

Analysing the HPKE Standard

Joël Alwen¹, Bruno Blanchet³, Eduard Hauck², Eike Kiltz², Benjamin Lipp³,
Doreen Riepel²

October 18, 2021

Wickr¹, Ruhr-University Bochum², Inria Paris³

Hybrid Public Key Encryption (HPKE)

- *Hybrid* in the spirit of the KEM/DEM paradigm:
asymmetric building block as Key Encapsulation Mechanism (KEM),
symmetric building block as Data Encapsulation Mechanism (DEM)
- Standard in development by the Crypto Forum Research Group
<https://github.com/cfrg/draft-irtf-cfrg-hpke>
Usage in TLS 1.3's Encrypted Client Hello (ECH) extension, and
the Messaging Layer Security (MLS) group messaging protocol

HPKE Specifies Different APIs

- Single-Shot Encryption: encrypt 1 message
- Single-Shot Secret Export: provide 1 secret of (almost) *arbitrary* length
- Multi-Shot: multiple messages, and multiple secrets

This work focuses on Single-Shot Encryption.

HPKE Specifies Different Modes

Modes vary in the involved long-term key material.

Mode names reflect authentication guarantees (to some extent).

	receiver key pair	sender key pair	pre-shared key
Base	y	n	n
Auth	y	y	n
PSK	y	n	y
AuthPSK	y	y	y

This work focuses on Auth mode.

Syntax: Authenticated Public Key Encryption (APKE)



Sender

$$(sk_S, pk_S) \stackrel{\$}{\leftarrow} \text{Gen}$$

$$c \stackrel{\$}{\leftarrow} \text{AuthEnc}(sk_S, pk_R, m)$$



Receiver

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

$$m \leftarrow \text{AuthDec}(sk_R, pk_S, c)$$



Syntax: Auth. Key Encapsulation Mechanism (AKEM)



Sender

$$(sk_S, pk_S) \stackrel{\$}{\leftarrow} \text{Gen}$$

$$(c, K) \stackrel{\$}{\leftarrow} \text{AuthEncap}(sk_S, pk_R)$$



Receiver

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

$$K \leftarrow \text{AuthDecap}(sk_R, pk_S, c)$$

Syntax: KeySchedule

The AKEM shared secret is not used directly by the APKE.

A key schedule function is used to derive an AEAD key and nonce from it:

$$key, nonce \leftarrow \text{KeySchedule}(K, info)$$

info is an arbitrary bitstring chosen by the application.

Generic APKE Construction

$\text{AuthEnc}(sk_S, pk_R, m, info)$:

```
     $c_1, K \xleftarrow{\$} \text{AuthEncap}(sk_S, pk_R)$   
     $key, nonce \leftarrow \text{KeySchedule}(K, info)$   
     $c_2 \leftarrow \text{Enc}_{\text{DEM}}(key, nonce, m)$   
return  $c_1, c_2$ 
```


Security Notions

Chosen-Ciphertext Indistinguishability (CCA)

confidentiality of KEM encapsulations and PKE ciphertexts

Authenticity (Auth)

unforgeability of KEM encapsulations and PKE ciphertexts

Security Experiments for APKE: Confidentiality

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Security Experiments for APKE: Confidentiality

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Outsider-CCA:

$\text{CHALL}(i \in [n], j \in [n], m_0, m_1)$ returns c of m_b .

Adv wins if guesses b correctly.

Security Experiments for APKE: Confidentiality

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Outsider-CCA:

$\text{CHALL}(i \in [n], j \in [n], m_0, m_1)$ returns c of m_b .

Adv wins if guesses b correctly.

Insider-CCA:

$\text{CHALL}(sk, j \in [n], m_0, m_1)$ returns c of m_b .

Adv wins if guesses b correctly.

Security Experiments for APKE: Authentication

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Security Experiments for APKE: Authentication

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Outsider-Auth:

$\text{CHALL}(i \in [n], j \in [n], c)$

Adv wins if c was not produced by AENC and decrypts correctly.

Security Experiments for APKE: Authentication

Setup:

n honest key pairs

b secret bit

Oracles:

$\text{AENC}(i \in [n], pk, m)$ returns c

$\text{ADec}(pk, j \in [n], c)$ returns m , or \perp if c was produced by AENC

Outsider-Auth:

$\text{CHALL}(i \in [n], j \in [n], c)$

Adv wins if c was not produced by AENC and decrypts correctly.

Insider-Auth:

$\text{CHALL}(i \in [n], sk, c)$

Adv wins if c was not produced by AENC and decrypts correctly.

Security Experiments for AKEM: Outsider-CCA

Setup:

n honest key pairs

Oracles for Outsider-CCA:

$\text{AENCAP}(i \in [n], pk)$ returns c, K , with K real-or-random.

$\text{ADECAP}(pk, j \in [n], c)$ returns K , being consistent with AENCAP .

Adv wins if it can distinguish between real and random.

Security Experiments for AKEM: Insider-CCA

Setup:

n honest key pairs

Oracles for Insider-CCA:

$\text{AENCAP}(i \in [n], pk)$ returns real c, K .

$\text{CHALL}(sk, j \in [n])$ returns c, K , with K real-or-random.

$\text{ADECAP}(pk, j \in [n], c)$ returns K , being consistent with CHALL .

Adv wins if it can distinguish between real and random.

Security Experiments for AKEM: Outsider-Auth

Setup:

n honest key pairs

Oracles for Outsider-Auth:

$\text{AENCAP}(i \in [n], pk)$ returns c, K .

$\text{ADECAP}(pk, j \in [n], c)$ returns K , with K real-or-random, and being consistent with AENCAP .

Adv wins if it can distinguish between real and random.

Three Composition Theorems Proved in This Work

Outsider-CCA and Insider-CCA:

If AKEM is (Outsider or Insider)-CCA, and KS is a PRF, and DEM is IND-CPA+INT-CTXT,
then APKE is (Outsider or Insider)-CCA.


Outsider-Auth:


If AKEM is Outsider-Auth and Outsider-CCA, and KS is a PRF, and DEM is INT-CTXT,
then APKE is Outsider-Auth.

Insider-Auth:


There is a concrete attack on the generic construction!


DH-AKEM for Prime-Order Groups


$$\left. \begin{array}{l} sk_S \leftarrow \mathbb{Z}_p^* \\ pk_S = g^{sk_S} \end{array} \right\} \text{Gen}$$


$$\begin{array}{l} sk_R \leftarrow \mathbb{Z}_p^* \\ pk_R = g^{sk_R} \end{array}$$

DH-AKEM for Prime-Order Groups


$$\left. \begin{array}{l} sk_S \leftarrow \mathbb{Z}_p^* \\ pk_S = g^{sk_S} \end{array} \right\} \text{Gen}$$


$$\begin{array}{l} sk_R \leftarrow \mathbb{Z}_p^* \\ pk_R = g^{sk_R} \end{array}$$

$$\begin{array}{l} sk_E \leftarrow \mathbb{Z}_p^* \\ pk_E = g^{sk_E} \\ c = pk_E \end{array}$$


$$\text{context} = c \parallel pk_R \parallel pk_S$$


$$\text{dh} = pk_R^{sk_E} \parallel pk_R^{sk_S}$$

$$K = H(\text{context}, \text{dh})$$

AuthEncap

DH-AKEM for Prime-Order Groups


$$\left. \begin{array}{l} sk_S \leftarrow \mathbb{Z}_p^* \\ pk_S = g^{sk_S} \end{array} \right\} \text{Gen}$$


$$\begin{array}{l} sk_R \leftarrow \mathbb{Z}_p^* \\ pk_R = g^{sk_R} \end{array}$$

$$\begin{array}{l} sk_E \leftarrow \mathbb{Z}_p^* \\ pk_E = g^{sk_E} \\ c = pk_E \end{array}$$





$$\text{context} = c \parallel pk_R \parallel pk_S$$

$$\text{dh} = pk_R^{sk_E} \parallel pk_R^{sk_S}$$

$$K = H(\text{context}, \text{dh})$$

DH-AKEM for Prime-Order Groups


$$\left. \begin{array}{l} sk_S \leftarrow \mathbb{Z}_p^* \\ pk_S = g^{sk_S} \end{array} \right\} \text{Gen}$$


$$\begin{array}{l} sk_R \leftarrow \mathbb{Z}_p^* \\ pk_R = g^{sk_R} \end{array}$$

$$\begin{array}{l} sk_E \leftarrow \mathbb{Z}_p^* \\ pk_E = g^{sk_E} \\ c = pk_E \end{array}$$



$$\begin{array}{l} \text{context} = c \parallel pk_R \parallel pk_S \\ dh = pk_R^{sk_E} \parallel pk_R^{sk_S} \\ K = H(\text{context}, dh) \end{array}$$

AuthDecap

$$\left\{ \begin{array}{l} \text{context} = c \parallel pk_R \parallel pk_S \\ dh = c^{sk_R} \parallel pk_S^{sk_R} \\ K = H(\text{context}, dh) \end{array} \right.$$

Elliptic Curves and Nominal Groups

The HPKE standard allows for different elliptic curves, in particular the NIST curves P-256, P-384, P-521, as well as Curve25519 and Curve448.

- NIST curves are **prime-order groups**. In particular, secret keys are chosen uniformly at random from \mathbb{Z}_p^* .
- Curve25519 and Curve448 are **not prime-order groups**. For each honestly generated public key, there is a small number of equivalent public keys.

Elliptic Curves and Nominal Groups

The HPKE standard allows for different elliptic curves, in particular the NIST curves P-256, P-384, P-521, as well as Curve25519 and Curve448.

- NIST curves are **prime-order groups**. In particular, secret keys are chosen uniformly at random from \mathbb{Z}_p^* .
- Curve25519 and Curve448 are **not prime-order groups**. For each honestly generated public key, there is a small number of equivalent public keys.

A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol. LBB, EuroSP 2019.

<https://hal.inria.fr/hal-02100345>

Elliptic Curves and Nominal Groups

In short: We do not assume a group structure, but only an exponentiation function with certain properties.

A nominal group \mathcal{N} consists of:

- finite set of elements \mathcal{G}
- base element g
- eff. comp. exponentiation function $\text{exp} : \mathcal{G} \times \mathbb{Z} \rightarrow \mathcal{G}$.
We write X^y for $\text{exp}(X, y)$.
- finite sets: honest exponents $\mathcal{E}_H \subset \mathbb{Z}$, exponents $\mathcal{E}_U \subset \mathbb{Z}$

exp is required to fulfill:

1. $(X^y)^z = X^{yz}$ for all $X \in \mathcal{G}$, $y, z \in \mathbb{Z}$ (correctness of Diffie-Hellman)
2. the discrete logarithm is unique in the set \mathcal{E}_U — needed to define the Diffie-Hellman oracle
3. \mathcal{N} is *rerandomisable* if rerandomisation preserves the distribution of public keys for secret keys chosen in \mathcal{E}_U

Elliptic Curves and Nominal Groups: Parameters

Let D_H be the uniform distribution on \mathcal{E}_H (honestly generated exponents).

Let D_U be the uniform distribution on \mathcal{E}_U .

Let $\Delta_{\mathcal{N}} := \Delta[D_H, D_U]$ be the statistical distance between the two distributions.

	P-256	P-384	P-521	Curve25519	Curve448
Security level $\kappa_{\mathcal{N}}$ (bits)	128	192	256	128	224
$\Delta_{\mathcal{N}} \leq$	0	0	0	2^{-125}	2^{-220}

For each honest secret key (exponent), we'll need to add a $\Delta_{\mathcal{N}}$ to the adversary advantage during the proof.

(for the definition of GapDH, see the paper.)

Theorems about DH-AKEM

Outsider-CCA and Insider-CCA:

GapDH assumption in rerandomisable nominal group \mathcal{N} and modeling H as a random oracle.

Outsider-Auth:

Square-GapDH assumption in rerandomisable nominal group \mathcal{N} and modeling H as a random oracle.

(exact security bounds differ between the three, see paper)

Exact Security of HPKE

Algorithm choices with their security level:

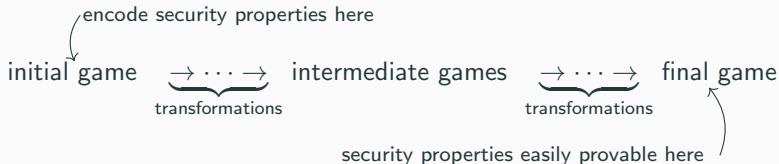
- Elliptic curves from 128 to 256 bit
- Hash functions from 256 to 512 bit
- AEAD with key length from 128 to 256 bit, the auth tag length is always 128 bit.

Our proofs find that the length of the auth tag limits the overall security level to 128 bit.

Current results on multi-key security of AEAD schemes are not sufficient to guarantee the expected security level of, e.g., AES-128-GCM.

The CryptoVerif Proof Assistant

Proof assistant for game-based, code-based cryptographic proofs



- supports secrecy, authentication, and indistinguishability properties
- built-in proof strategy, and can be guided in detail
- if the proof concludes, we have asymptotic security
- computes exact security probability bound
 - depending on number of queries, runtime of adversary, length of inputs

More Notes on CryptoVerif

- Assumptions have to be expressed in multi-instance versions
 - a proof of 2-user security \Rightarrow n -user security cannot be expressed
 - in this work, better security bound when starting directly with multi-user notion!
- There are other limitations, which is what lets CryptoVerif go far enough to treat large protocols! (SSH, TLS, WireGuard, ...)
- CryptoVerif learning material available at <https://cryptoverif.inria.fr/tutorial>

Conclusion, Contributions of This Work

- HPKE Auth mode satisfies its desired security properties
 - Proofs of composition theorems for Outsider-CCA, Insider-CCA, and Outsider-Auth of the AKEM/DEM construction
 - Hand-written non-tight proof of single-user/two-user \Rightarrow multi-user security notions for AKEM, to close gap to proofs of, e.g., PQ KEMs
 - Proof of PRF-security of HPKE's KeySchedule
 - Proofs for Outsider-CCA, Insider-CCA, Outsider-Auth of DH-AKEM
- Introduction of (Rerandomisable) Nominal Groups to cover non-prime-order groups

Paper: <https://eprint.iacr.org/2020/1499>

CryptoVerif models: <https://doi.org/10.5281/zenodo.4297811>