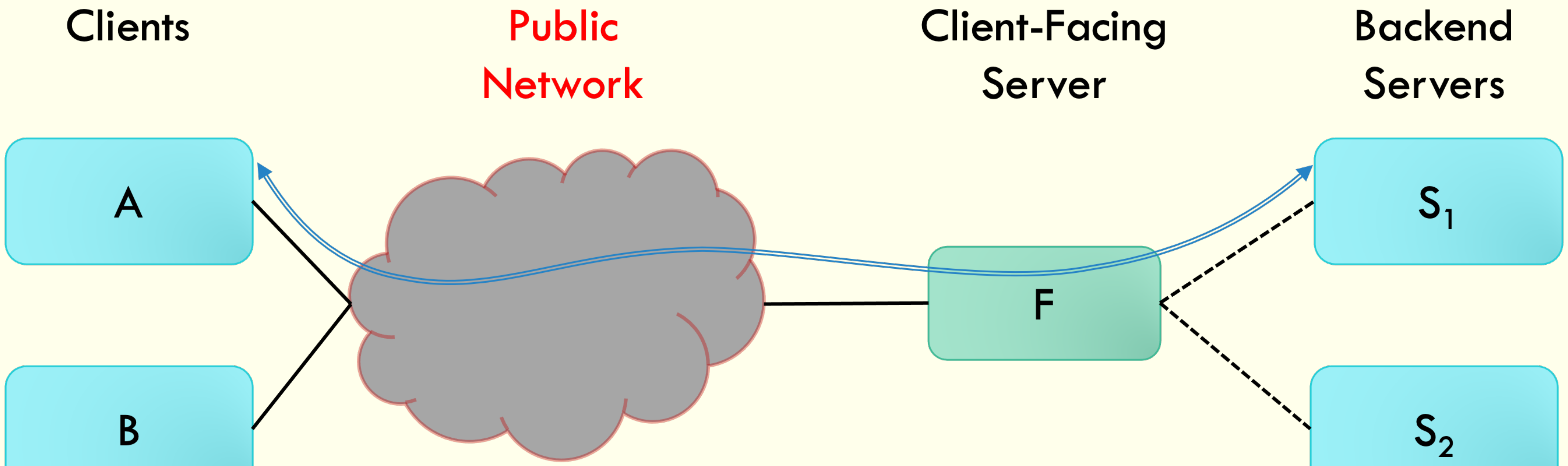
A Symbolic Analysis of Privacy for TLS 1.3 with Encrypted Client Hello

Karthikeyan Bhargavan[1]
Vincent Cheval[1]
Christopher Wood[2]

(1) INRIA Paris
Equipe Prosecco

(2) Cloudflare

TLS Deployment Scenario



The basic TLS 1.3 handshake

Diffie-Hellman key exchange

ClientHello
+Key Share
+Signature Algorithm
+Pre share key

→

ServerHello
+Key Share
+Pre share key

{EncryptedExtensions}
{CertificateRequest}
{Certificate}
{CertificateVerify}
{Finished}
[Application Data]

←

{Certificate}
{CertificateVerify}
{Finished}

→

[Application Data] ↔ [Application Data]

In green: Not always sent

{ X } : Encrypted with Handshake traffic key

[X] : Encrypted with Application traffic key

Several features

Negotiating Connection Parameters : HelloRetryRequest

Certificate-based Client Authentication

Pre-Shared Keys and Tickets

0RTT

Post Handshake Authentication

Other TLS extensions (e.g. SNI)

Verifying TLS requires to consider many scenarios

Security goals

Authentication and Integrity Goals

Server Authentication

Client Authentication

Key and Transcript Agreement

Data Stream Integrity

Key Uniqueness

Downgrade Resilience

Confidentiality

Key Secrecy

Key Indistinguishability

1RTT Data Forward Secrecy

0RTT Data Secrecy

Security goals

Authentication and Integrity Goals

Server Authentication (1,3,4)

Client Authentication (1,3,4)

Key and Transcript Agreement (1,3,4)

Data Stream Integrity (1,2,3,4)

Key Uniqueness (3,4)

Downgrade Resilience (4)

Confidentiality

Key Secrecy (1,2,3,4)

Key Indistinguishability (1)

1RTT Data Forward Secrecy (1,3,4)

0RTT Data Secrecy (1,2,3,4)

Automated verification to the rescue →

1. CryptoVerif
2. F*
3. Tamarin
4. ProVerif

Security goals

Authentication and Integrity Goals

Server Authentication (1,3,4)

Client Authentication (1,3,4)

Key and Transcript Agreement (1,3,4)

Data Stream Integrity (1,2,3,4)

Key Uniqueness (3,4)

Downgrade Resilience (4)

Confidentiality

Key Secrecy (1,2,3,4)

Key Indistinguishability (1)

1RTT Data Forward Secrecy (1,3,4)

0RTT Data Secrecy (1,2,3,4)

Automated verification to the rescue →

1. CryptoVerif
2. F*
3. Tamarin
4. ProVerif

These models do not cover all features

Security goals

Privacy

Client Identity Privacy

Client Unlinkability

Server Extension Privacy

Client Extension Privacy

Server Identity Privacy

Security goals

Privacy

Client Identity Privacy	No automated proofs
Client Unlinkability	
Server Extension Privacy	
Client Extension Privacy	Extension in ClientHello
Server Identity Privacy	SNI in ClientHello

Security goals

Privacy

Client Identity Privacy

Client Unlinkability

Server Extension Privacy

Client Extension Privacy

Server Identity Privacy

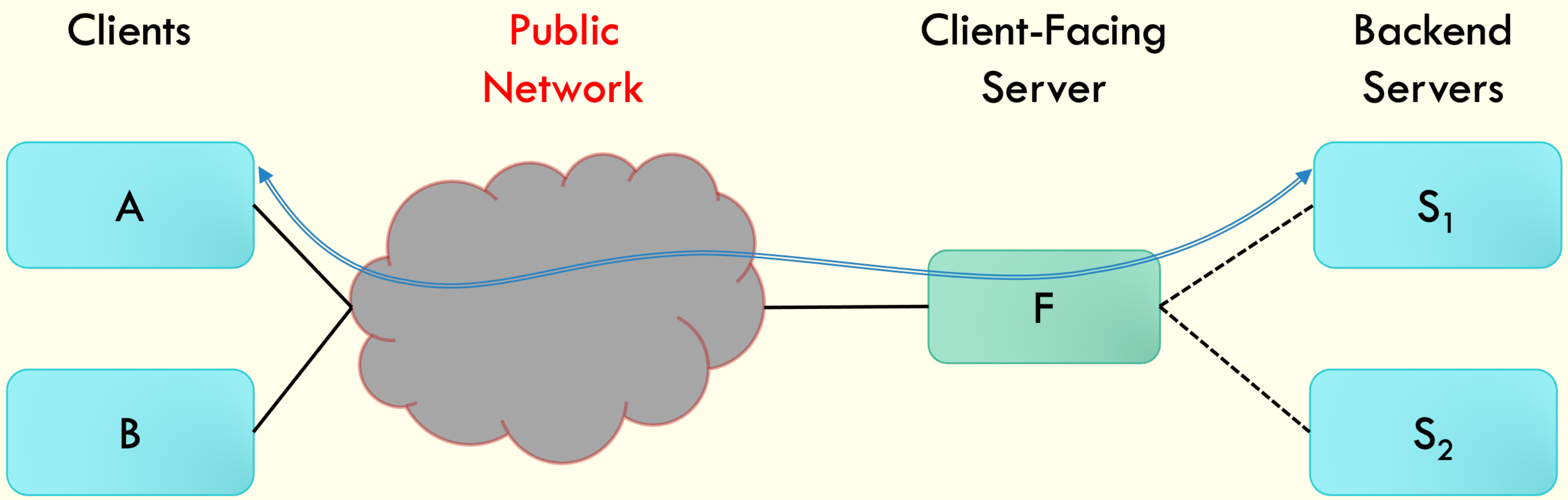
No automated proofs

Extension in ClientHello

SNI in ClientHello

Encrypted Client Hello guarantees
all these privacy goals

ECH

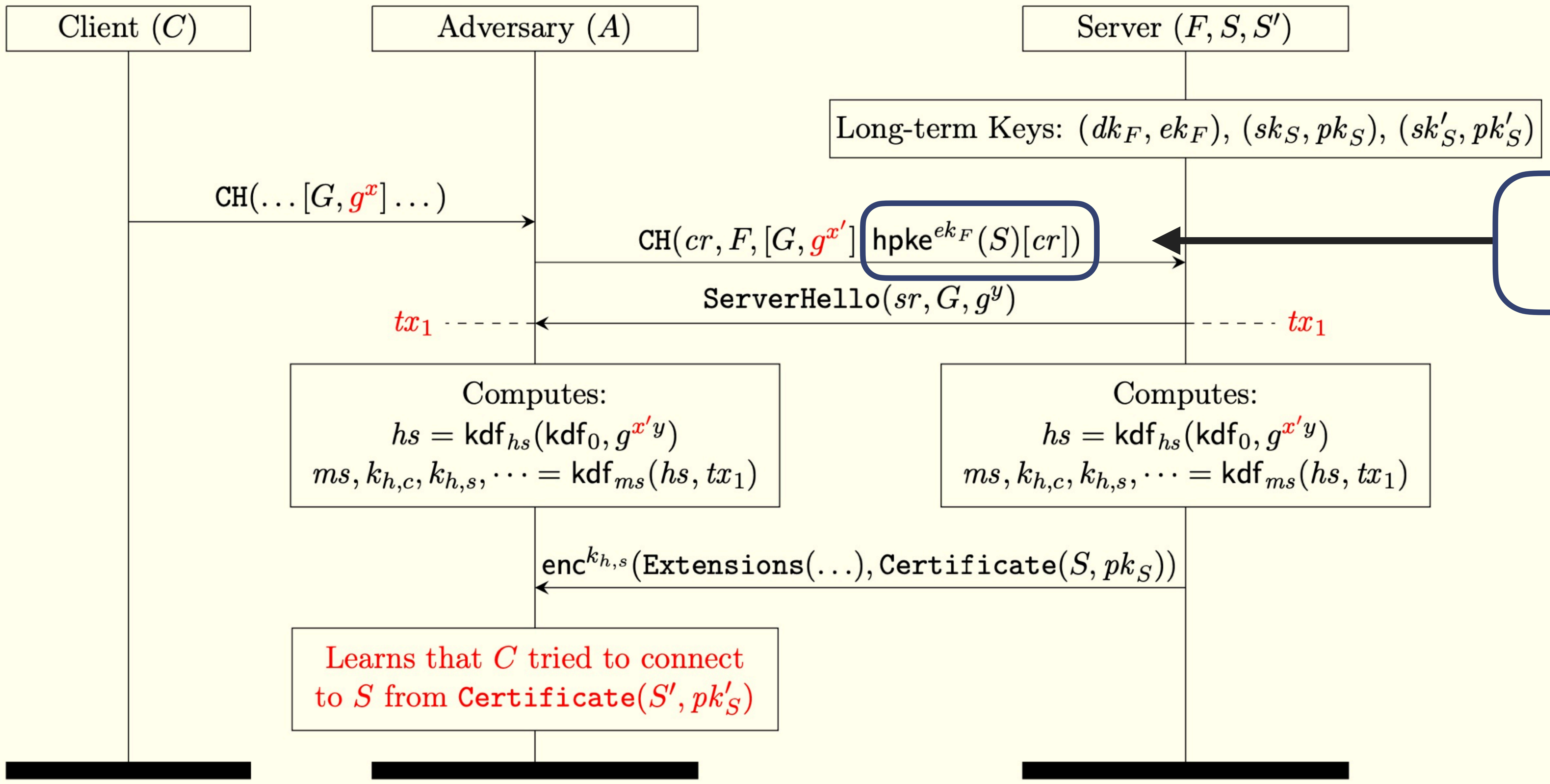


Goal: Privacy of the identity of the backend server

Main idea: Encrypt sensitive informations (e.g. server identity of the backend server) with a public key of the client-facing server

Not so easy: Several previous designs were vulnerable

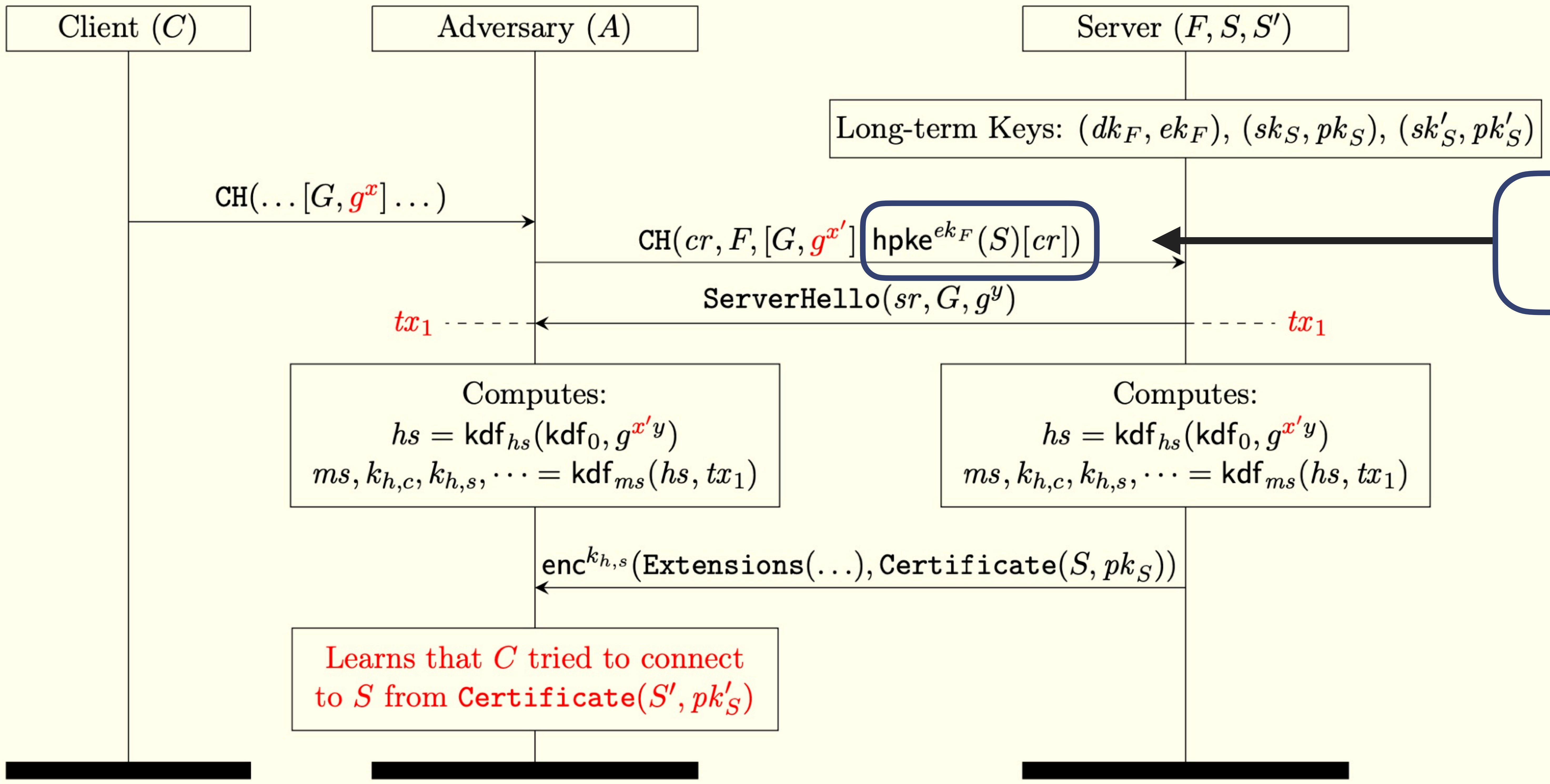
First draft: Encrypt the SNI and ClientHello.random



Hybrid Public Key Encryption

Not so easy: Several previous designs were vulnerable

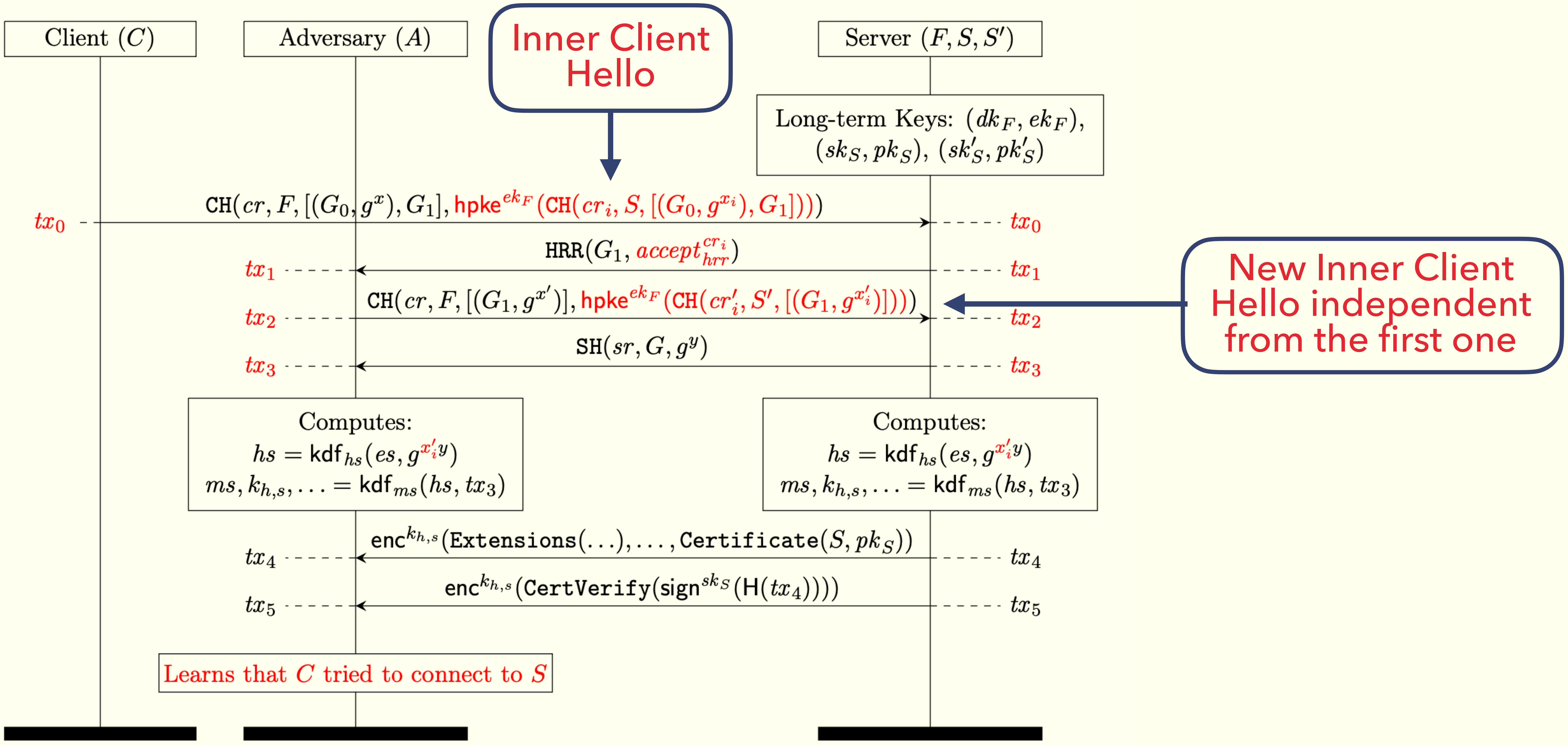
First draft: Encrypt the SNI and ClientHello.random



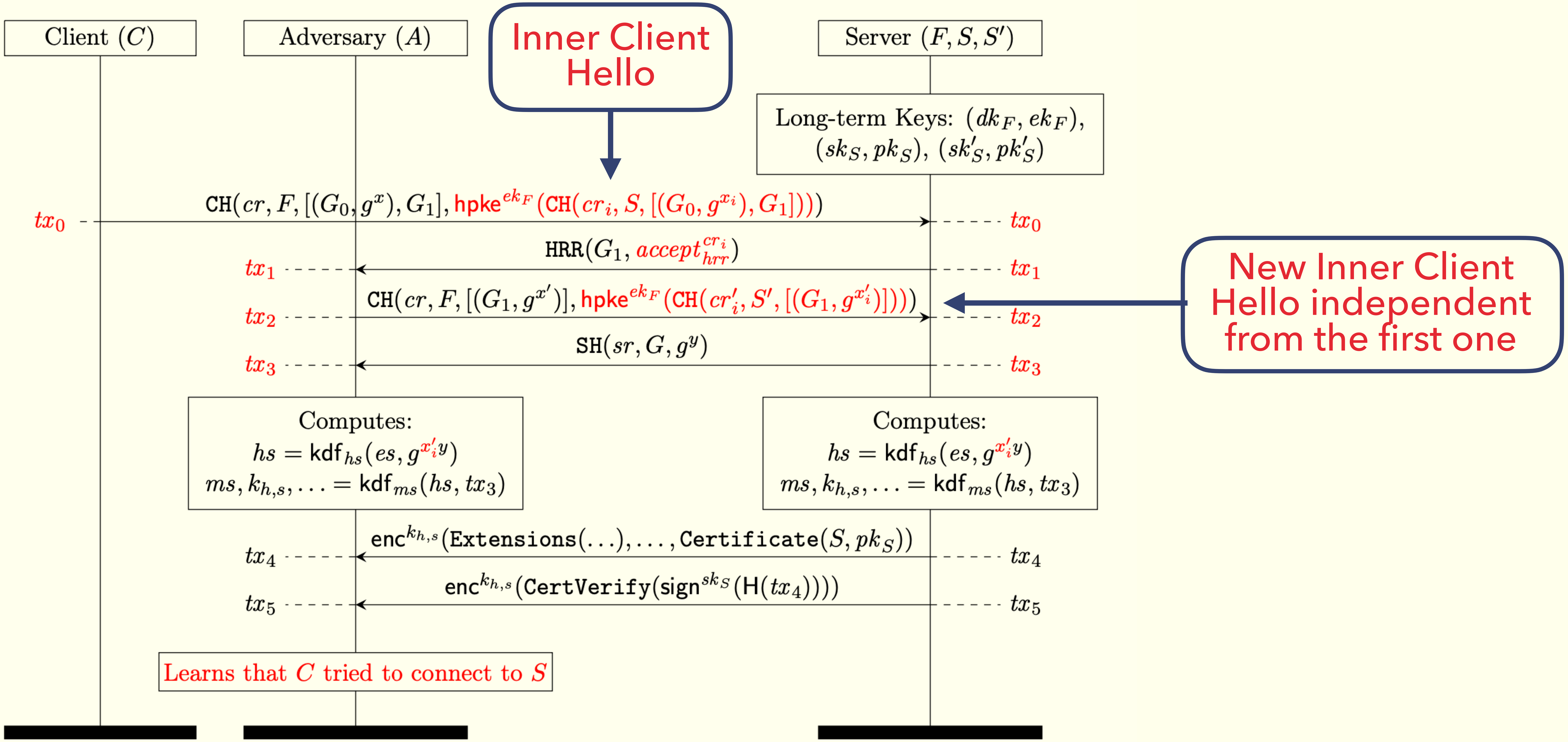
Hybrid Public Key Encryption

Main idea: Encrypt the whole Client Hello destined for the backend server (inner) and bind it with the Client Hello for the Client-Facing server (outer)

Not so easy: HelloRetryRequest



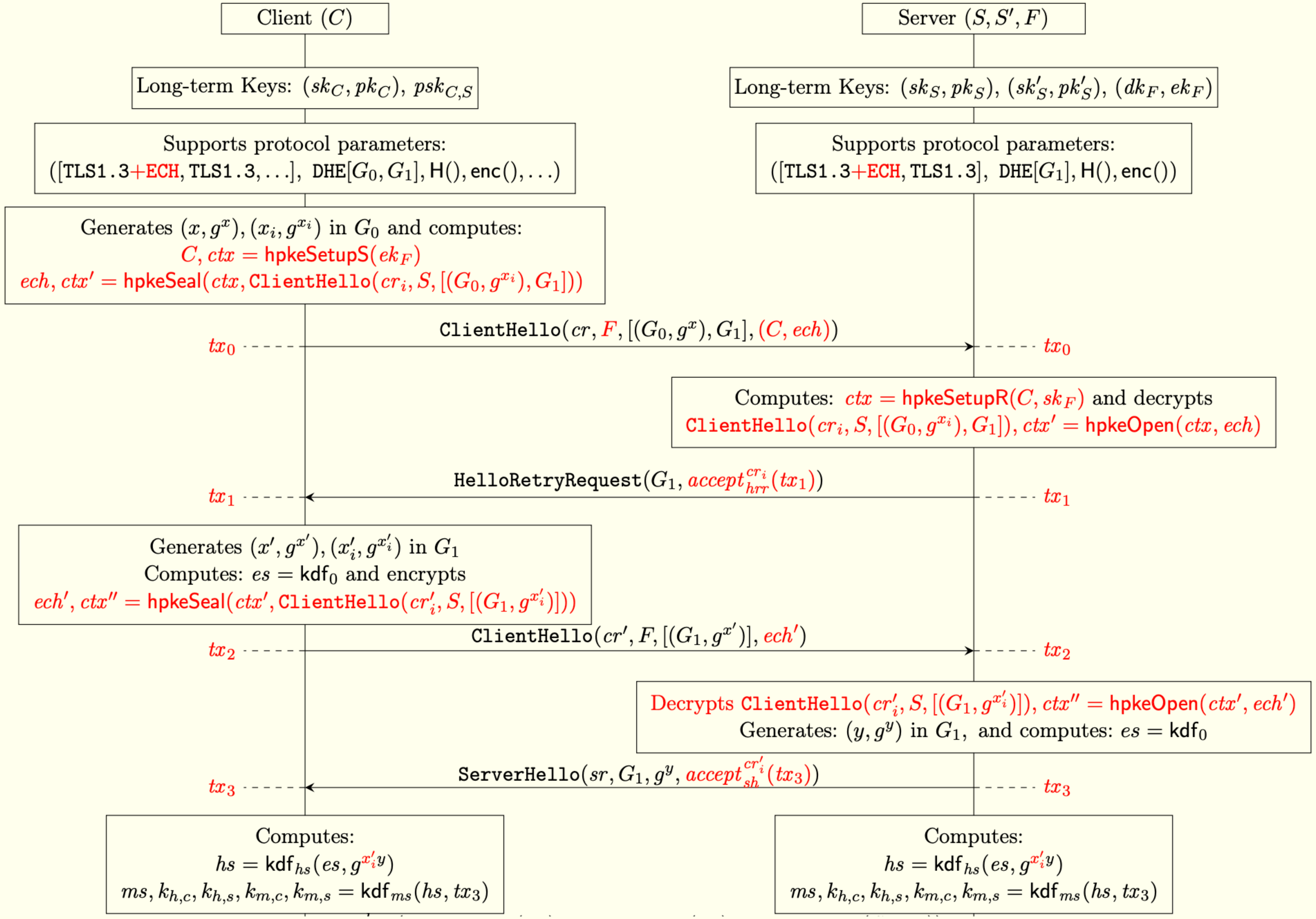
Not so easy: HelloRetryRequest



The encryption of the second Inner Client Hello must be linked to the first Inner Client Hello

Encrypted Client Hello (ECH)

The context ctx is updated after each encryption (ctx', ctx'')



The context ctx is updated after each decryption (ctx', ctx'')

Attacker model

The attacker can…

Read / Write

Intercept

But they do not…

Break cryptograhy

Use side channels

Dolev-Yao models

Concurrent systems where dishonest parties have complete control over network communication ***but*** cryptography is idealised

Automated Verification Tool : ProVerif

Our model

Focus only on TLS 1.3 (no version negociation)

Model all features presented before (e.g. HRR, PHA, PSK, Ticket, ECH, 1RTT and 0RTT Data)

Model all security properties presented before (i.e. Authentication, Integrity, Confidentiality and Privacy goals)

Proving all properties with all features is too taxing on ProVerif in computation time or memory consumption

OOT = 48H and OOM = 100GB

Our model

Focus only on TLS 1.3 (no version negociation)

Model all features presented before (e.g. HRR, PHA, PSK, Ticket, ECH, 1RTT and 0RTT Data)

Model all security properties presented before (i.e. Authentication, Integrity, Confidentiality and Privacy goals)

```
(*************************)
(*    Functionnalities   *)
(*************************)

(* When `false`, an honest client will always send its key share with the group.
  Moreover, an honest server will never send a HRR request.*)
letfun allow_HRR = false.

(* When `false`, honest clients and servers are not expecting and sending a new
  session ticket respectively. *)
letfun allow_PH_new_session_ticket = true.

(* When `false`, honest clients and servers will never send or try to receive
  Post Handshake Application Data. *)
letfun allow_PH_data = false.

(* When `false`, honest servers will request post handshake authentication
  and honest clients will never wait for one. *)
letfun allow_PH_authentication = false.

(* When `false`, honest clients and servers will never send or try to receive
  early data. *)
letfun allow_early_data = false.

(************************************************)
(*    Safety of Keys, Cipher suite and group    *)
(************************************************)

(* When `true`, private keys of Ech configuration can be compromised. *)
letfun allow_compromised_Ech_keys = false.
```

Proving all properties with all features is too taxing on ProVerif in computation time or memory consumption
OOT = 48H and OOM = 100GB

Parametrized model

Simple configuration file allows us to activate/deactivate:

- Features
- Compromised keys
- Server and client behavior

621 runs of ProVerif

Our results (Authentication, Integrity, Confidentiality)

✓ : Feature enabled ✗ : Feature disabled

Sanity Checks

Security preservation

Computation time

	Property	1-RTT	HRR	CC	PHA	PSK-DHE	TKT	0-RTT	Time
TLS	All	✓	✓	✓	✓	✓	✓	✓	10h7m
TLS + ECH	SEC, UNIQ	✓	✓	✓	✓	✓	✓	✗	2h48m
	SEC0	✓	✓	✗	✓	✓	✓	✓	55m
	FS, INT	✓	✗	✓	✗	✓	✓	✗	3h40m
	CAUTH	✓	✗	✗	✓	✓	✓	✓	2h39m
		✓	✓	✓	✗	✓	✓	✗	3h26m
	SAUTH, AGR	✓	✓	✓	✗	✓	✓	✗	3h26m
	DOWN	✓	✓	✗	✗	✓	✓	✗	34h16m

Downgrade resilience w.r.t. ECH

Assumptions for Privacy of Server Identity

Equivalence between two scenarios

$$H_e(c_1, fs_1, bs_1) \mid \ldots \mid H_e(c_n, fs_n, p_n) \mid \; H_e(A, fs^*, BS_1) \mid \ldots$$

and

$$H(c_1, fs_1, bs_1) \mid \ldots \mid H_e(c_n, fs_n, p_n) \mid \; H_e(A, fs^*, BS_2) \mid \ldots$$

1. HPKE private key of Client-facing server fs^* is uncompromised

   If not: The can directly decrypt the ECH extension to obtain the identity of the backend server

2. BS_1 and BS_2 both have a certificate long term key or none of them have one.

   If not : The basic handshake where the server must send its certificate will only succeed in one of the scenarios

3. A share a (different,uncompromised) PSK with both BS_1 and BS_2 or with neither of them.

   If not : The number of messages sent will differ

Our results (Privacy)

For Privacy properties, 1RTT and 0RTT are disabled

✓ : Feature enabled ✗ : Feature disabled

	Property	HRR	CC	PHA	PSK-DHE	TKT	Time
TLS	IND, CIP UNL, S-EXT	✓	✓	✗	✓	✓	17H15
	CIP,UNL	✓	✓	✓	✓	✗	10h10m
ECH	IND	✗	✓	✗	✓	✓	21h16m
		✓	✗	✗	✗	✓	12h47
	SIP	✗	✗	✗	✓	✓	24h27m
		✓	✗	✗	✗	✗	1h13m
	CIP, UNL	✗	✓	✗	✓	✗	21h42m
		✗	✗	✗	✓	✓	35h22m
		✗	✓	✓	✗	✗	3h27m
	S-EXT,C-EXT	✓	✓	✗	✓	✗	21h20m

Privacy properties requires more time and memory

Ongoing work: Improve ProVerif to reduce memory consumption

Thank you !

Questions ?