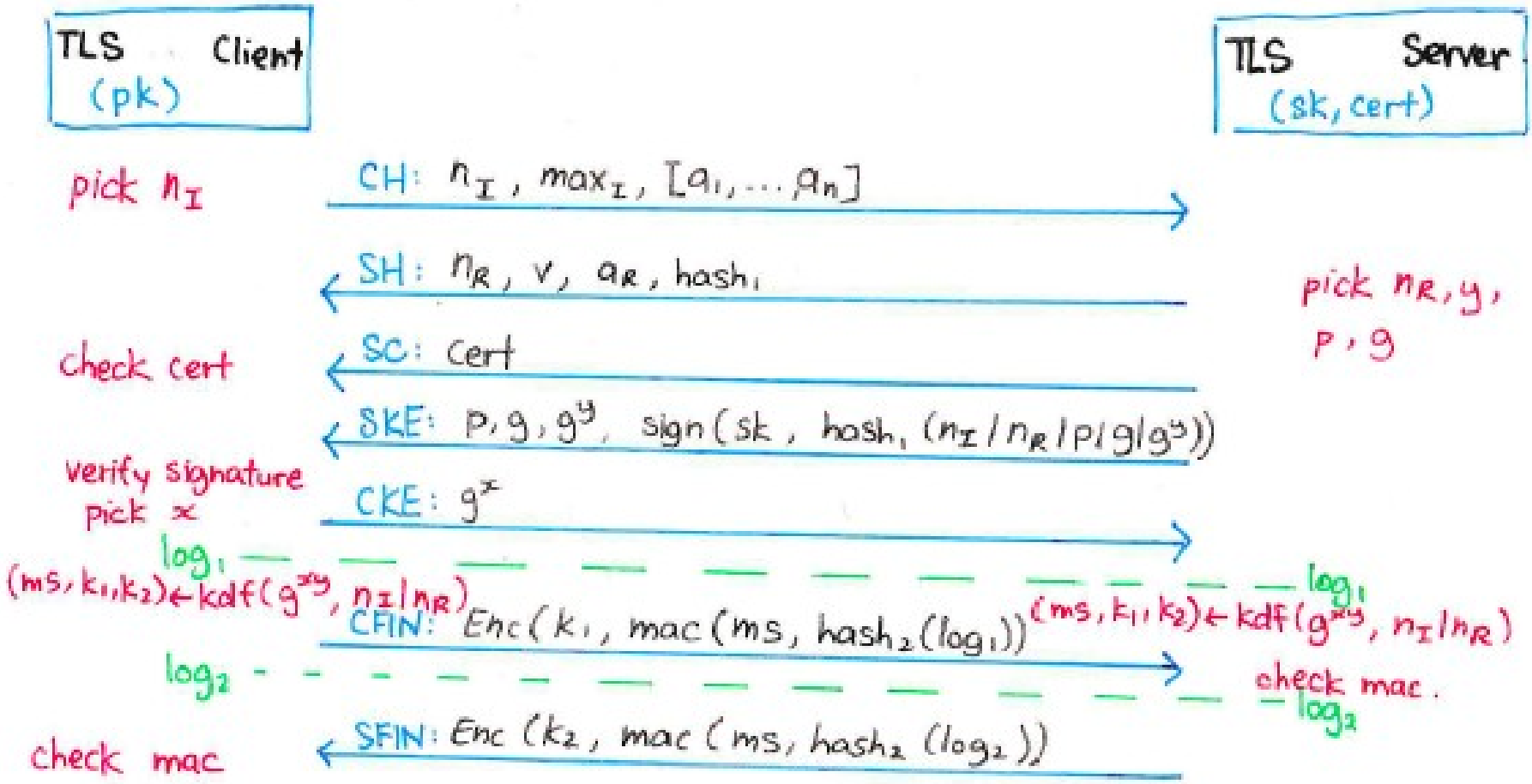
TLS Case Study

- For the rest of our discussion on downgrade resilience, we will use TLS as a case study to show the strength of the model in formalizing and improving downgrade resilience of real-world protocols.
- All this discussion is limited to TLS-(EC)DHE, with no client authentication.
  - In other words, we come from the point-of-view of a server offering TLS 1.3, trying to authenticate itself to clients and start sessions with them.
- TLS 1.2 and 1.3 (draft 10) offer some downgrade resilience, under stronger conditions but these conditions are not realistic
  - Logjam constitutes a downgrade attack on both TLS 1.2 and TLS 1.3 (draft 10)
- Small change to TLS 1.3 which assures downgrade resilience under more reasonable assumptions

TLS 1.2

- We formalize TLS 1.2 using the notation from [BBF+16]. For simplicity, we will consider the variant of TLS 1.2 with no client authentication.
- We will also assume that the server has a signing key *sk* and certificate *cert*. The client has the corresponding public key *pk* and a way to check *cert* (e.g.via a CA's public key).
- In the client hello, the client will provide a number of configurations, $[a_1 \ldots a_n]$. Each of these should specify the algorithms to be used later in the protocol (mac, hash_2, kdf, enc, sign).
- Logjam is a downgrade attack on TLS 1.2, where a configuration "DHE" is downgraded to one using DHE-EXPORT.
  - For simplicity, we will denote the configuration DHE instead of the full tuple.

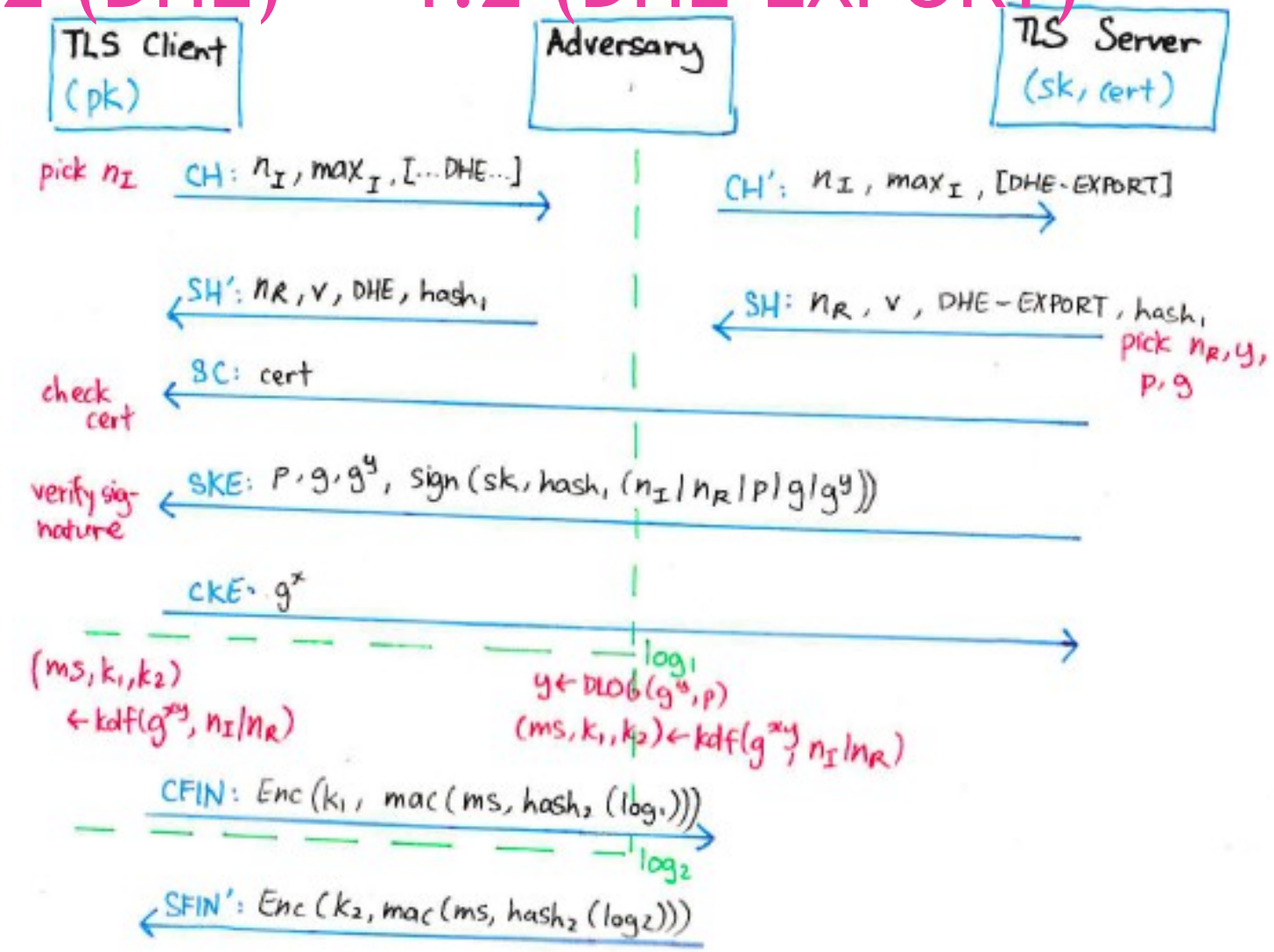
TLS 1.2


TLS Client
(pk)
TLS Server
(sk, cert)
pick n_I
CH: n_I, max_I, $[a_1, \ldots a_n]$
SH: n_R, v, a_R, $hash_1$
pick n_R, y, p, g
check cert
SC: cert
SKE: p, g, g^y, sign(sk, $hash_1$, (n_I / n_R / p / g / g^y))
verify signature
pick x
CKE: g^x
log_1
$(ms, k_1, k_2) \leftarrow kdf(g^{xy}, n_I | n_R)$
CFIN: Enc(k_1, mac(ms, $hash_2(log_1)$))
log_1
$(ms, k_1, k_2) \leftarrow kdf(g^{xy}, n_I | n_R)$
log_2
check mac.
log_2
check mac
SFIN: Enc(k_2, mac(ms, $hash_2(log_2)$))

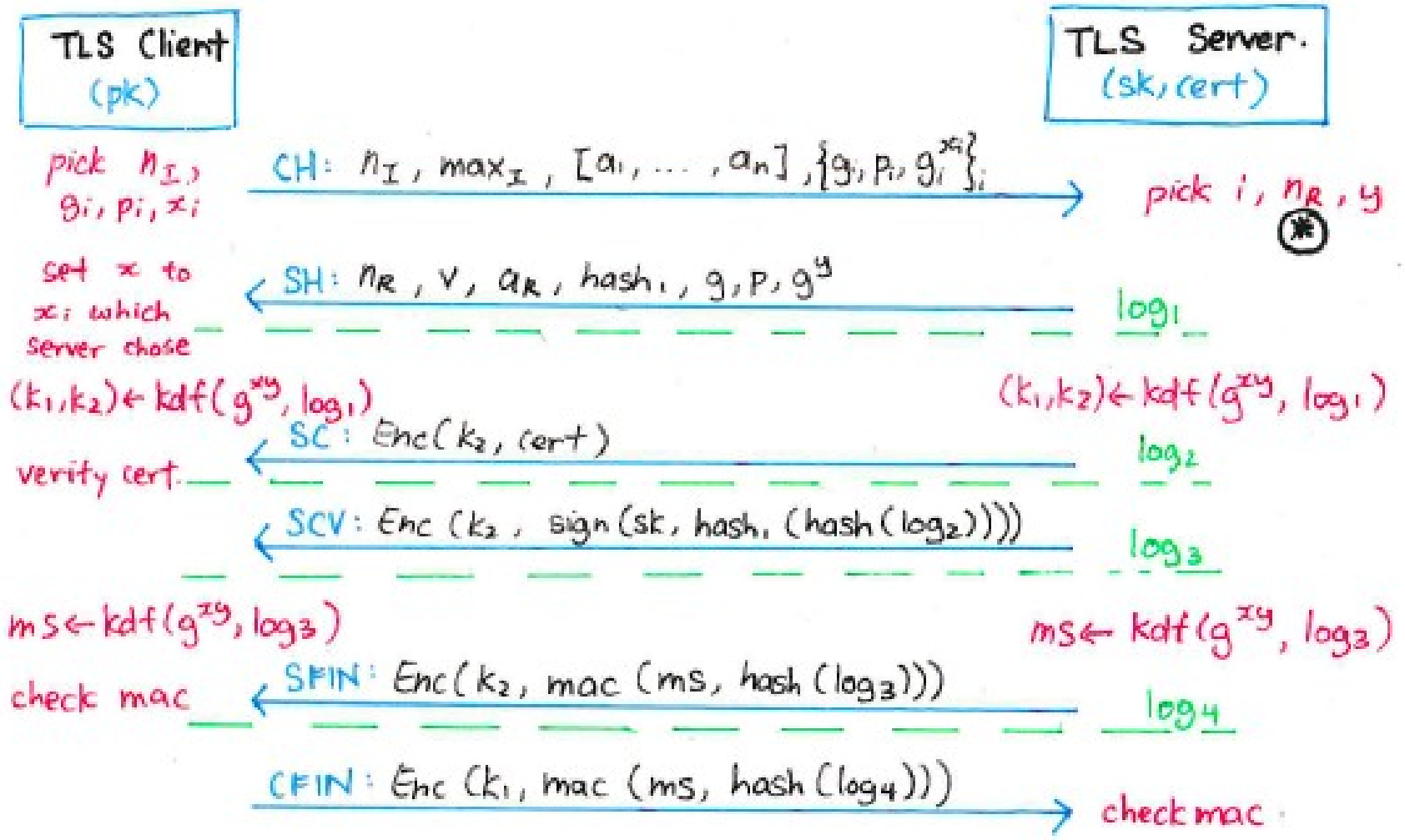
Does TLS 1.2 offer downgrade protection?

- Yes. But it is a strong requirement.
- First, we need that the signature scheme (sign) is secure
  - This is a reasonable assumption. Without it we have no way to verify the server.
- We need the final MAC to be unforgeable
- We also need that the adversary cannot retrieve *ms*, k_2 to compute the MAC himself
  - Which depends on kdf and $hash_2$
- Are these reasonable assumptions?
  - Logjam shows us that this is not. Attackers can use a weak group to retrieve *ms*, k_2

Logjam downgrade:
1.2 (DHE) → 1.2 (DHE-EXPORT)

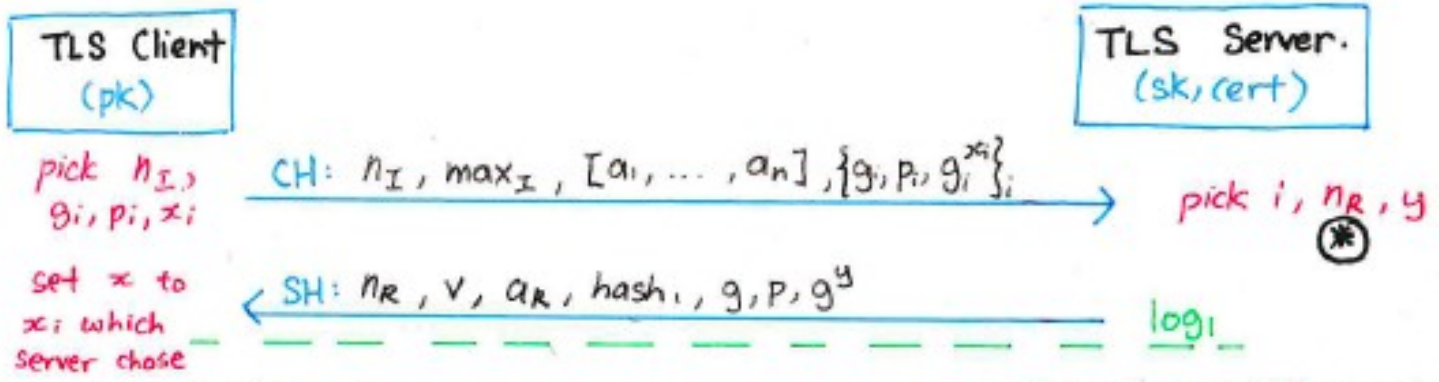


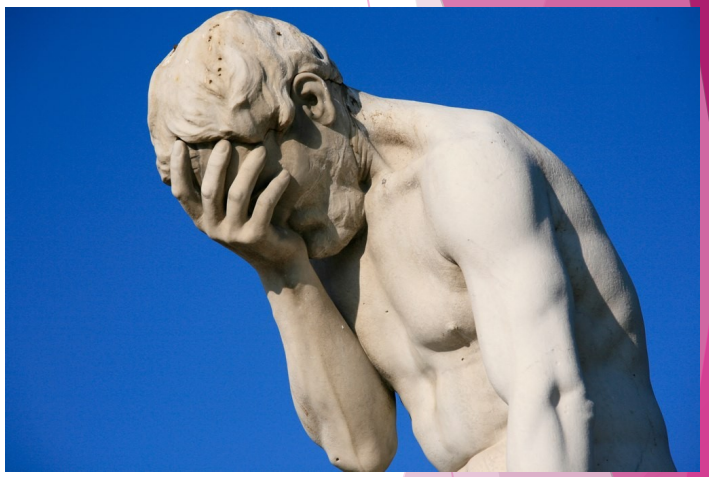
TLS 1.3 (draft 10)



Does the problem go away with TLS 1.3?

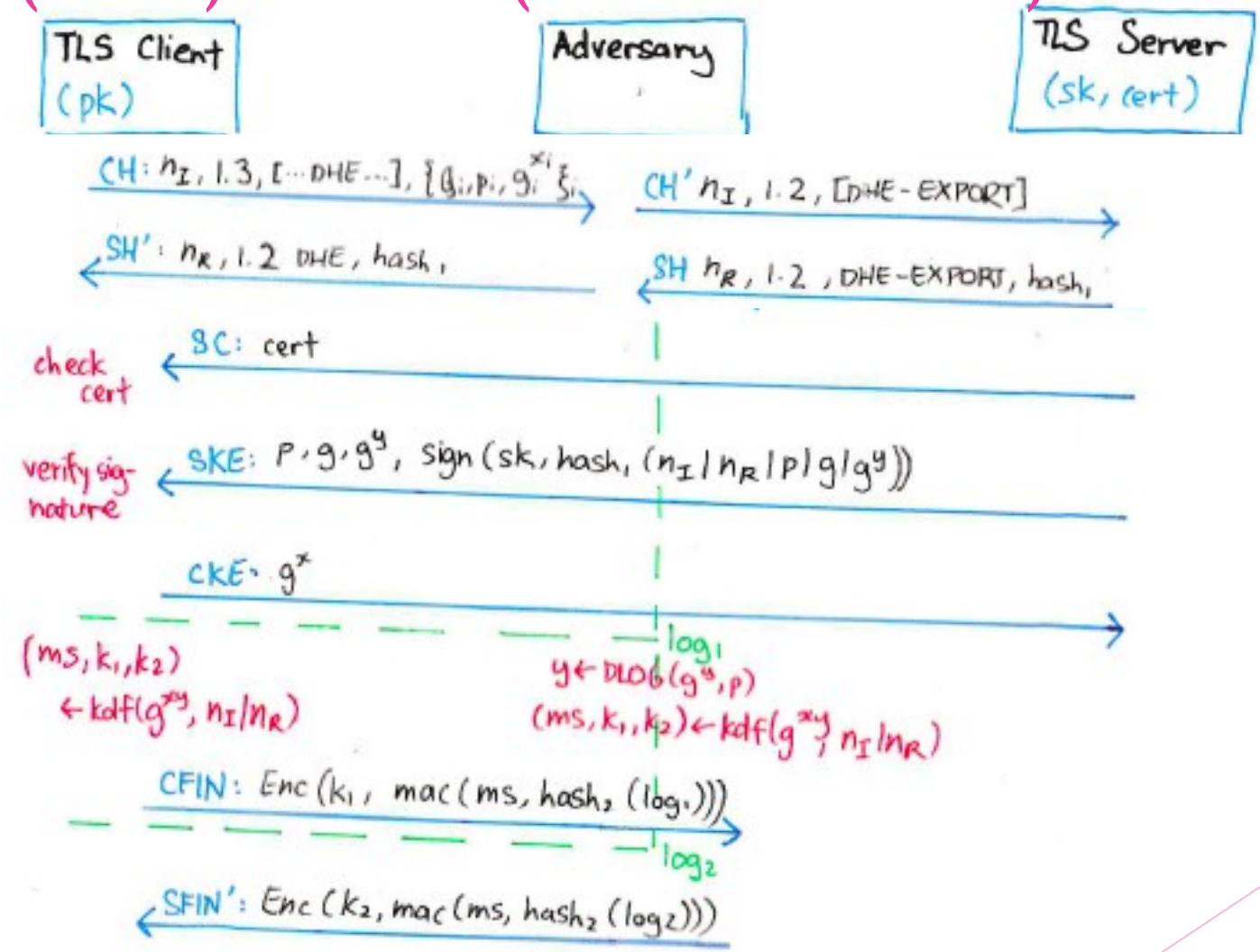
- No. Because an attacker can downgrade the connection to TLS 1.2





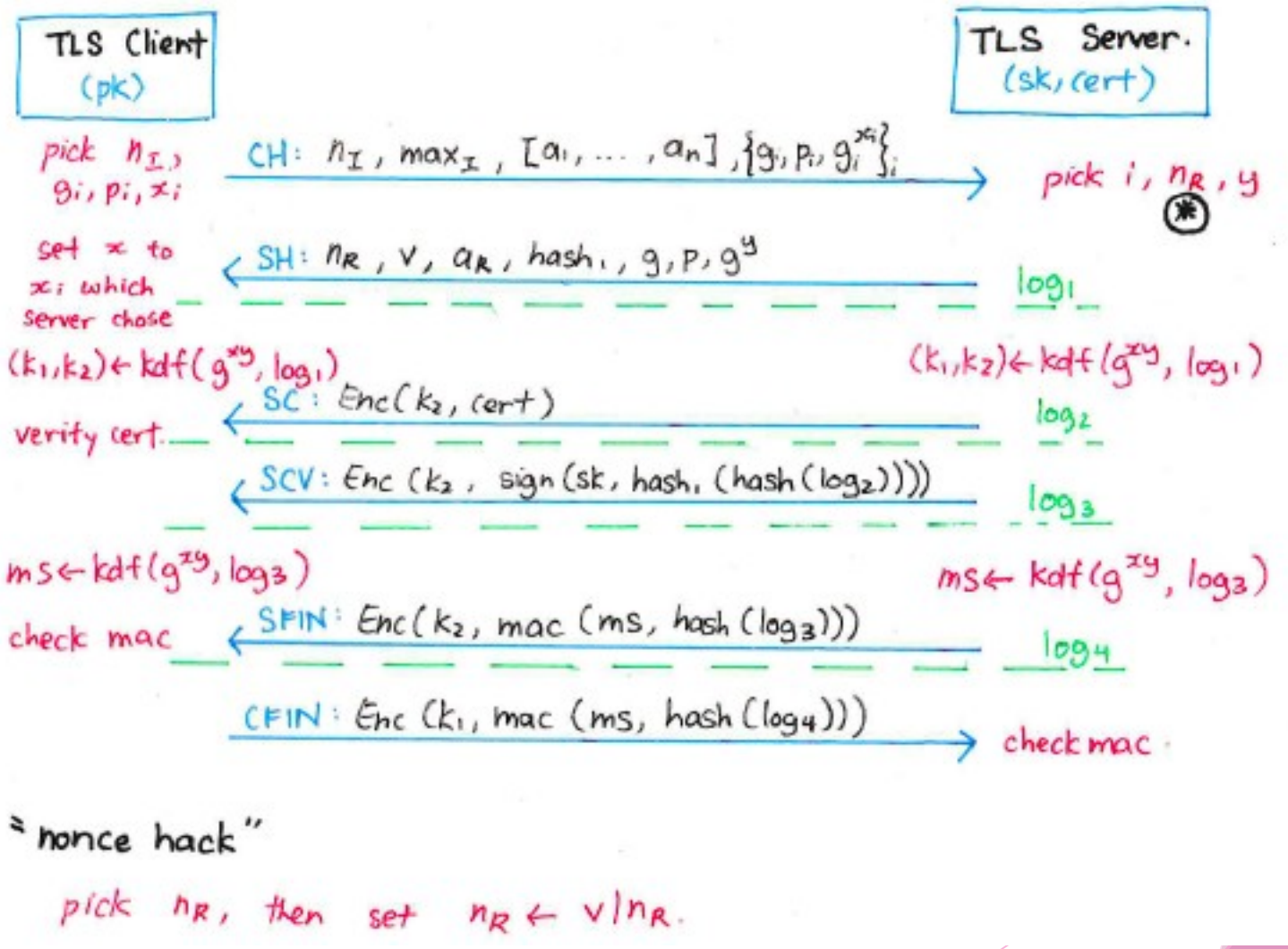
- A client needs to be able to talk to a server that only offers TLS 1.2, so if the server returns *v*=1.2, the client will proceed as in TLS 1.2
- This is a *version* downgrade
- So, unfortunately, TLS 1.3 (draft 10) is only secure against downgrades under the same assumptions we used for TLS 1.2

Logjam again:
1.3 (DHE) → 1.2 (DHE-EXPORT)



A simple “hack” to fix the problem

- Thankfully, there is a simple hack to fix the problem. We just include the version number in the server nonce.
- This means *v* will be signed by the server



Reflections from RSAC 2019

- To shift gears, I wanted to talk a little about my experiences from RSAC 2019
- RSA is a conference primarily aimed at practitioners.
  - Sessions discussing computer security in industry
  - Large expo to sell security products
  - Many “networking events”
- I attended two sessions which brought up points which I found very relevant to our discussions on the TLS standardization process.
  - “Hacked by Crypto” by Bret Jordan (Director, Symantec)
  - ”Assume possible interference” by Julie Tsai (InfoSec leader “passionate about bettering society through technology”)

“Hacked by Crypto” (Bret Jordan)

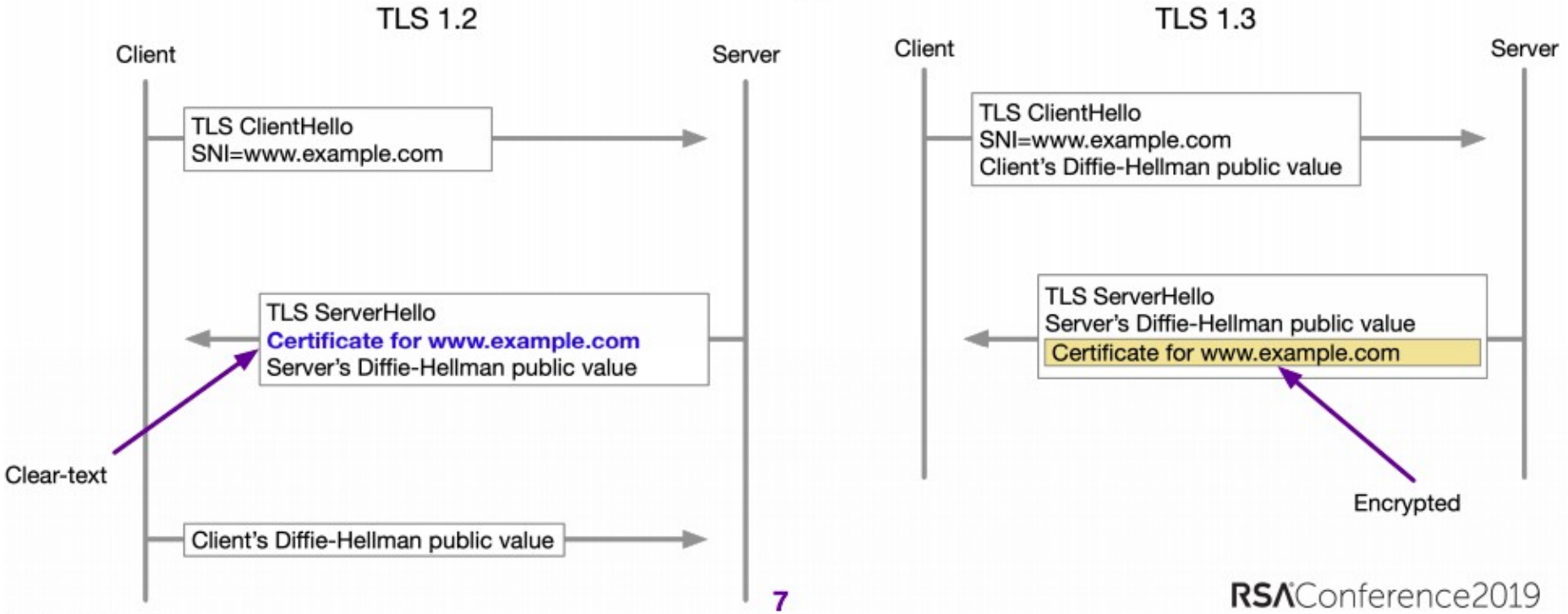
TLS 1.3 - RFC 8446 / Aug 2018

- Enhanced Security
  - Removed older broken crypto
  - Removed vulnerable TLS 1.2 configuration options
  - Restricted to Perfect Forward Secrecy (PFS) based Ciphers
- Reduced latency with improved performance and speed
  - TLS handshake only requires 1 round trip now instead of 2
  - Each roundtrip can add 100-300 ms, mobile networks add more

“Hacked by Crypto” (Bret Jordan)

TLS 1.3 - Potential Complication

- Server certificate is now encrypted



Graphic Source: Bret Jordan

“Hacked by Crypto” (Bret Jordan)

TLS 1.3 - Potential Complications

- Policy-based bypass is limited and less reliable
  - SNI can no longer be validated against server certificate
  - Server certificate can only be validated with full TLS termination
  - Harder to prevent phishing, stage 1&2 malware delivery, data exfiltration, and fraudulent transactions
  - Harder to prevent data leakage & industrial espionage
- URL categorization is limited to client provided SNI and IP addresses (however, SNI can be faked)
- No static RSA support for in-the-datacenter offline decrypt

“Hacked by Crypto” (Bret Jordan)

- Talked about users who need to basically be protected from themselves
- “I don’t care if software sees what my mom is typing into Google, I just really care that she doesn’t get phished!”
- Talked about his interactions with standards bodies who did not want to listen to his use-cases
- Claims that they respond with lots of rage in an attempt to “make him go away”
- Encourages practitioners to get involved in the standardization process and voice their use-cases.
  - “Make them know that you are the fly that won’t go away”

“Assume Possible Interference” (Julie Tsai)

- Not blindly using security technology, but ensuring that it aligns to your use case
- An example: “If your current policy requires you to inspect all data on the network except those coming from particular sites, and TLS 1.3 doesn’t allow you to discriminate via whitelisting anymore, maybe you should discuss the benefits of just looking at all the data”

Discussion

- Do we have an overly simplistic view of what is “correct” and what is “wrong”?
- How can we better communicate in the standardization process?