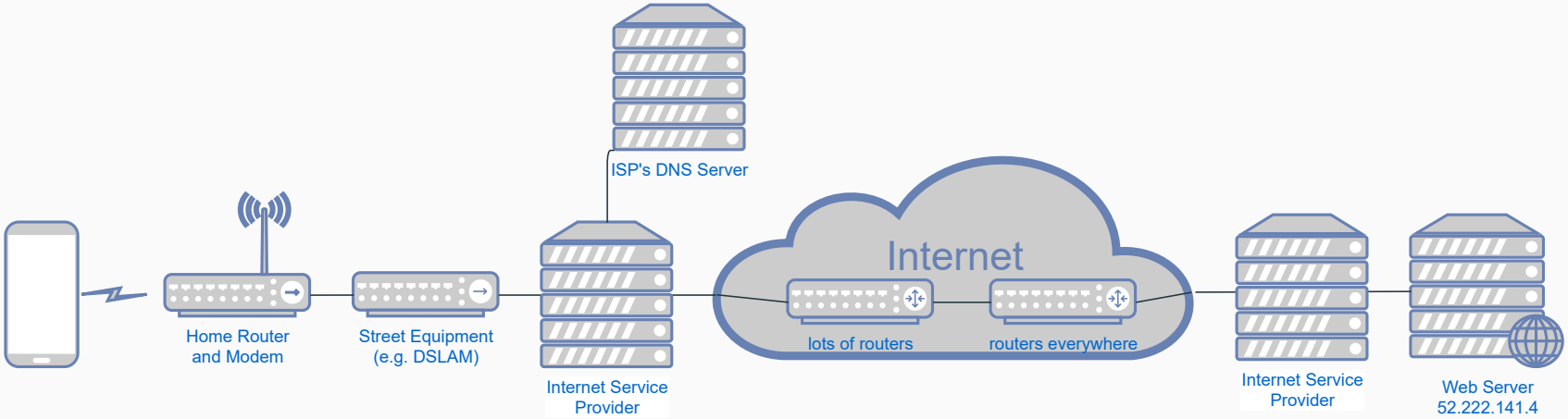
A New Internet Standard for Hybrid Public Key Encryption

Benjamin Lipp

January 19, 2021

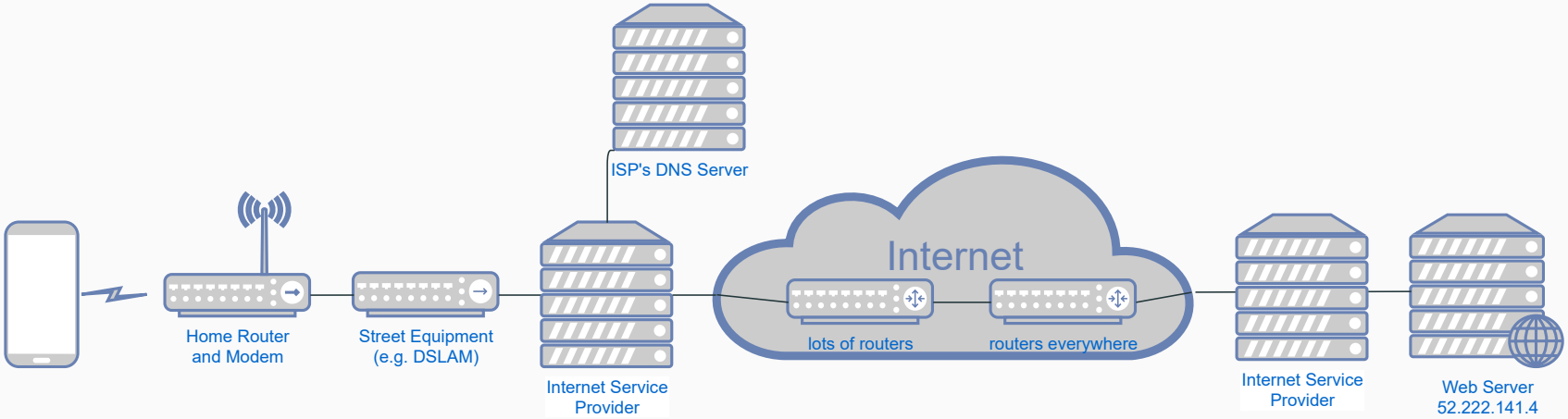
Inria Paris, Prosecco

Context: The Internet



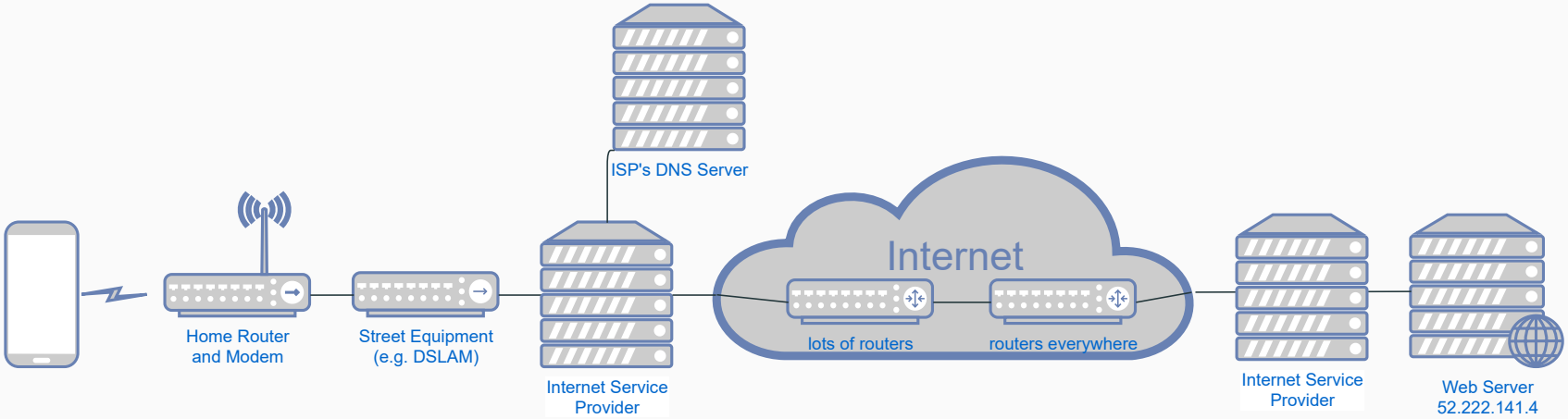
- “the global system of interconnected computer systems using the Internet protocol suite.” (Wikipedia)

Context: The Internet



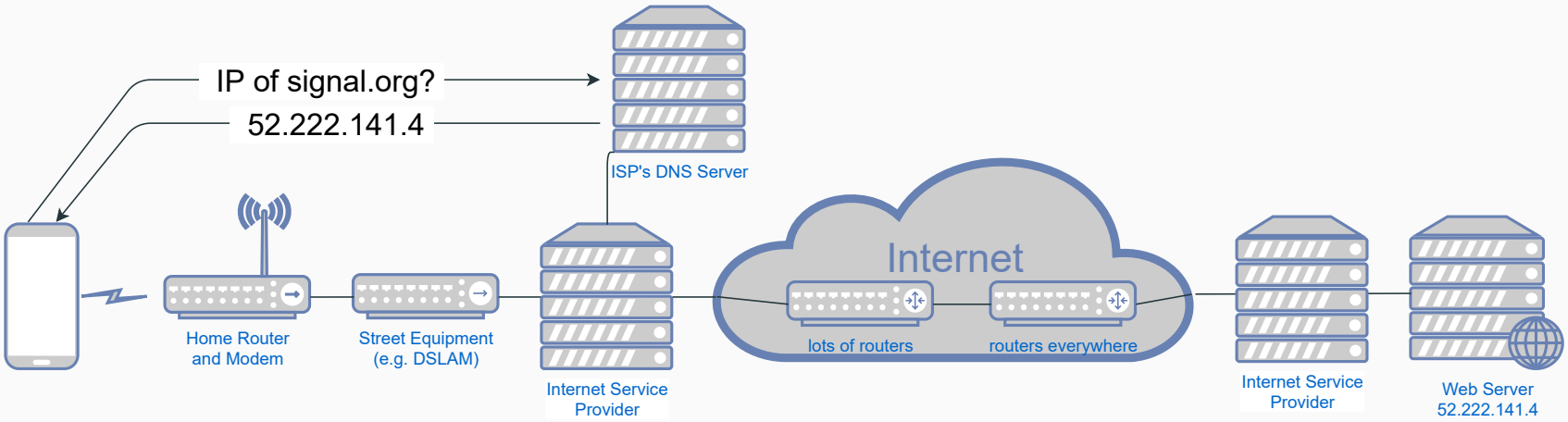
- “the global system of interconnected computer systems using the Internet protocol suite.” (Wikipedia)
- computers address each other via an IP address

Context: The Internet



- "the global system of interconnected computer systems using the Internet protocol suite." (Wikipedia)
- computers address each other via an IP address
- domain names abstract over IP addresses via the Internet's addressbook, the Domain Name System (DNS)

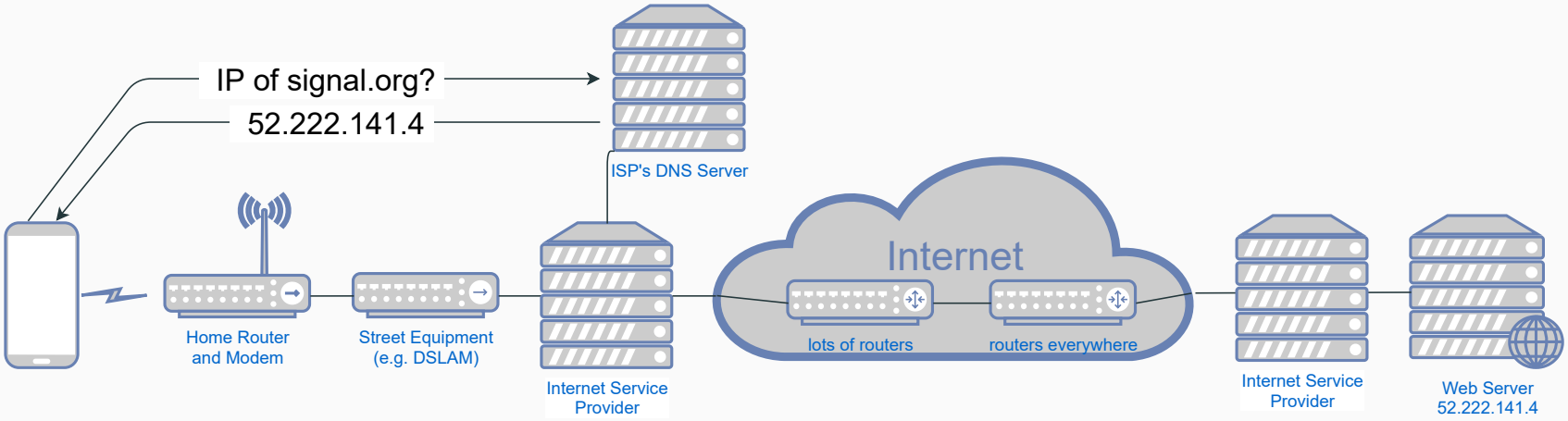
Context: The Internet



- "the global system of interconnected computer systems using the Internet protocol suite." (Wikipedia)
- computers address each other via an IP address
- domain names abstract over IP addresses via the Internet's addressbook, the Domain Name System (DNS)

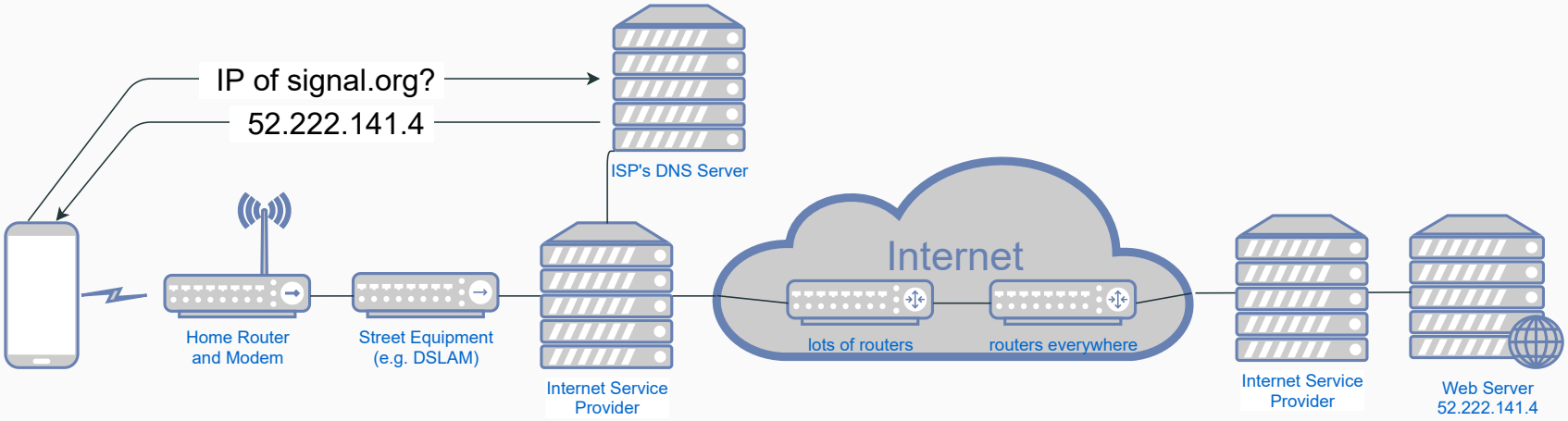
Accessing Services Securely via Transport Layer Security (TLS)

- every equipment on the path could tamper with the packets



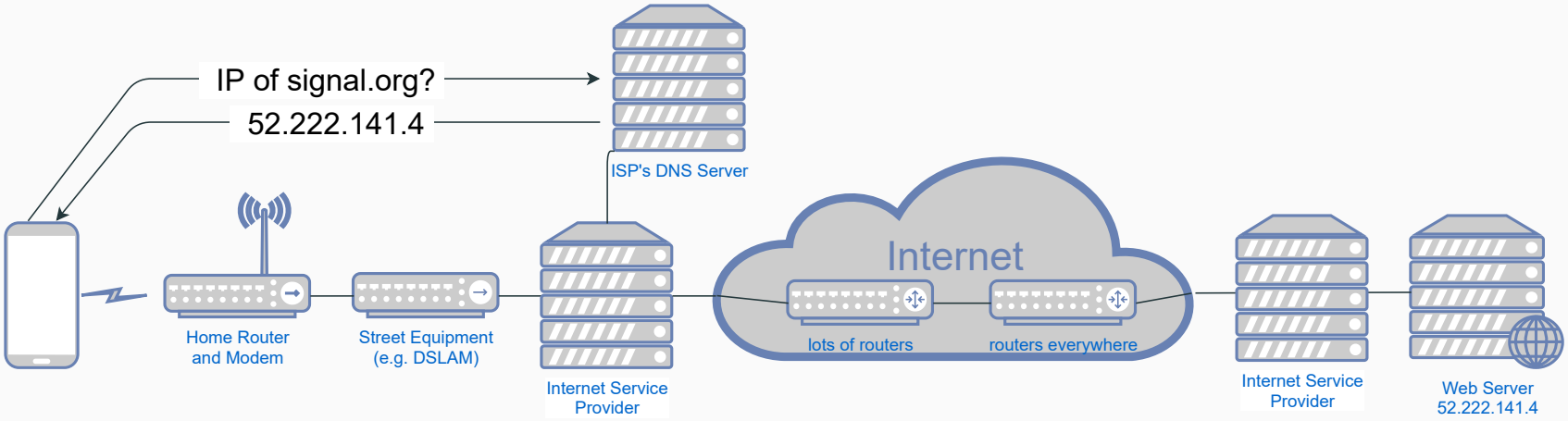
Accessing Services Securely via Transport Layer Security (TLS)

- every equipment on the path could tamper with the packets
- encryption protects against reading contents

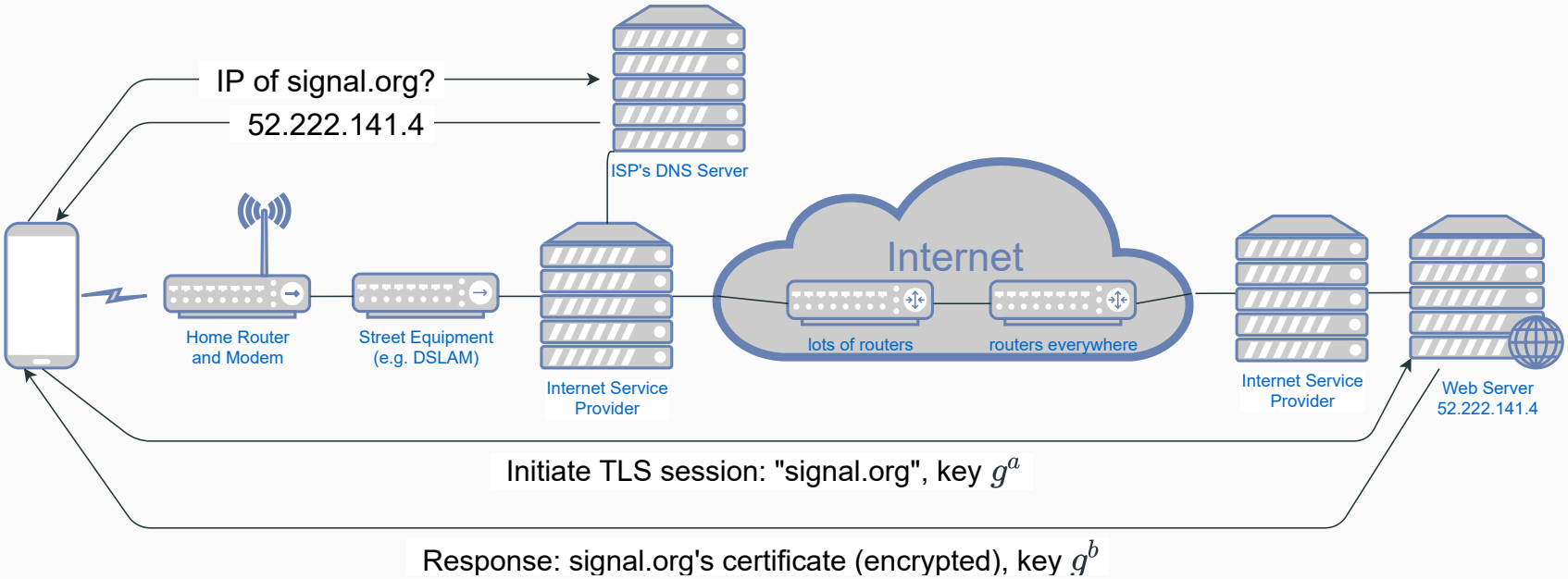


Accessing Services Securely via Transport Layer Security (TLS)

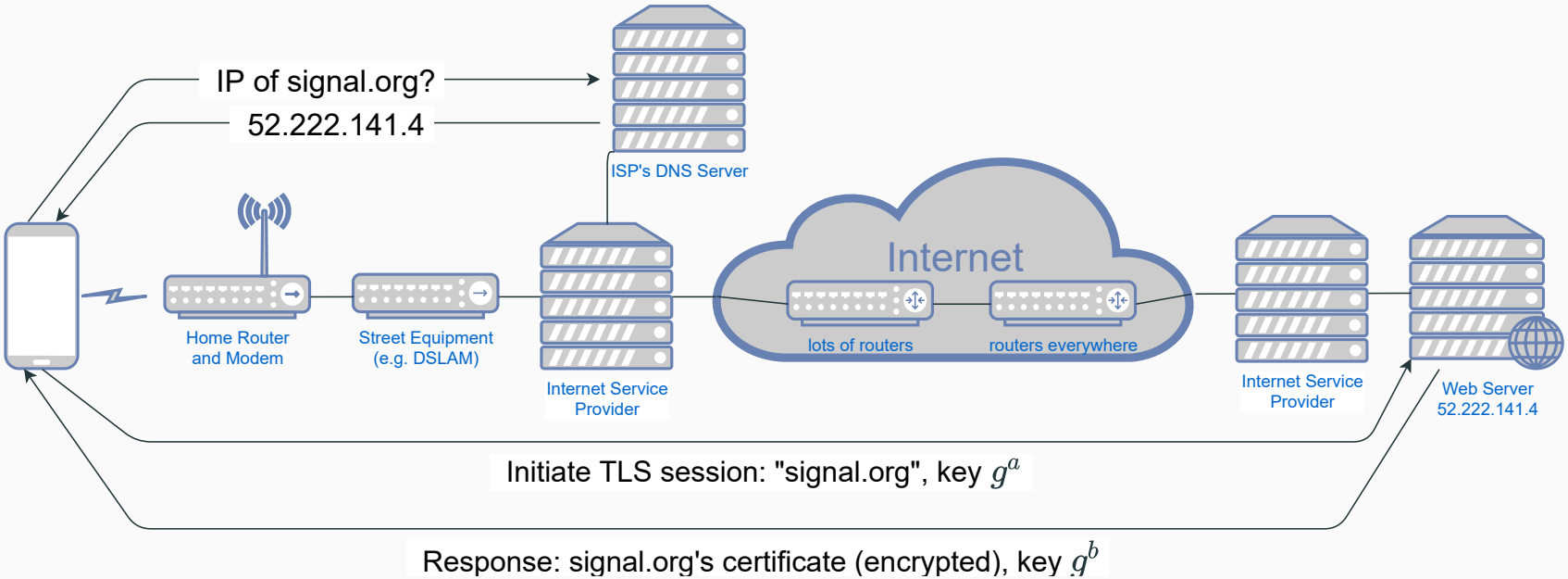
- every equipment on the path could tamper with the packets
- encryption protects against reading contents
- authentication protects against changes of contents and impersonation



Accessing Services Securely via Transport Layer Security (TLS)



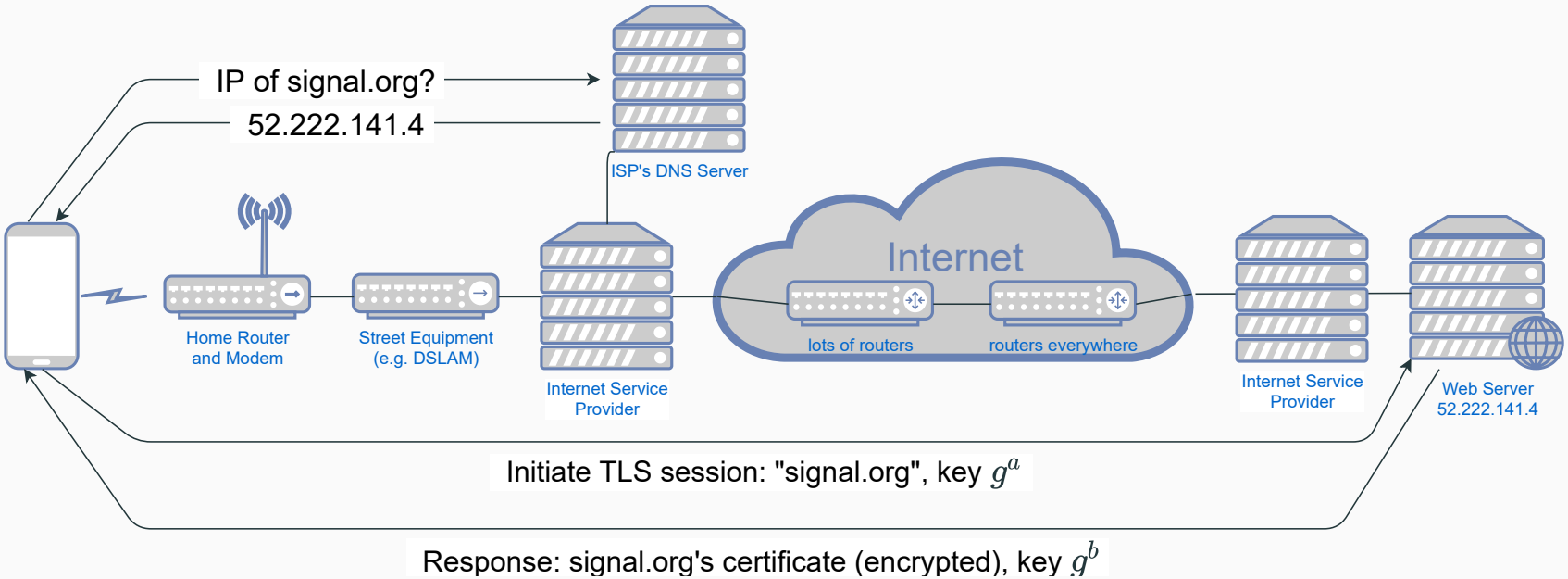
Accessing Services Securely via Transport Layer Security (TLS)



- key exchange protocol: compute $(g^a)^b = (g^b)^a = g^{ab}$ and derive a session key from that for subsequent data.

  This is secure by a Diffie-Hellman cryptographic assumption.

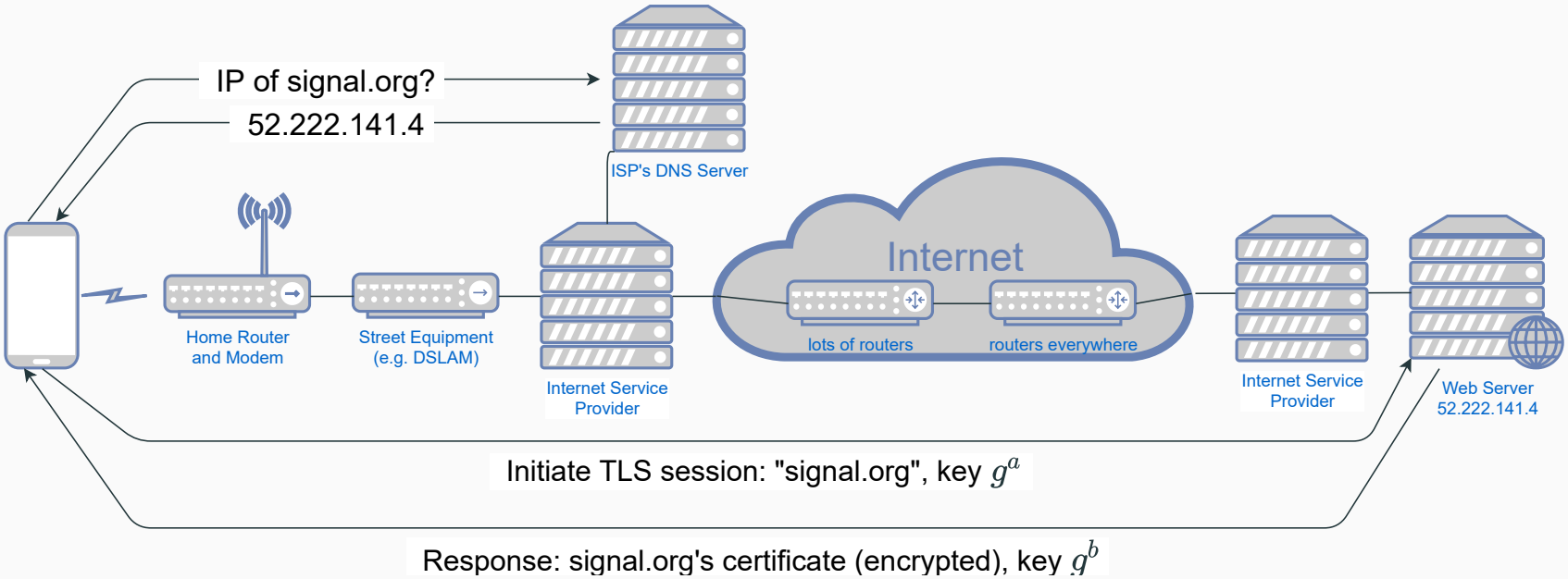
Accessing Services Securely via Transport Layer Security (TLS)



- key exchange protocol: compute $(g^a)^b = (g^b)^a = g^{ab}$ and derive a session key from that for subsequent data.

  This is secure by a Diffie-Hellman cryptographic assumption.
- The server proves affiliation to a domain by a certificate that has been signed by a certificate authority

Accessing Services Securely via Transport Layer Security (TLS)



- key exchange protocol: compute $(g^a)^b = (g^b)^a = g^{ab}$ and derive a session key from that for subsequent data.

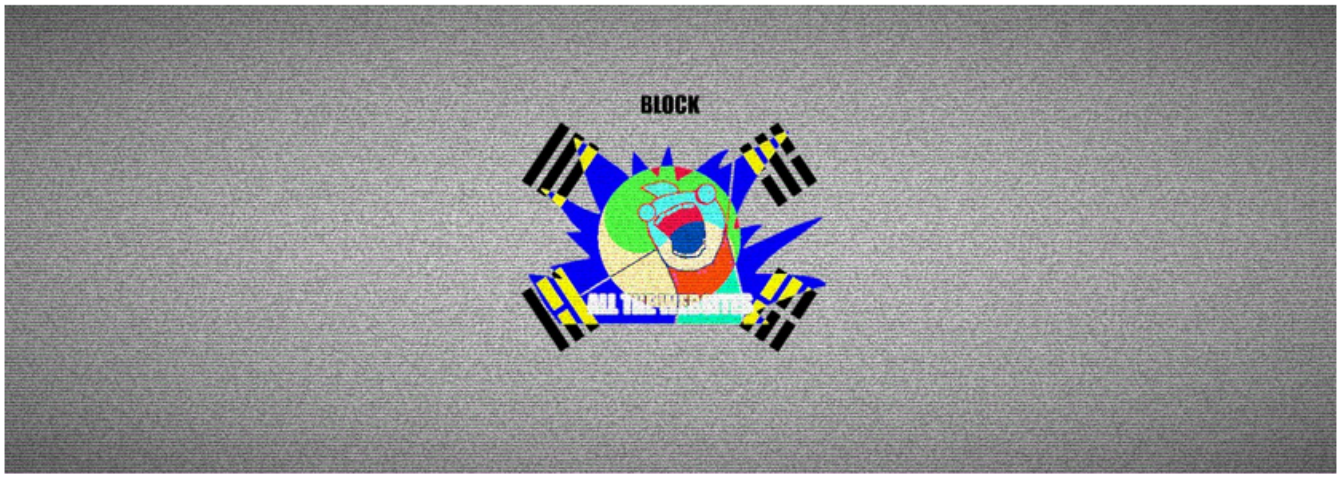
  This is secure by a Diffie-Hellman cryptographic assumption.
- The server proves affiliation to a domain by a certificate that has been signed by a certificate authority
- The server will know that all successfully decrypted data is from the same client that started the connection.

The Server Name Indication has been Plain Text ...

... and thus TLS is useless to hide the destination of traffic.

South Korea is Censoring the Internet by Snooping on SNI Traffic

By Sergiu Gatlan — February 13, 2019 — 06:19 PM — 1



South Korea has been blocking HTTP websites that are on their censor list for a while now and they have recently started using SNI filtering to block their counterparts served over HTTPS.

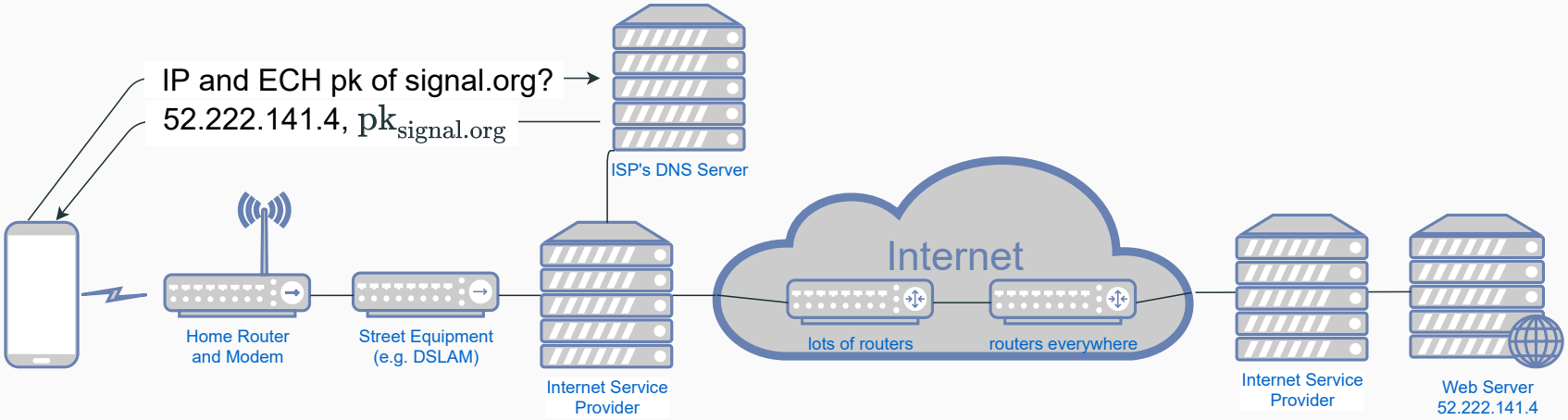
A warning page bearing the seals of the Korea Communications Standards Commission (KCSC) and the Korean National Police Agency is displayed for blocked HTTP websites, while TLS sites blocked using Server Name Indication (SNI) filtering will only throw a "This site can't be reached" error.

https://www.bleepingcomputer.com/news/security/south-korea-is-censoring-the-internet-by-snooping-on-sni-traffic/

Hide the Payload of Server Name Indication

Since 2018, experiments have been conducted to encrypt the Server Name Indication

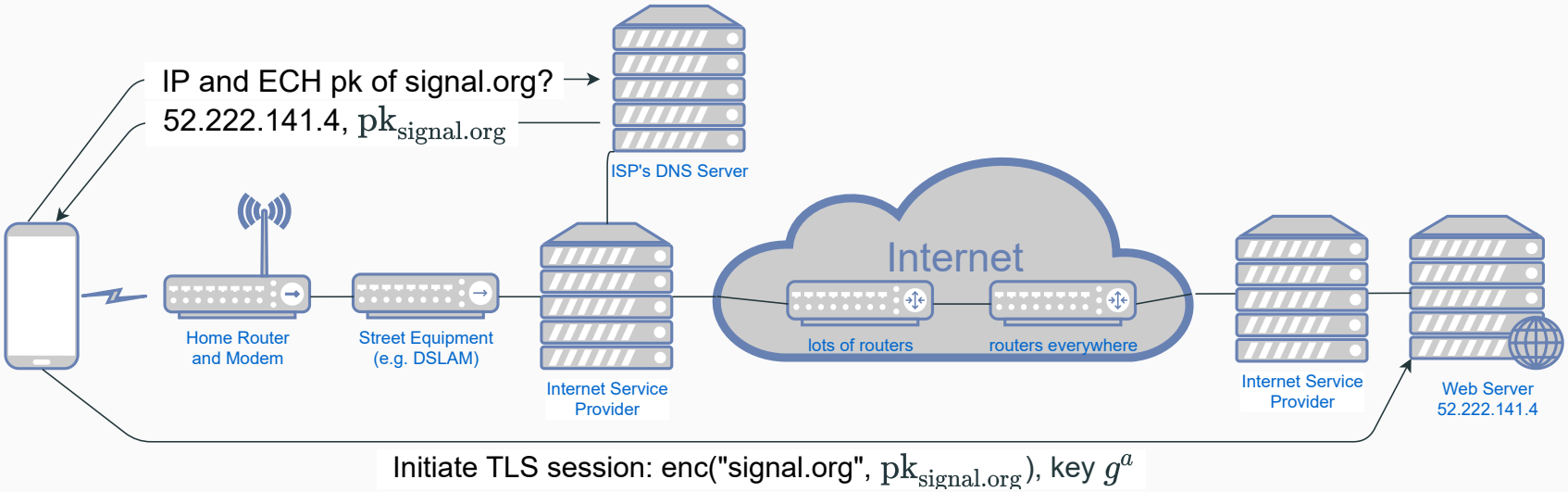
`https://blog.cloudflare.com/esni/`



Hide the Payload of Server Name Indication

Since 2018, experiments have been conducted to encrypt the Server Name Indication

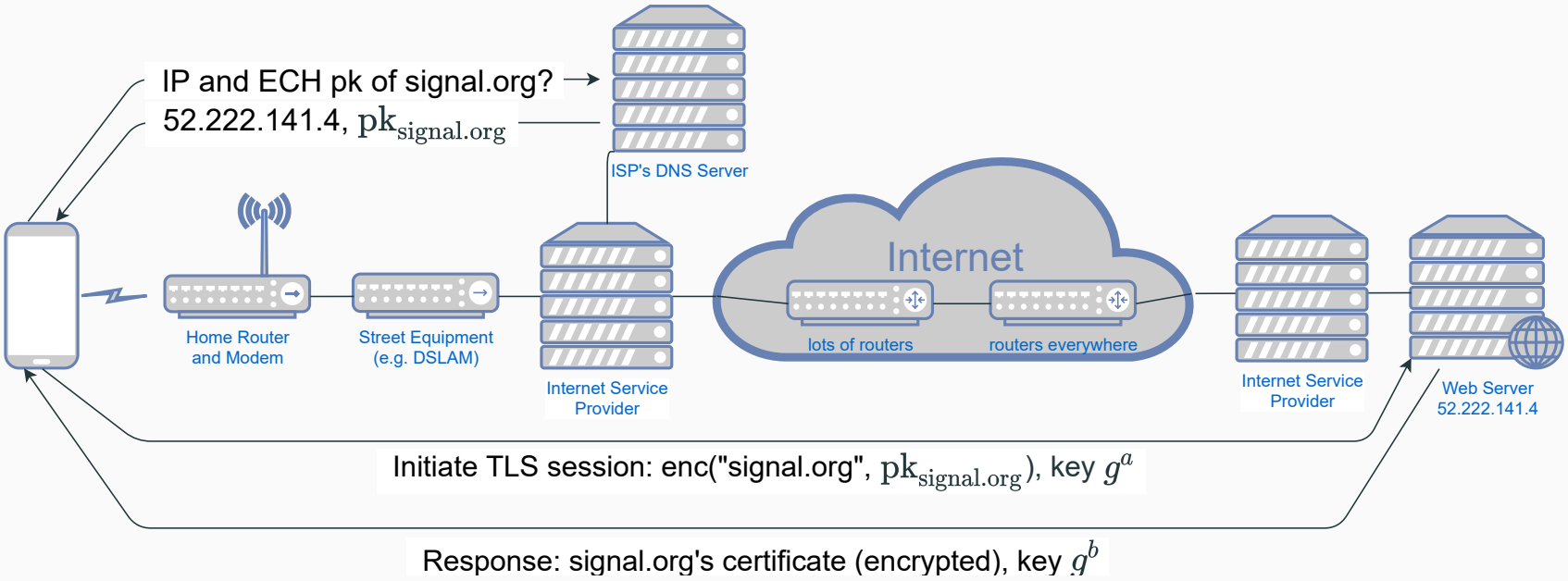
`https://blog.cloudflare.com/esni/`



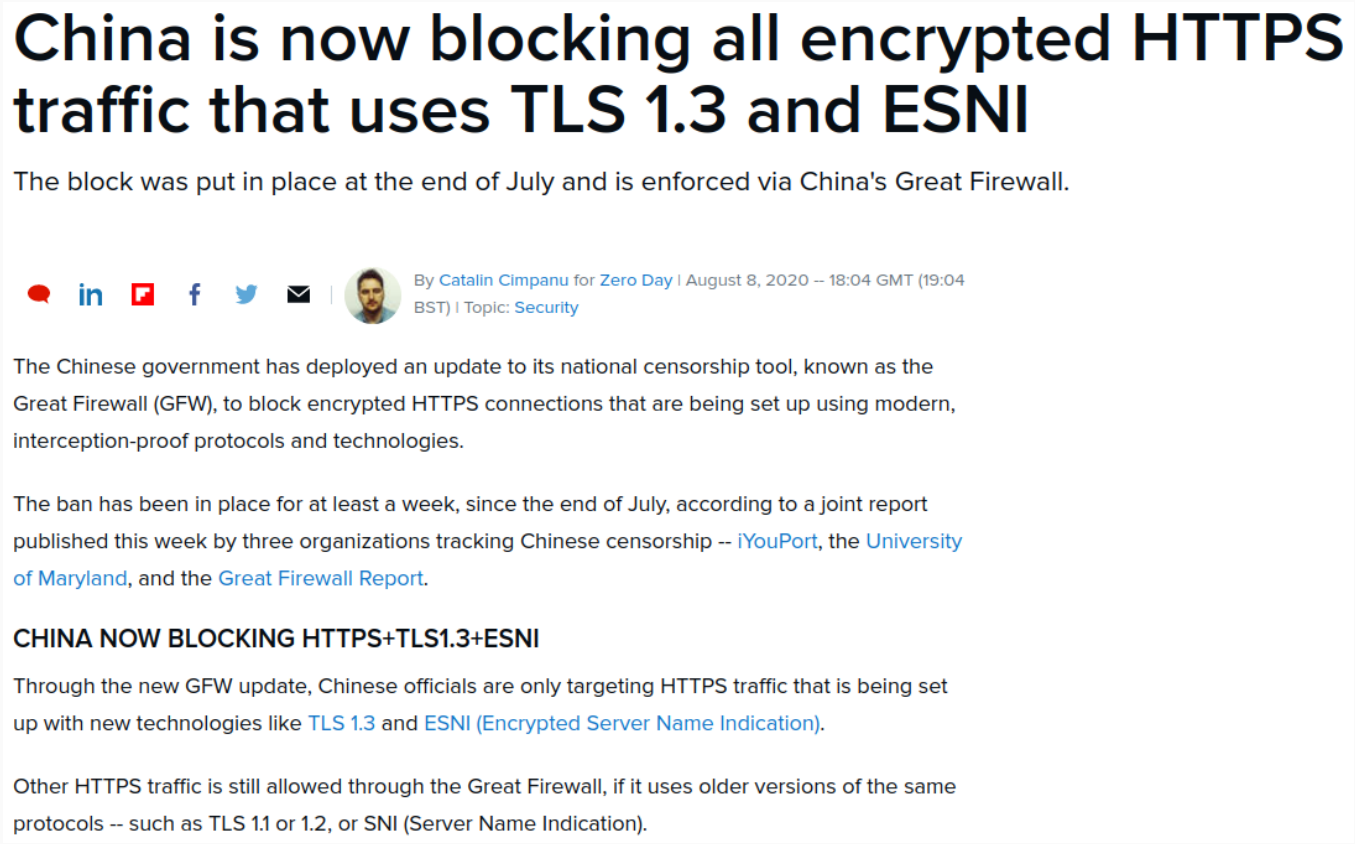
Hide the Payload of Server Name Indication

Since 2018, experiments have been conducted to encrypt the Server Name Indication

`https://blog.cloudflare.com/esni/`



China Blocking ALL Traffic Using Encrypted SNI



China is now blocking all encrypted HTTPS traffic that uses TLS 1.3 and ESNI

The block was put in place at the end of July and is enforced via China's Great Firewall.

By Catalin Cimpanu for Zero Day | August 8, 2020 -- 18:04 GMT (19:04 BST) | Topic: Security

The Chinese government has deployed an update to its national censorship tool, known as the Great Firewall (GFW), to block encrypted HTTPS connections that are being set up using modern, interception-proof protocols and technologies.

The ban has been in place for at least a week, since the end of July, according to a joint report published this week by three organizations tracking Chinese censorship -- iYouPort, the University of Maryland, and the Great Firewall Report.

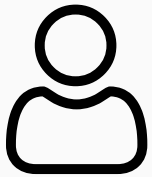
CHINA NOW BLOCKING HTTPS+TLS1.3+ESNI

Through the new GFW update, Chinese officials are only targeting HTTPS traffic that is being set up with new technologies like TLS 1.3 and ESNI (Encrypted Server Name Indication).

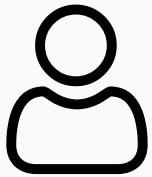
Other HTTPS traffic is still allowed through the Great Firewall, if it uses older versions of the same protocols -- such as TLS 1.1 or 1.2, or SNI (Server Name Indication).

`https://www.zdnet.com/article/china-is-now-blocking-all-encrypted-https-traffic-using-tls-1-3-and-esni/`

Public-Key Encryption



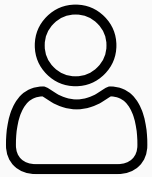
Sender



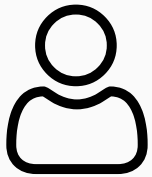
Receiver

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender

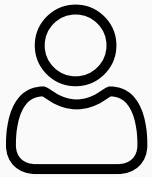


Receiver

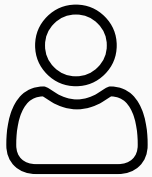
$(sk_R, pk_R) \leftarrow \text{Gen}$

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender



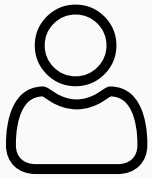
Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

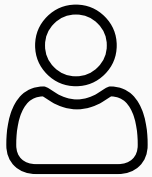
$c \leftarrow \text{pk-enc}(pk_R, m)$

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender



Receiver

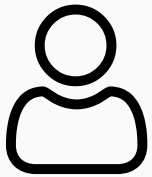
$(sk_R, pk_R) \leftarrow \text{Gen}$

$c \leftarrow \text{pk-enc}(pk_R, m)$

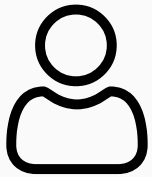
c

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender



Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

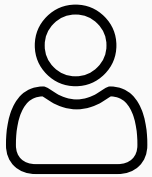
$c \leftarrow \text{pk-enc}(pk_R, m)$

c

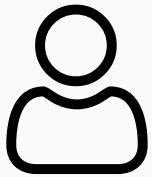
$m \leftarrow \text{pk-dec}(sk_R, c)$

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender



Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

$c \leftarrow \text{pk-enc}(pk_R, m)$

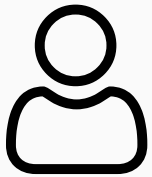
c

$m \leftarrow \text{pk-dec}(sk_R, c)$

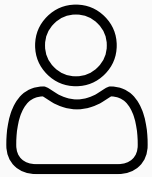
- the mathematical structure makes it slow;
  huge overhead for each message

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Public-Key Encryption



Sender



Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

$c \leftarrow \text{pk-enc}(pk_R, m)$

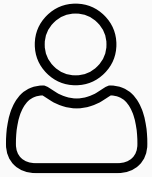
c

$m \leftarrow \text{pk-dec}(sk_R, c)$

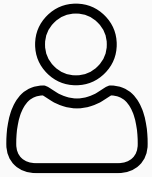
- the mathematical structure makes it slow;
  huge overhead for each message
- “easy” key distribution (identity stays complicated)

visualizations of this style are adapted from a joint talk with Doreen Riepel and Eduard Hauck

Symmetric Encryption or: Data Encapsulation Mechanism



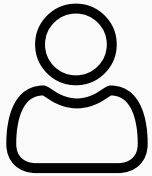
Sender



Receiver

Symmetric Encryption or: Data Encapsulation Mechanism

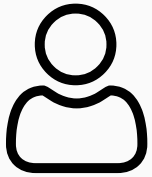
Sender



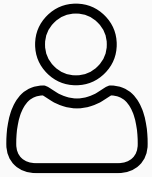
Receiver

$c \leftarrow \text{sym-enc}(k, m)$

Symmetric Encryption or: Data Encapsulation Mechanism



Sender

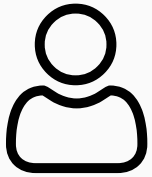


Receiver

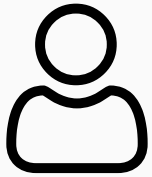
$c \leftarrow \text{sym-enc}(k, m)$

c

Symmetric Encryption or: Data Encapsulation Mechanism



Sender

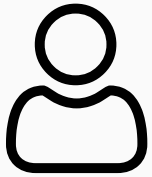


Receiver

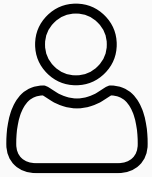
$c \leftarrow \text{sym-enc}(k, m)$

$m \leftarrow \text{sym-dec}(k, c)$

Symmetric Encryption or: Data Encapsulation Mechanism



Sender



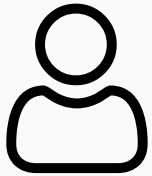
Receiver

$c \leftarrow \text{sym-enc}(k, m)$

$m \leftarrow \text{sym-dec}(k, c)$

Symmetric Encryption or: Data Encapsulation Mechanism

Sender



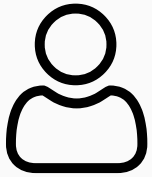
Receiver

$c \leftarrow \text{sym-enc}(k, m)$

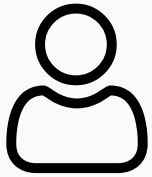
$m \leftarrow \text{sym-dec}(k, c)$

- We assume that both parties know a secret key k

Symmetric Encryption or: Data Encapsulation Mechanism



Sender



Receiver

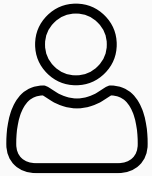
$c \leftarrow \text{sym-enc}(k, m)$

$m \leftarrow \text{sym-dec}(k, c)$

- We assume that both parties know a secret key k
- mostly operations on bits: fast

Symmetric Encryption or: Data Encapsulation Mechanism

Sender



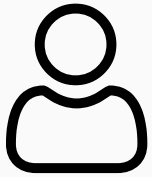
Receiver

$c \leftarrow \text{sym-enc}(k, m)$

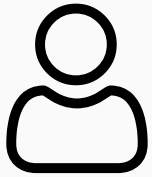
$m \leftarrow \text{sym-dec}(k, c)$

- We assume that both parties know a secret key k
- mostly operations on bits: fast
- quasi arbitrary amounts of data without overhead
  (by using a secure *operation mode*)

Symmetric Encryption or: Data Encapsulation Mechanism



Sender



Receiver

$c \leftarrow \text{sym-enc}(k, m)$

c ⟶ $m \leftarrow \text{sym-dec}(k, c)$

- We assume that both parties know a secret key k
- mostly operations on bits: fast
- quasi arbitrary amounts of data without overhead
  (by using a secure *operation mode*)
- Examples: AES-256-GCM, ChaCha20Poly1305

Hybrid Can Mean Many Things ...

- Usually: combine the best of different systems.

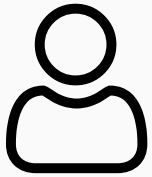
Hybrid Can Mean Many Things ...

- Usually: combine the best of different systems.
- Here: the combination of asymmetric and symmetric cryptographic systems.

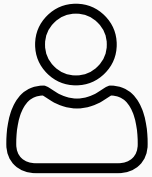
Hybrid Can Mean Many Things …

- Usually: combine the best of different systems.
- Here: the combination of asymmetric and symmetric cryptographic systems.
- Idea: Encrypt a fresh symmetric key to a public key, and use it for subsequent symmetric encryption of the payload.

Key Encapsulation Mechanism



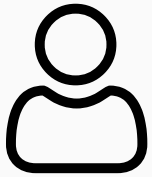
Sender



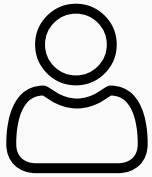
Receiver

Similar to Public-Key Encryption, but the payload is always a uniformly random key.

Key Encapsulation Mechanism



Sender

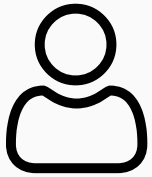


Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

Similar to Public-Key Encryption, but the payload is always a uniformly random key.

Key Encapsulation Mechanism



Sender

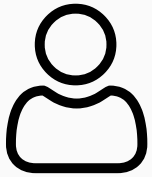
Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

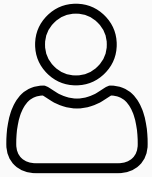
$k, c \leftarrow \text{encap}(pk_R)$

Similar to Public-Key Encryption, but the payload is always a uniformly random key.

Key Encapsulation Mechanism



Sender



Receiver

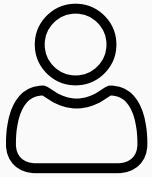
$(sk_R, pk_R) \leftarrow \text{Gen}$

$k, c \leftarrow \text{encap}(pk_R)$

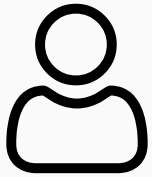
c

Similar to Public-Key Encryption, but the payload is always a uniformly random key.

Key Encapsulation Mechanism



Sender



Receiver

$(sk_R, pk_R) \leftarrow \text{Gen}$

$k, c \leftarrow \text{encap}(pk_R)$

c

$k \leftarrow \text{decap}(sk_R, c)$

Similar to Public-Key Encryption, but the payload is always a uniformly random key.

Hybrid Encryption: Use the best from both worlds

- asymmetric primitive to encrypt a symmetric key
  Key Encapsulation Mechanism (KEM)

Hybrid Encryption: Use the best from both worlds

- asymmetric primitive to encrypt a symmetric key
  Key Encapsulation Mechanism (KEM)
- symmetric primitive to encrypt the actual data
  Data Encapsulation Mechanism (DEM)

Hybrid Encryption: Use the best from both worlds

- asymmetric primitive to encrypt a symmetric key
  Key Encapsulation Mechanism (KEM)
- symmetric primitive to encrypt the actual data
  Data Encapsulation Mechanism (DEM)

HPKE roughly is computing the following

$$k, enc \leftarrow \text{encap}(pk_R)$$
$$c \leftarrow \text{sym-enc}(k, m)$$

and sending enc, c to the recipient.

Hybrid Encryption: Use the best from both worlds

- asymmetric primitive to encrypt a symmetric key
  Key Encapsulation Mechanism (KEM)
- symmetric primitive to encrypt the actual data
  Data Encapsulation Mechanism (DEM)

HPKE roughly is computing the following

$$k, enc \leftarrow \text{encap}(pk_R)$$
$$c \leftarrow \text{sym-enc}(k, m)$$

and sending enc, c to the recipient. The recipient uses

$$k \leftarrow \text{decap}(sk_R, enc)$$
$$m \leftarrow \text{sym-dec}(k, c)$$

to retrieve the message.

Why a new Standard for an old technique?

Hybrid Encryption is an old idea. Why a new standard?

- modern crypto

Why a new Standard for an old technique?

Hybrid Encryption is an old idea. Why a new standard?

- modern crypto
- provable security

Why a new Standard for an old technique?

Hybrid Encryption is an old idea. Why a new standard?

- modern crypto
- provable security
- test vectors

Why a new Standard for an old technique?

Hybrid Encryption is an old idea. Why a new standard?

- modern crypto
- provable security
- test vectors
- freely implementable (no patents or paywalls)

Cryptographic Security Properties of HPKE

"The message shall remain private"

- more specific: an adverary shall not be able to distinguish a ciphertext of a plaintext m_1 and from one of a plaintext m_2 (plaintexts of the same length)

Cryptographic Security Properties of HPKE

"The message shall remain private"

- more specific: an adverary shall not be able to distinguish a ciphertext of a plaintext m_1 and from one of a plaintext m_2
  (plaintexts of the same length)
- even if we encrypt and decrypt messages under the same key for the adversary
  (except the challenge ciphertext)

Cryptographic Security Properties of HPKE

"The message shall remain private"

- more specific: an adverary shall not be able to distinguish a ciphertext of a plaintext m_1 and from one of a plaintext m_2 (plaintexts of the same length)
- even if we encrypt and decrypt messages under the same key for the adversary (except the challenge ciphertext)
- We call this Chosen-Ciphertext Indistinguishability or IND-CCA2

Cryptographic Security Properties of HPKE

"The message shall remain private"

- more specific: an adverary shall not be able to distinguish a ciphertext of a plaintext m_1 and from one of a plaintext m_2 (plaintexts of the same length)
- even if we encrypt and decrypt messages under the same key for the adversary (except the challenge ciphertext)
- We call this Chosen-Ciphertext Indistinguishability or IND-CCA2
- this covers confidentiality and integrity

The CryptoVerif Automatic Protocol Prover

The CryptoVerif Automatic Protocol Prover

- security games in a probabilistic process calculus
  (the applied π-calculus)

The CryptoVerif Automatic Protocol Prover

initial game

- security games in a probabilistic process calculus
  (the applied π-calculus)

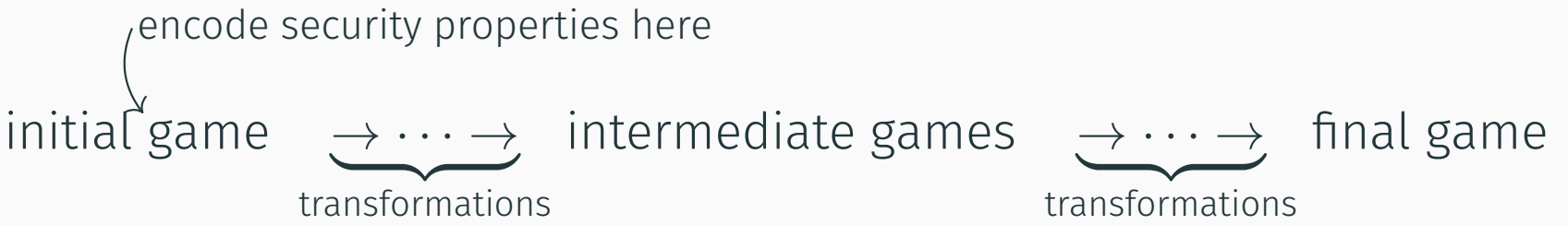
The CryptoVerif Automatic Protocol Prover

- security games in a probabilistic process calculus
  (the applied π-calculus)

The CryptoVerif Automatic Protocol Prover

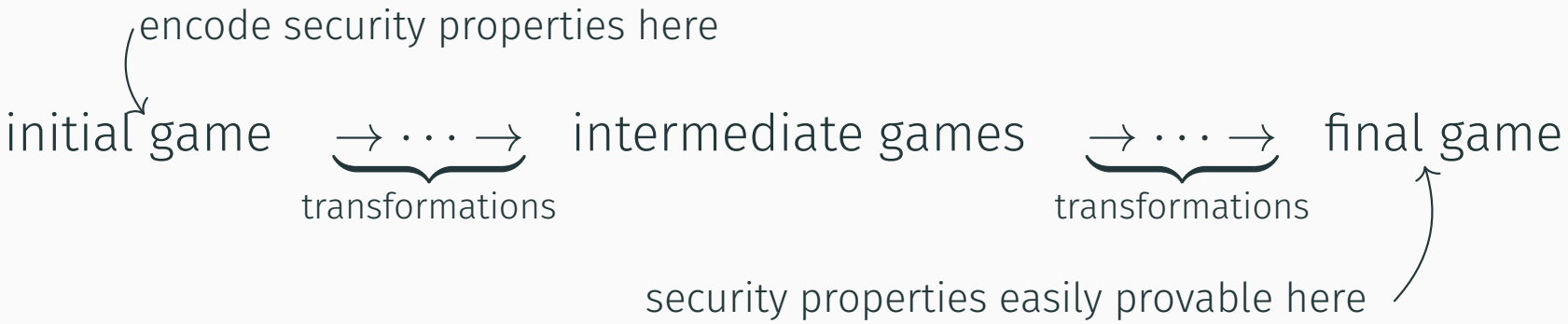
- security games in a probabilistic process calculus (the applied π-calculus)

The CryptoVerif Automatic Protocol Prover



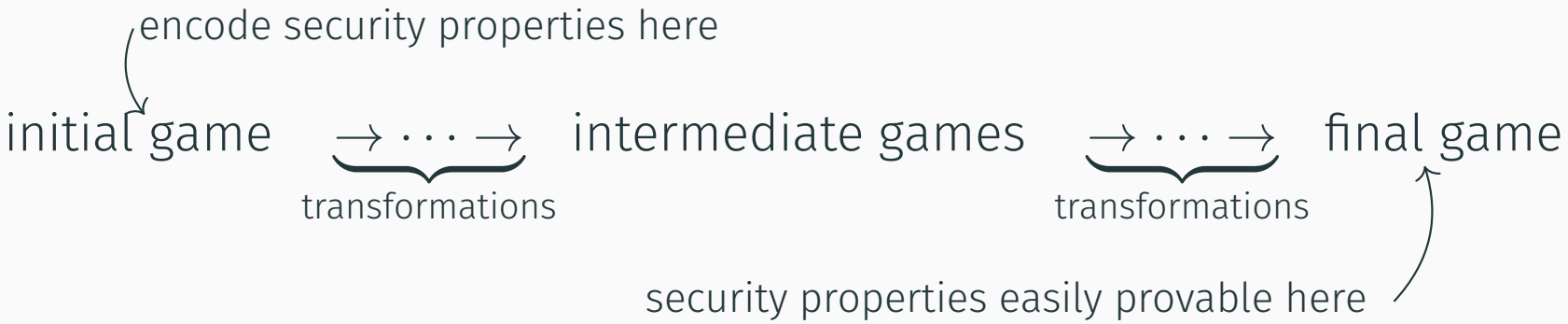
- security games in a probabilistic process calculus
  (the applied π-calculus)

The CryptoVerif Automatic Protocol Prover



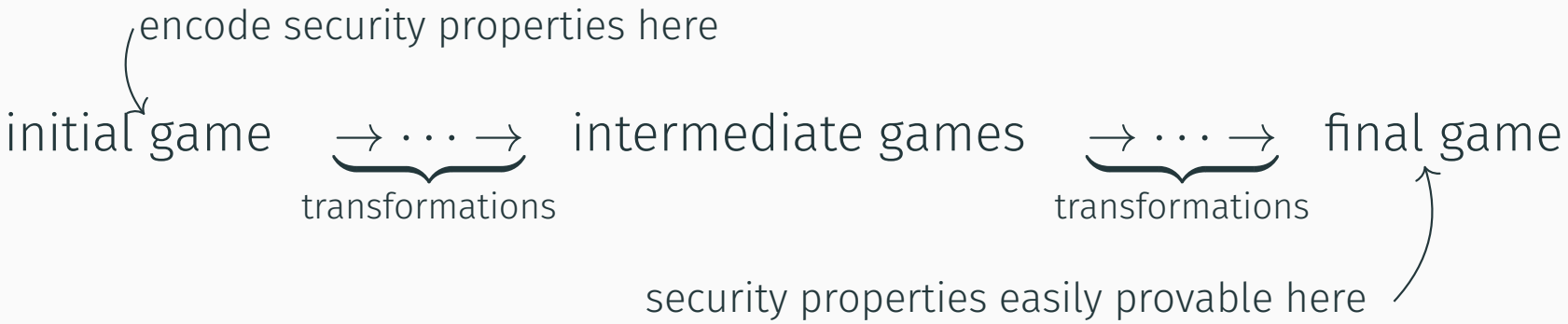
- security games in a probabilistic process calculus
  (the applied π-calculus)

The CryptoVerif Automatic Protocol Prover



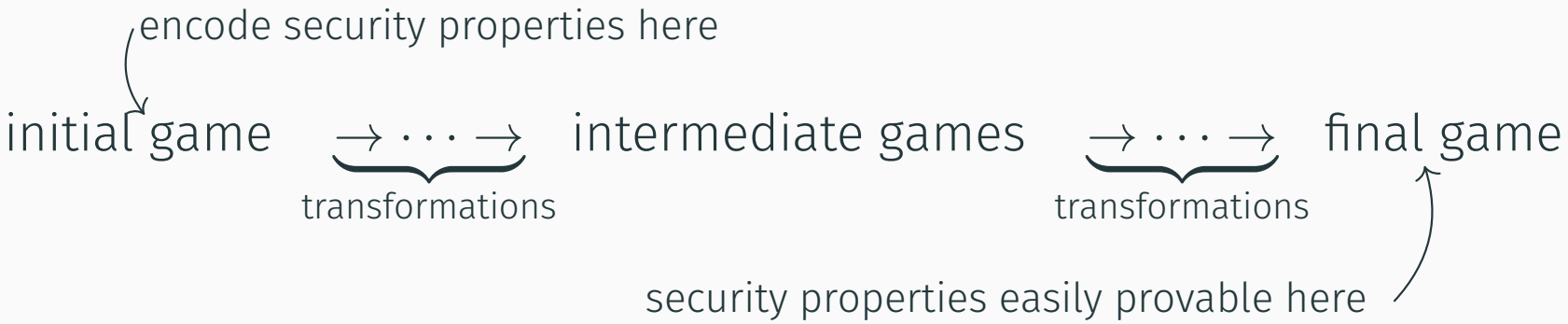
- security games in a probabilistic process calculus (the applied π-calculus)
- built-in proof strategy

The CryptoVerif Automatic Protocol Prover



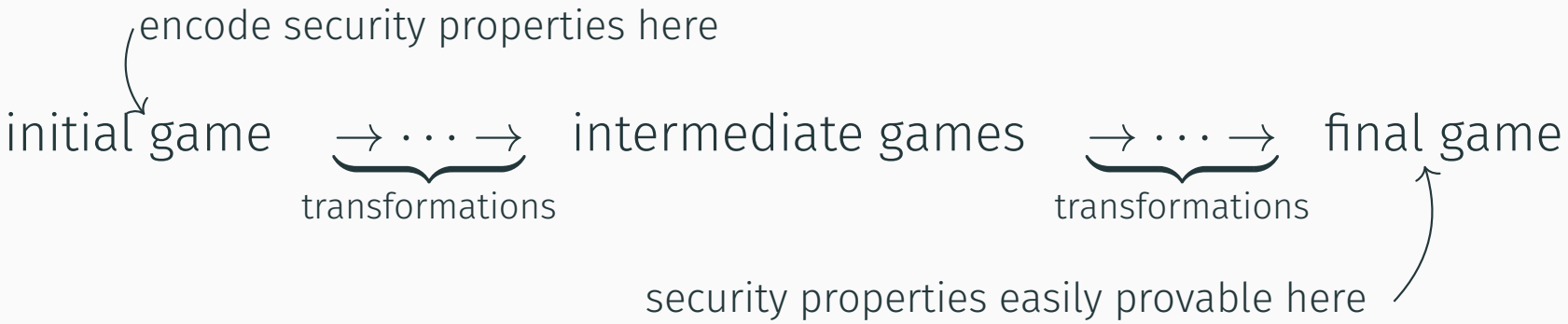
- security games in a probabilistic process calculus (the applied π-calculus)
- built-in proof strategy
- supports secrecy, correspondence, and equivalence properties

The CryptoVerif Automatic Protocol Prover



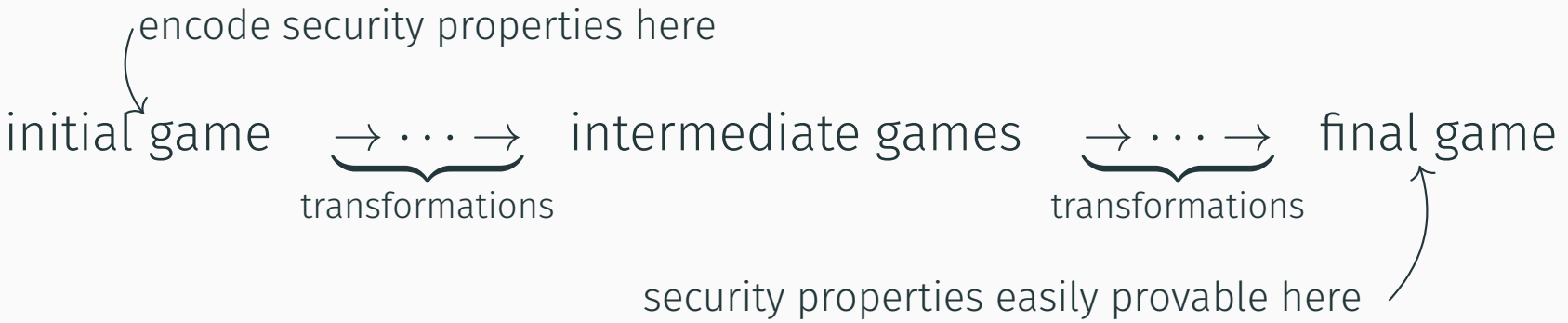
- security games in a probabilistic process calculus (the applied π-calculus)
- built-in proof strategy
- supports secrecy, correspondence, and equivalence properties
- if the proof concludes, we have asymptotic security

The CryptoVerif Automatic Protocol Prover



- security games in a probabilistic process calculus (the applied π-calculus)
- built-in proof strategy
- supports secrecy, correspondence, and equivalence properties
- if the proof concludes, we have asymptotic security
- computes exact security probability bound

The CryptoVerif Automatic Protocol Prover



- security games in a probabilistic process calculus
  (the applied π-calculus)
- built-in proof strategy
- supports secrecy, correspondence, and equivalence properties
- if the proof concludes, we have asymptotic security
- computes exact security probability bound
  depending on number of queries, runtime of adversary, length of inputs

Verified Implementations

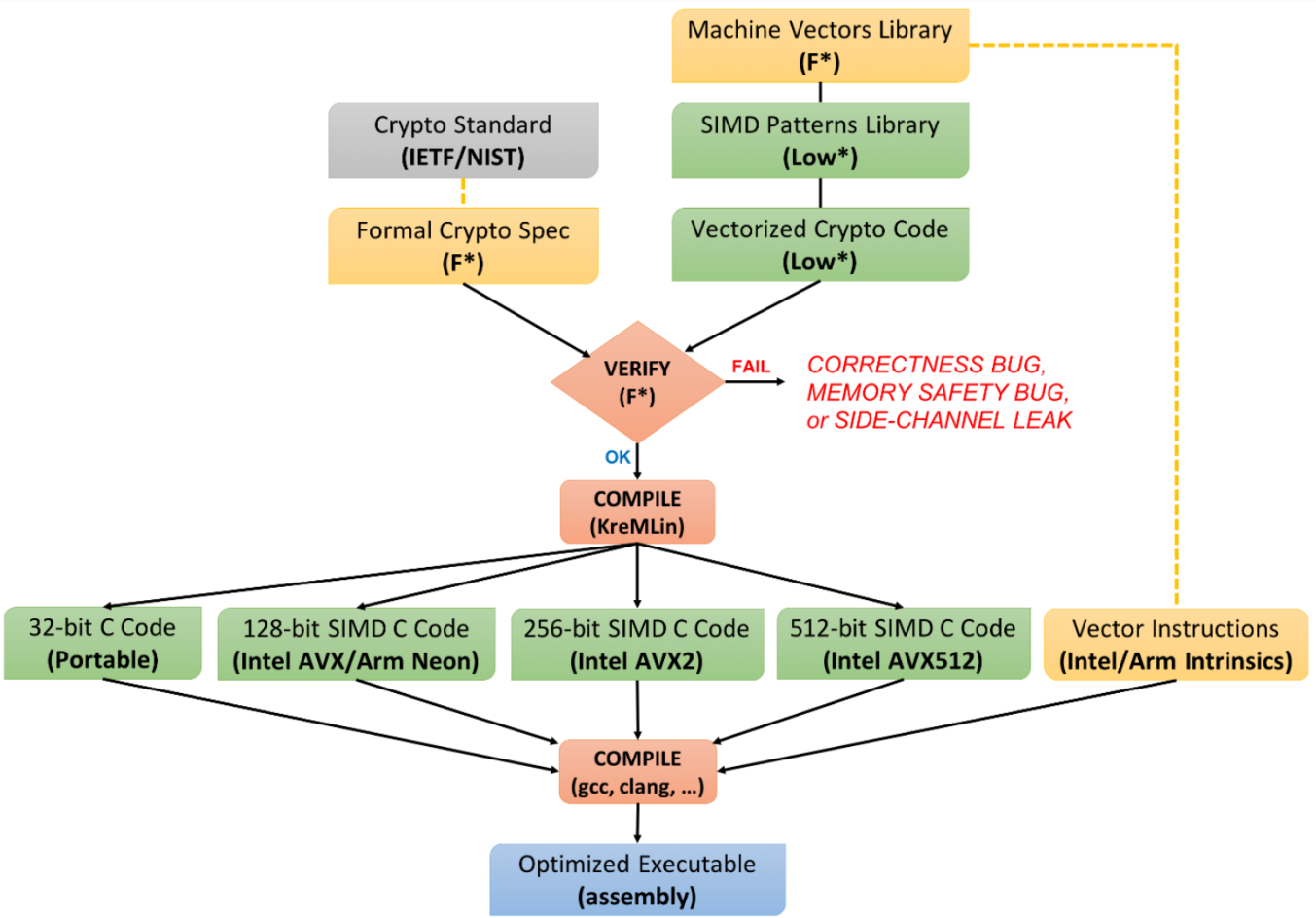
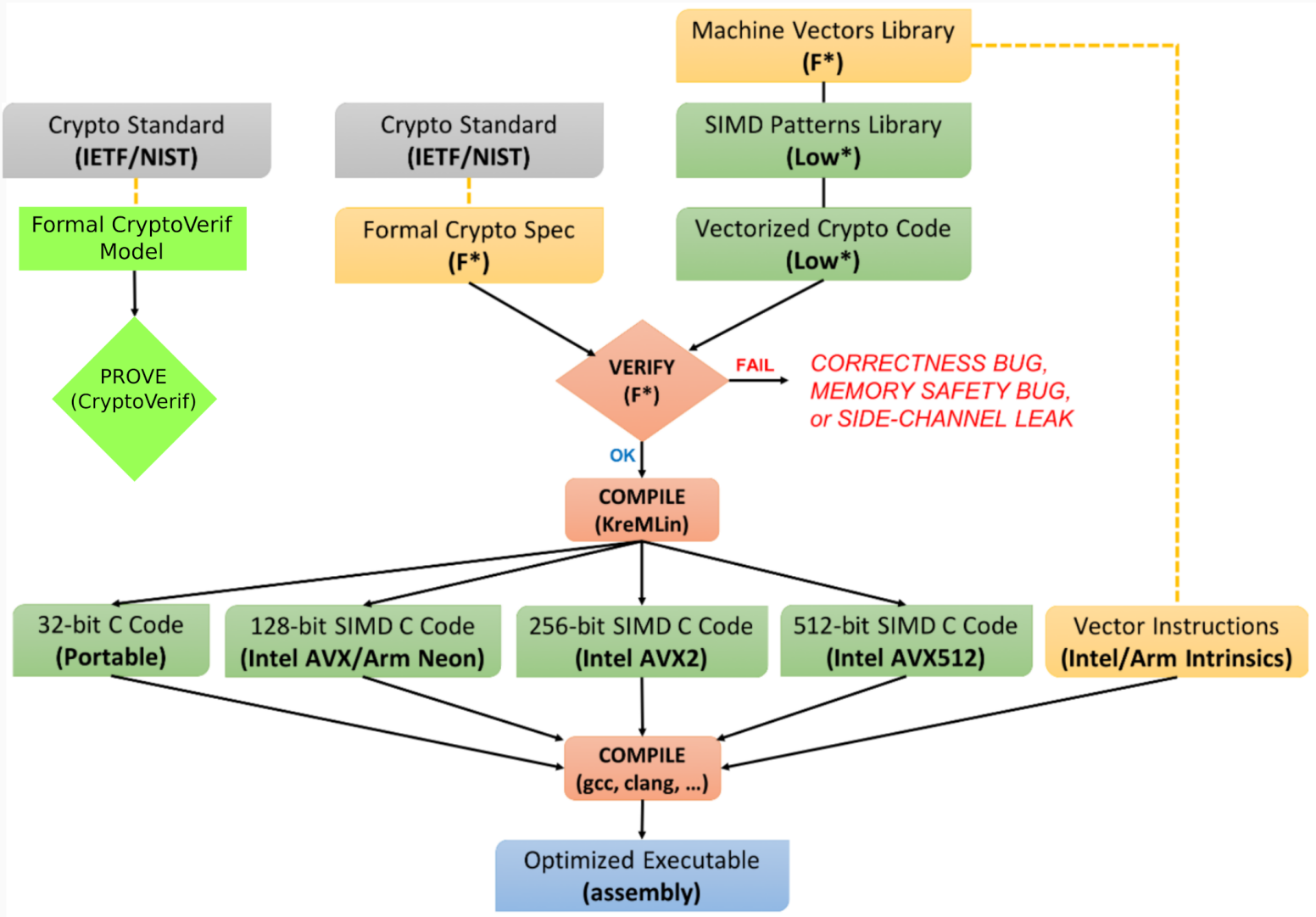


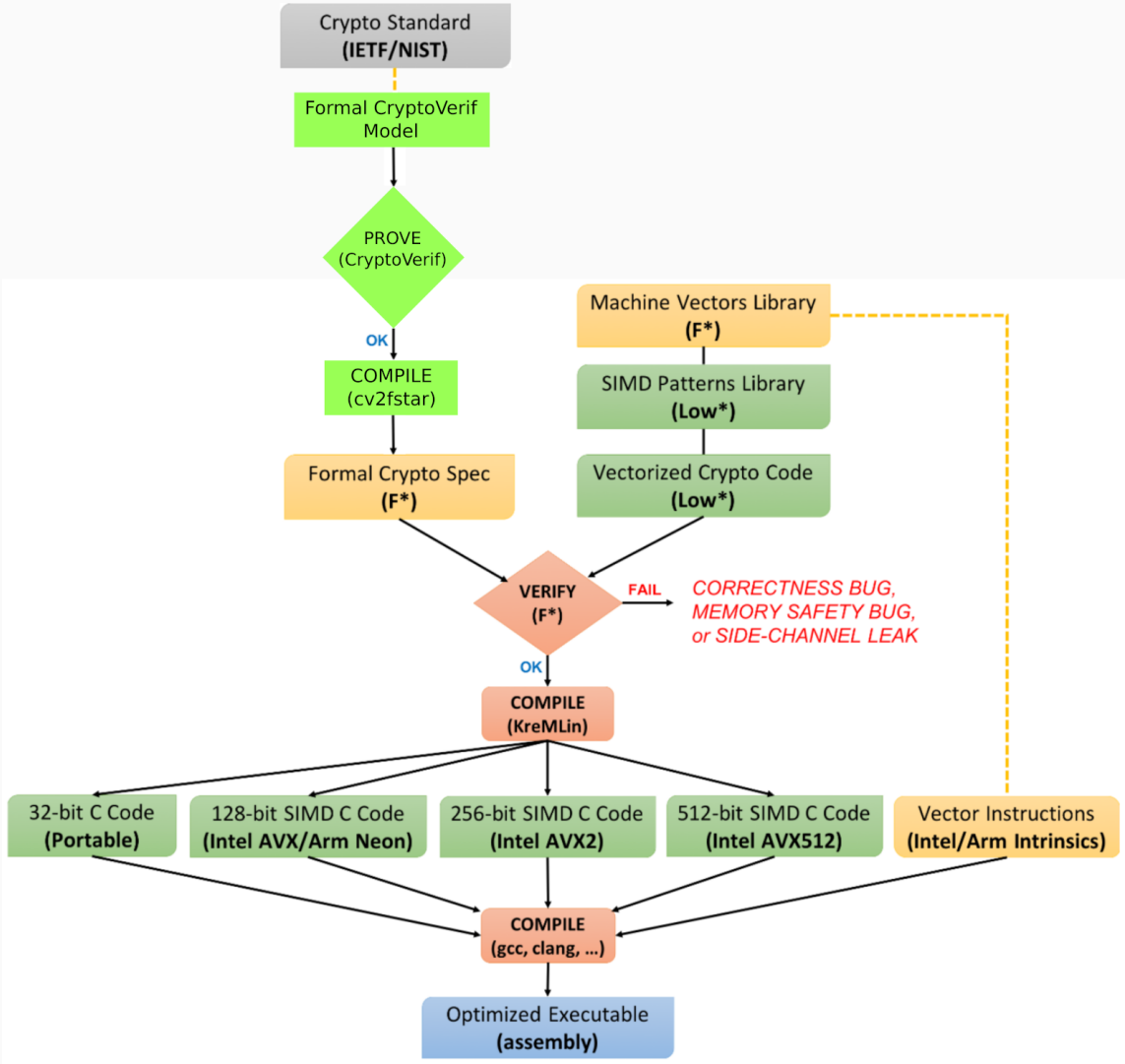
Figure 1: HACL×N programming and verification workflow.

Marina Polubelova, Karthikeyan Bhargavan, Jonathan Protzenko, Benjamin Beurdouche, Aymeric Fromherz, Natalia Kulatova, Santiago Zanella-Béguelin: HACLxN: Verified Generic SIMD Crypto (for all your favorite platforms) `https://ia.cr/2020/572`

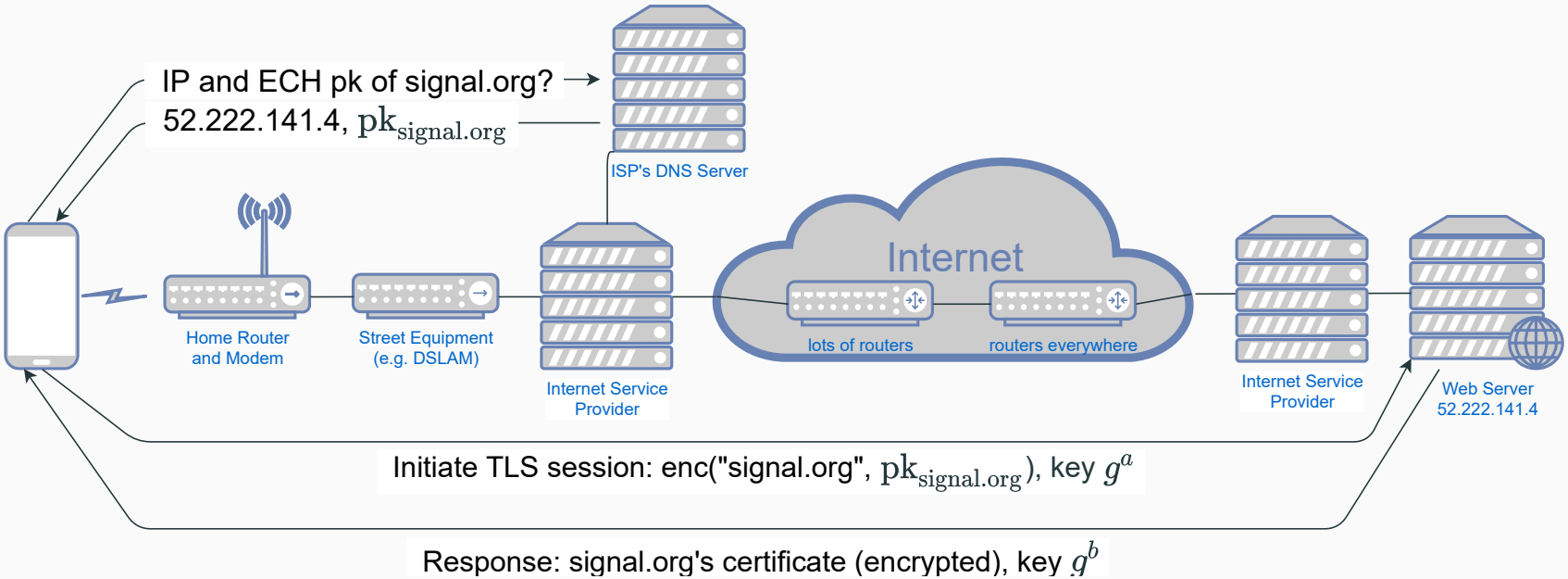
Verified Implementations and Cryptographic Proofs



Link between Implementations and Cryptographic Proofs



Conclusion



- The new Hybrid Public Key Encryption standard: `tools.ietf.org/html/draft-irtf-cfrg-hpke-07` by Richard L. Barnes, Karthik Bhargavan, Benjamin Lipp, Christopher A. Wood
- The under-submission paper analysing cryptographic security: Analysing the HPKE Standard, `ia.cr/2020/1499` by Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, Doreen Riepel