Password-Based AKEM and HPKE

[1]**Axel Durbet**, [1]Reihaneh Safavi-Naini and [2]Jean-François Biasse

[1]University of Calgary, Calgary, Canada
[2]University of South Florida, Tampa, Florida, USA

Table of Contents

Our Work

Password Authenticated Key Encapsulation Mechanism (PAKEM)

- Establish a session key in one round
- Authenticate the client using a password
- Achieve confidentiality, authentication, and password safety

Password Authenticated Hybrid Encryption (PAHE)

- Extend PAKEM to arbitrary-length messages
- Authenticate the client via password
- Preserve confidentiality, authentication, and password safety

Filling the Gap?

Key Establishment

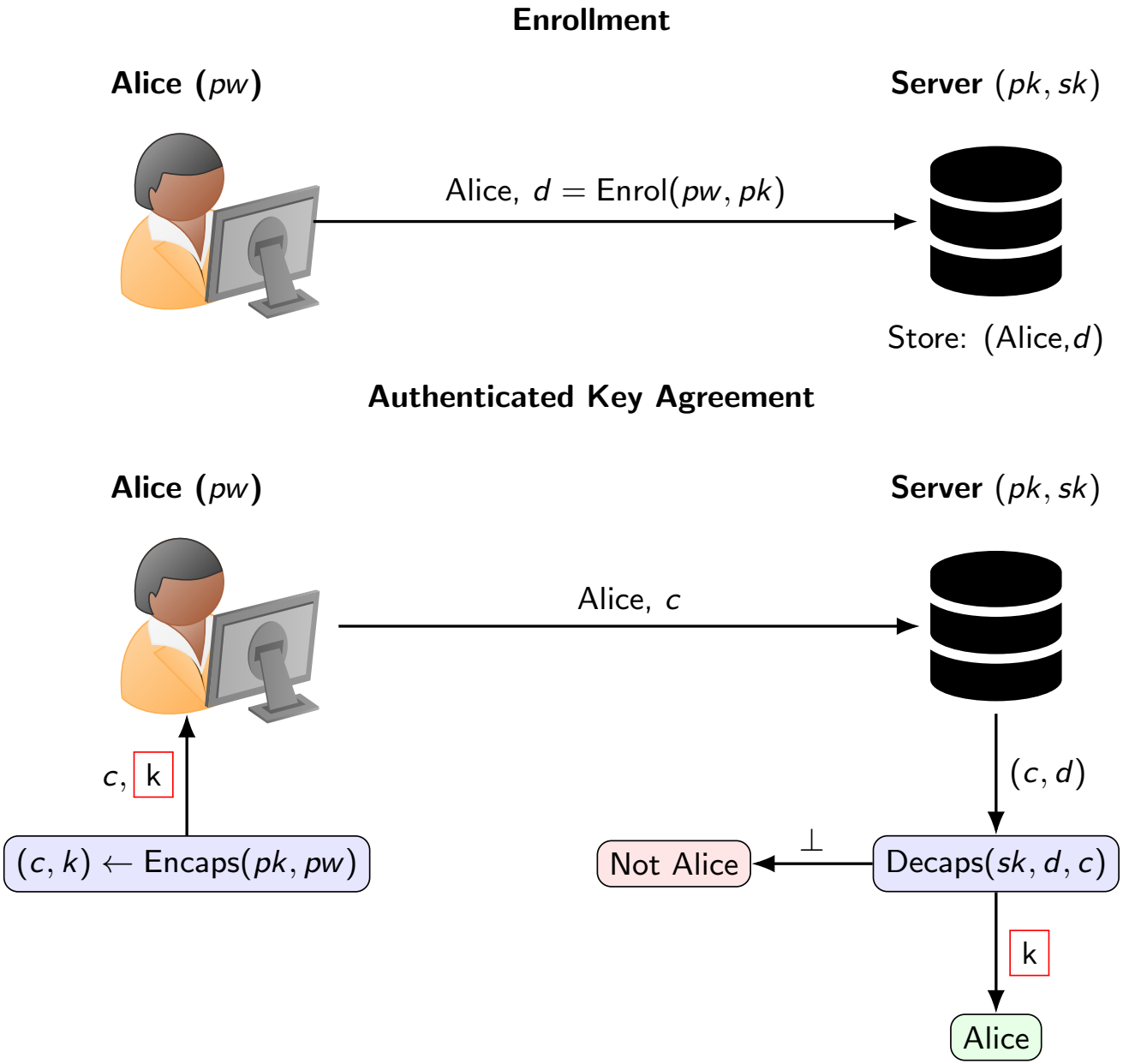
	One Message	Authenticated	via Password?
KEM	✓	×	×
AKE	×	✓	×
AKEM	✓	✓	×
PAKE	×	✓	✓
PAKEM	✓	✓	✓

Hybrid Encryption

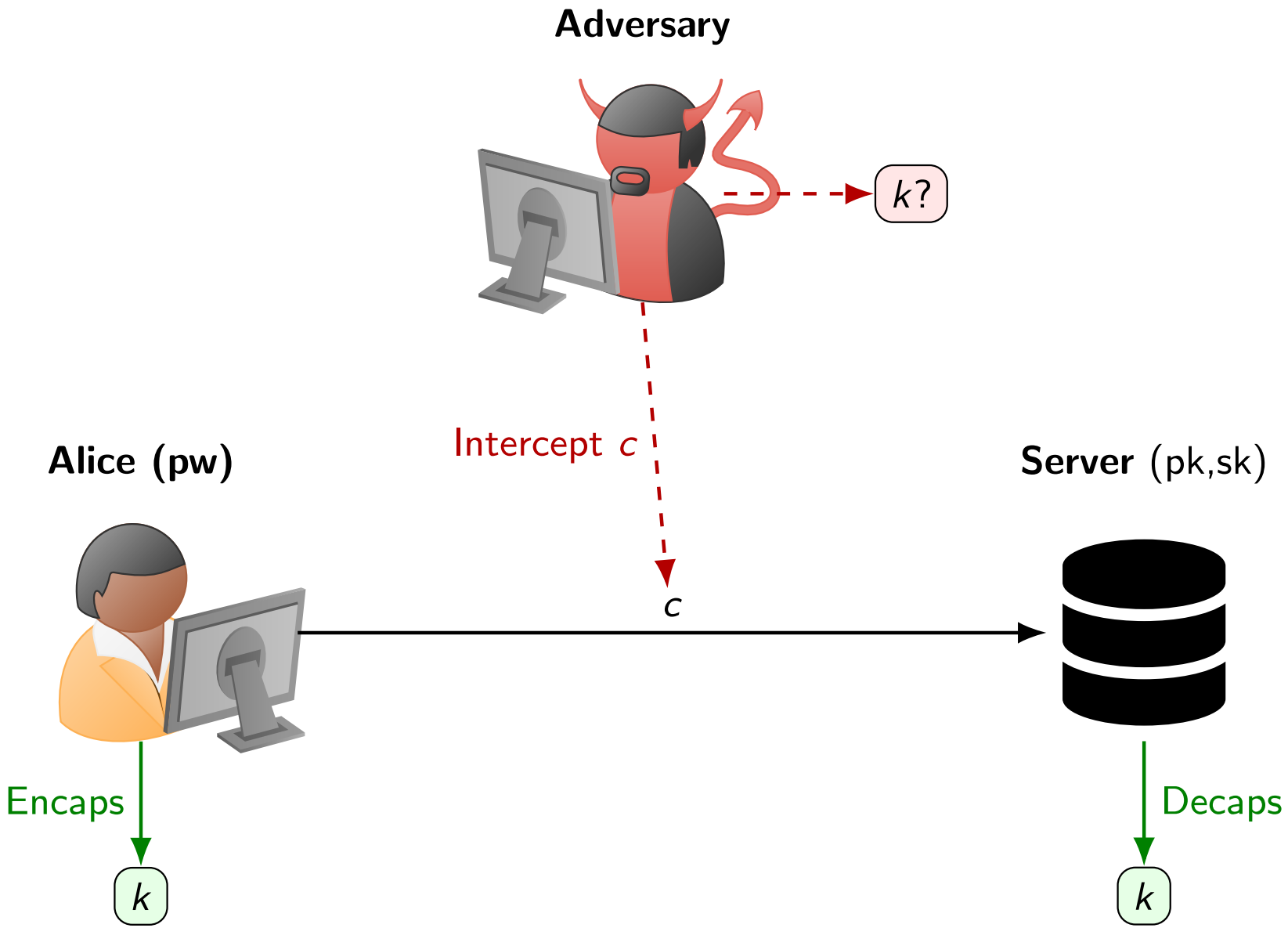
	One Message	Authenticated	via Password?	Arbitrary Length
KEM/DEM	✓	×	×	✓
PAKE + SKE	×	✓	✓	✓
HPKE	✓	✓	×	✓
PAPKE	✓	✓	✓	×
PAHE	✓	✓	✓	✓

Table of Contents

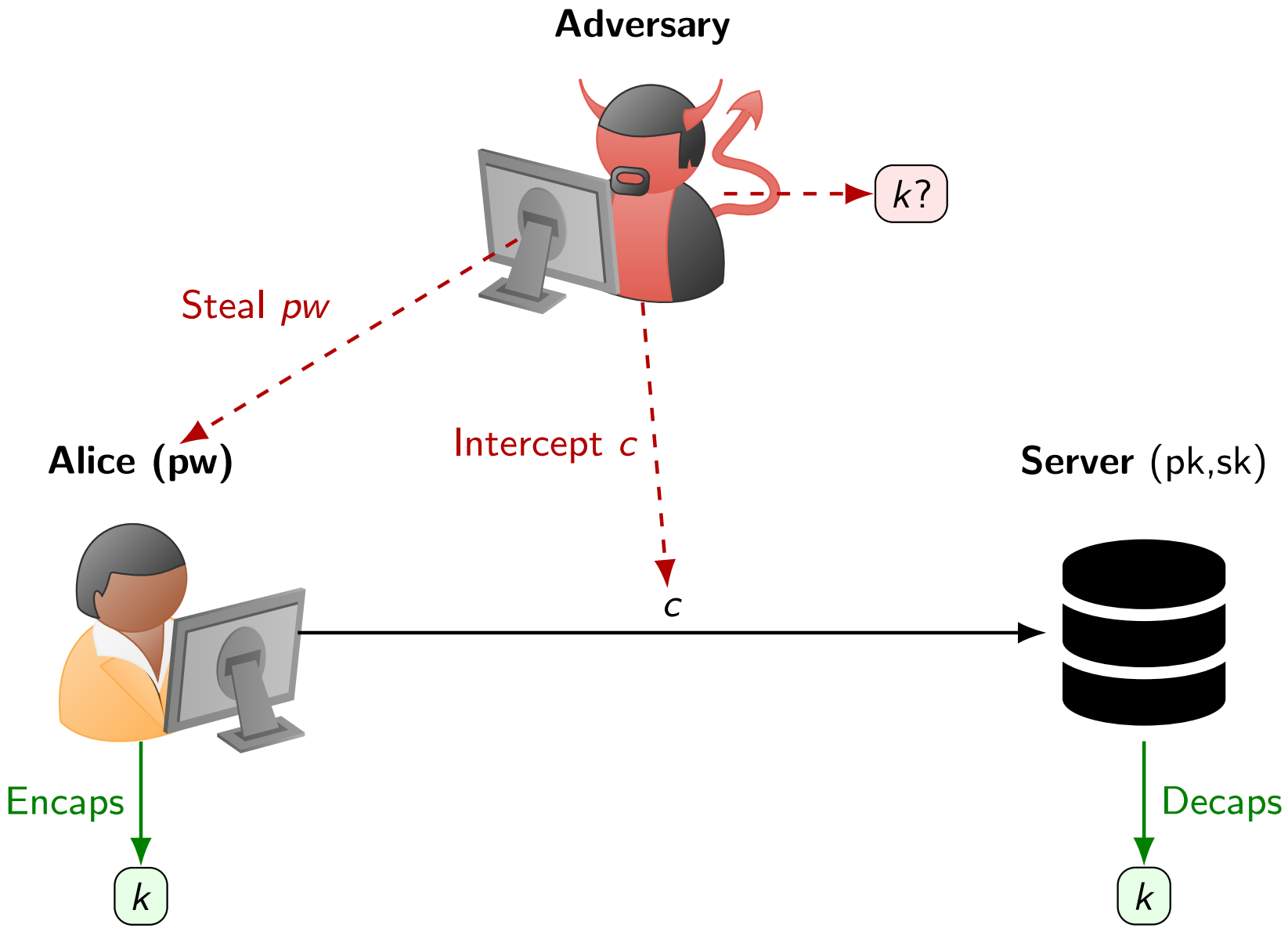
Overview


Enrollment
Alice (pw)
Server (pk, sk)
Alice, $d = \text{Enrol}(pw, pk)$
Store: (Alice, d)
Authenticated Key Agreement
Alice (pw)
Server (pk, sk)
Alice, c
c, k
(c, d)
$(c, k) \leftarrow \text{Encaps}(pk, pw)$
Not Alice
$\perp$
$\text{Decaps}(sk, d, c)$
k
Alice

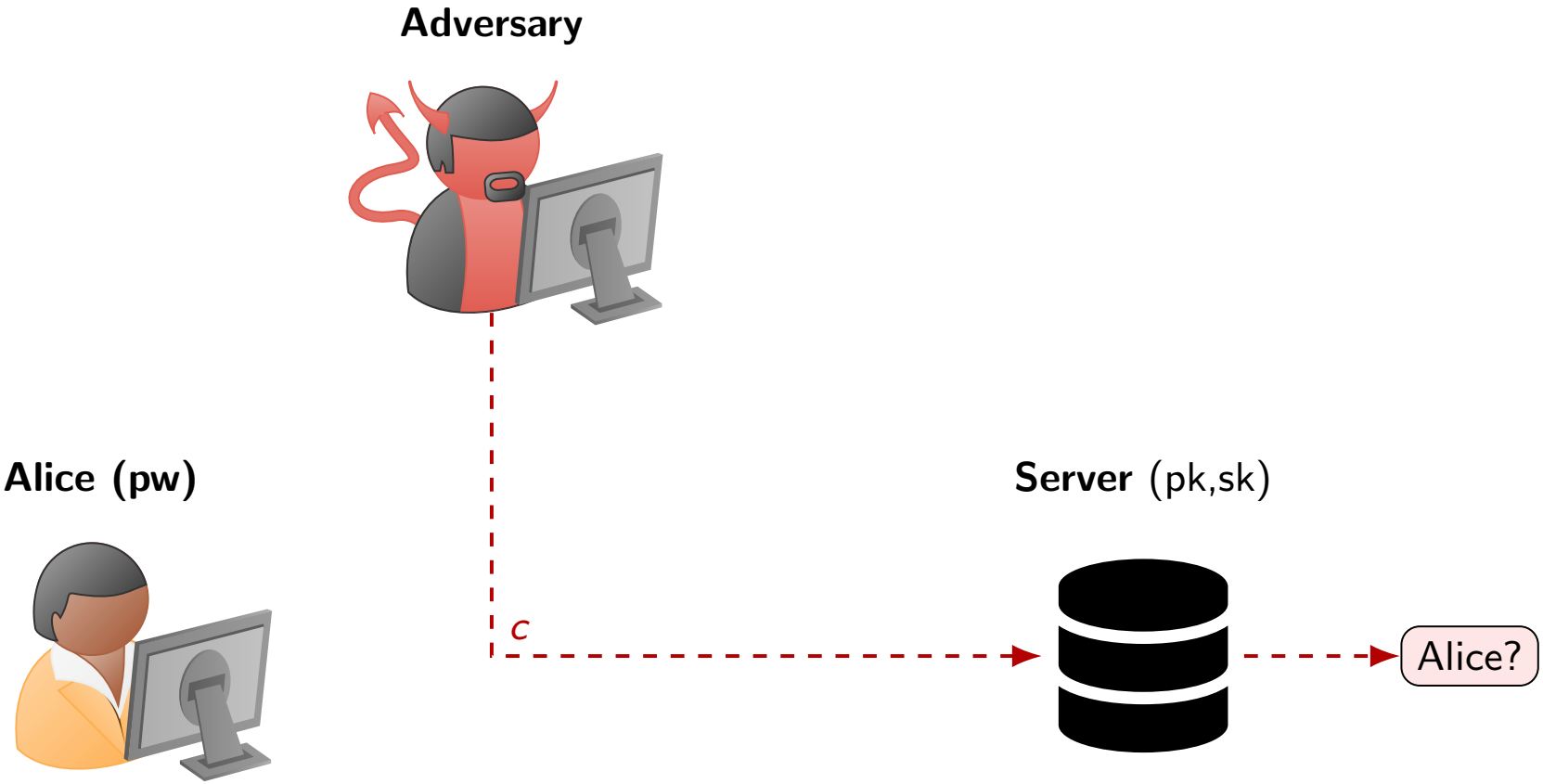
Outsider-CCA



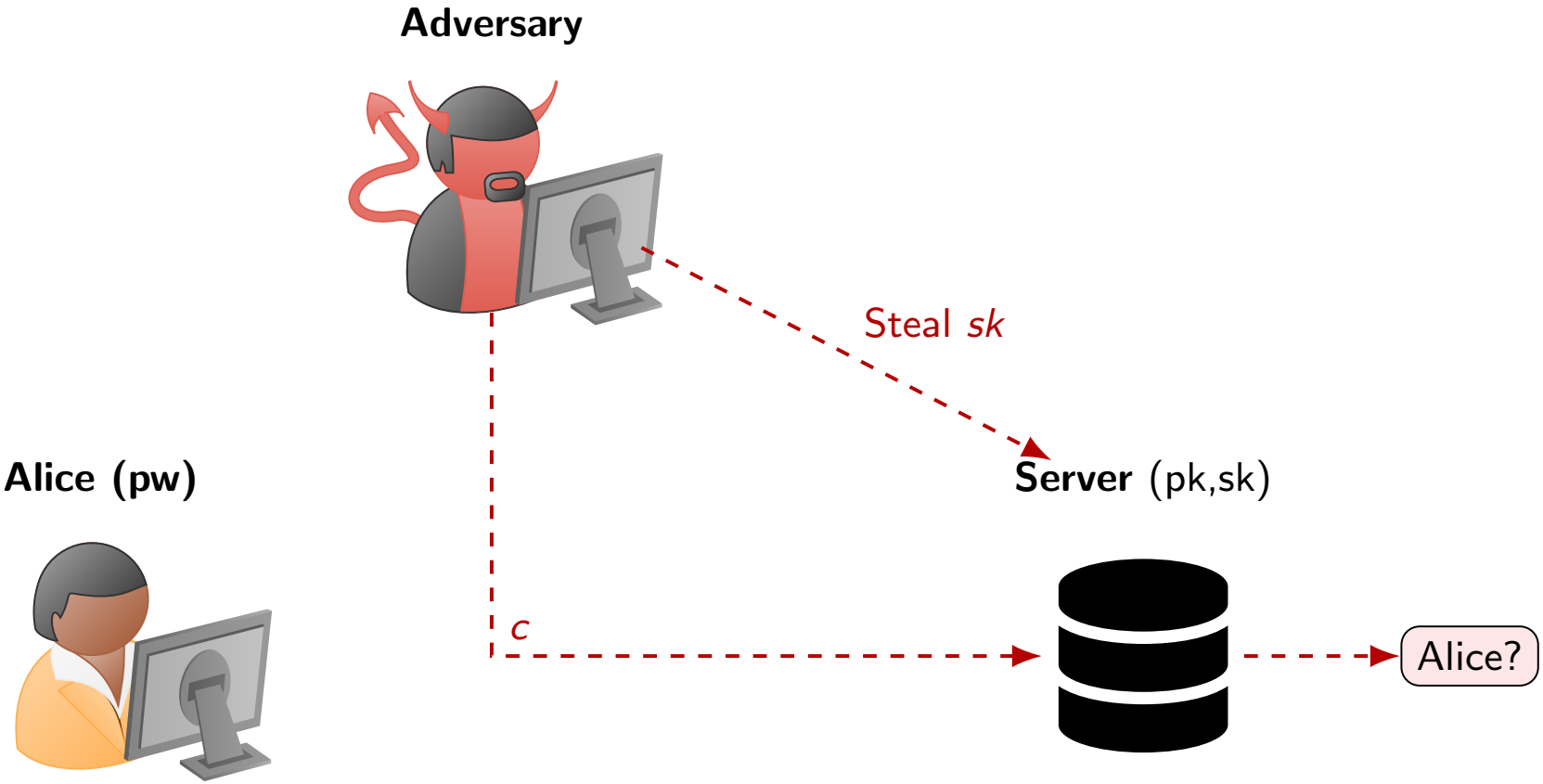
Insider-CCA


Adversary
k?
Steal pw
Intercept c
Alice (pw)
Server (pk,sk)
c
Encaps
Decaps
k
k

Outsider-Auth



Insider-Auth



Generic Construction

$\underline{\mathsf{Gen}(\lambda)}$:
$(pk_1, sk_1) \leftarrow K.KeyGen(1^\lambda)$
Return pk_1, sk_1

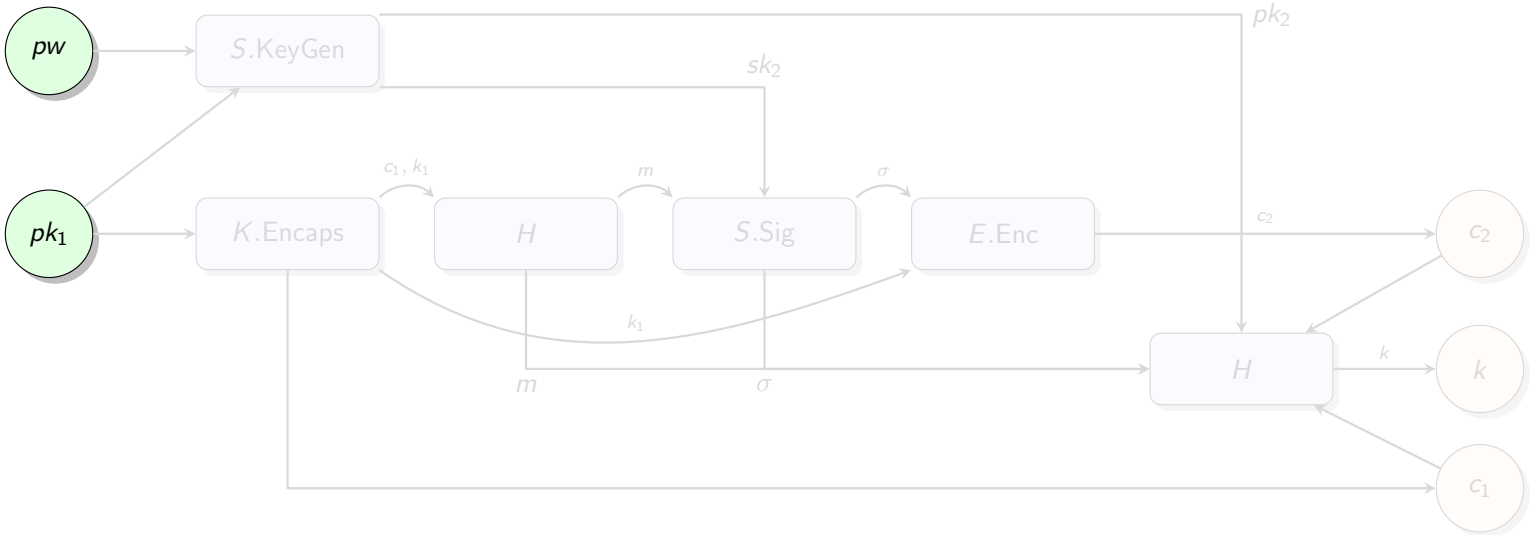
$\underline{\mathsf{Encaps}(pk_1, pw)}$:
$(c_1, k_1) \leftarrow K.Encaps(pk_1)$
$m \leftarrow H(c_1||k_1)$
$(pk_2, sk_2) \leftarrow S.KeyGen(1^\lambda; H(pw||pk_1))$
$\sigma \leftarrow S.Sig(sk_2, m)$
$c_2 \leftarrow E.Enc(k_1, \sigma)$
Return $c = (c_1, c_2), k = H(c_1||c_2||m||\sigma||pk_2)$

$\underline{\mathsf{Enrol}(pk_1, pw)}$:
$r \leftarrow H(pw||pk_1)$
$(pk_2, sk_2) \leftarrow S.KeyGen(1^\lambda; r)$
Return $d = pk_2$

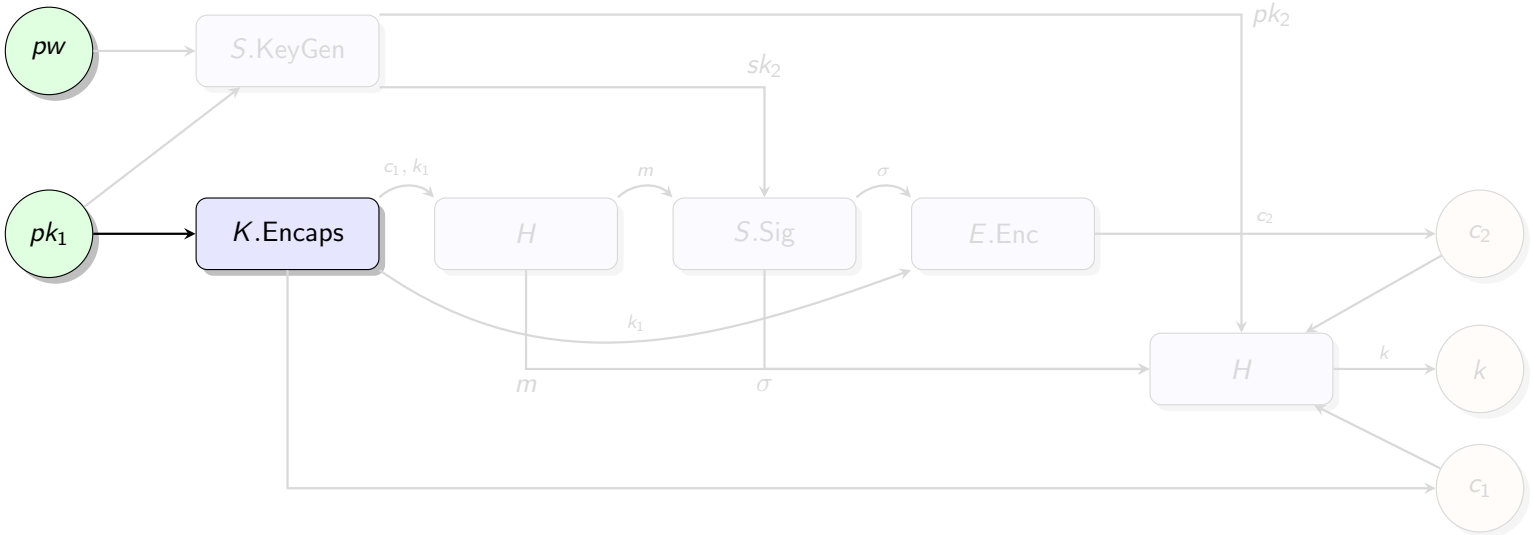
$\underline{\mathsf{Decaps}(sk_1, pk_2, c = (c_1, c_2))}$:
$k_1 \leftarrow K.Decaps(sk_1, c_1)$
$m \leftarrow H(c_1||k_1)$
$\sigma \leftarrow E.Dec(k_1, c_2)$
If $S.Vrfy(pk_2, m, \sigma)$:
  Return $k = H(c_1||c_2||m||\sigma||pk_2)$
Else: **Return** $\perp$

Figure: PAKEM from KEM K, Signature S, DEM E and Hash function H.

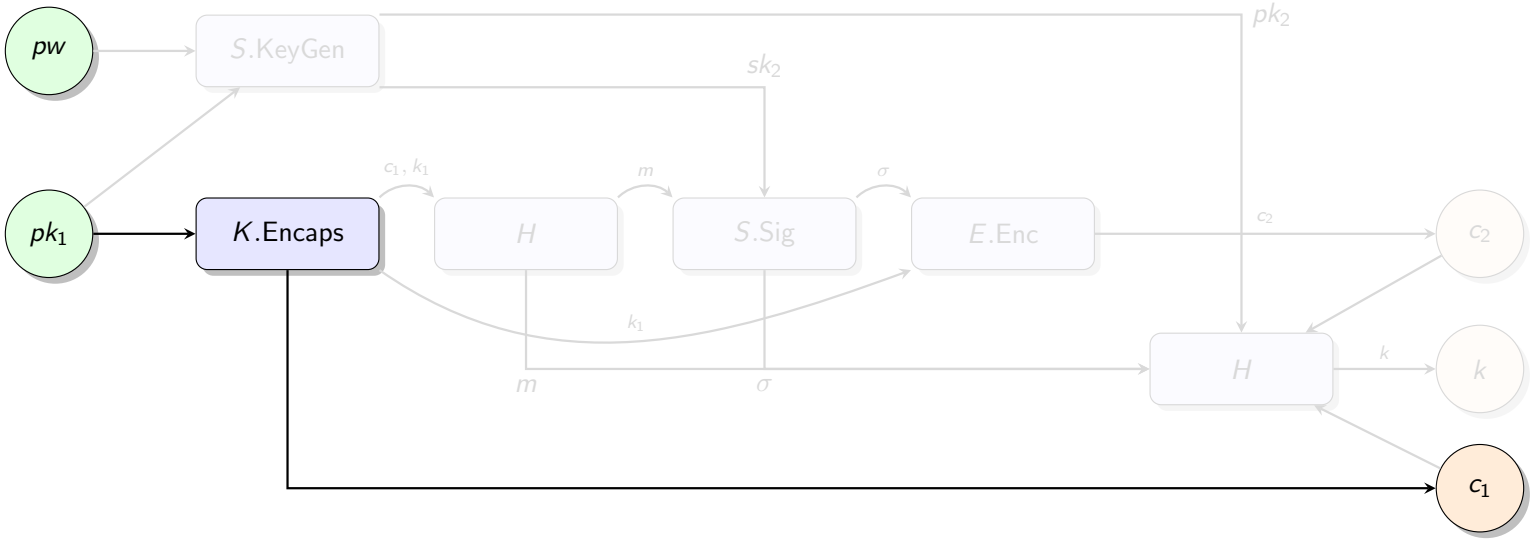
Generic Construction



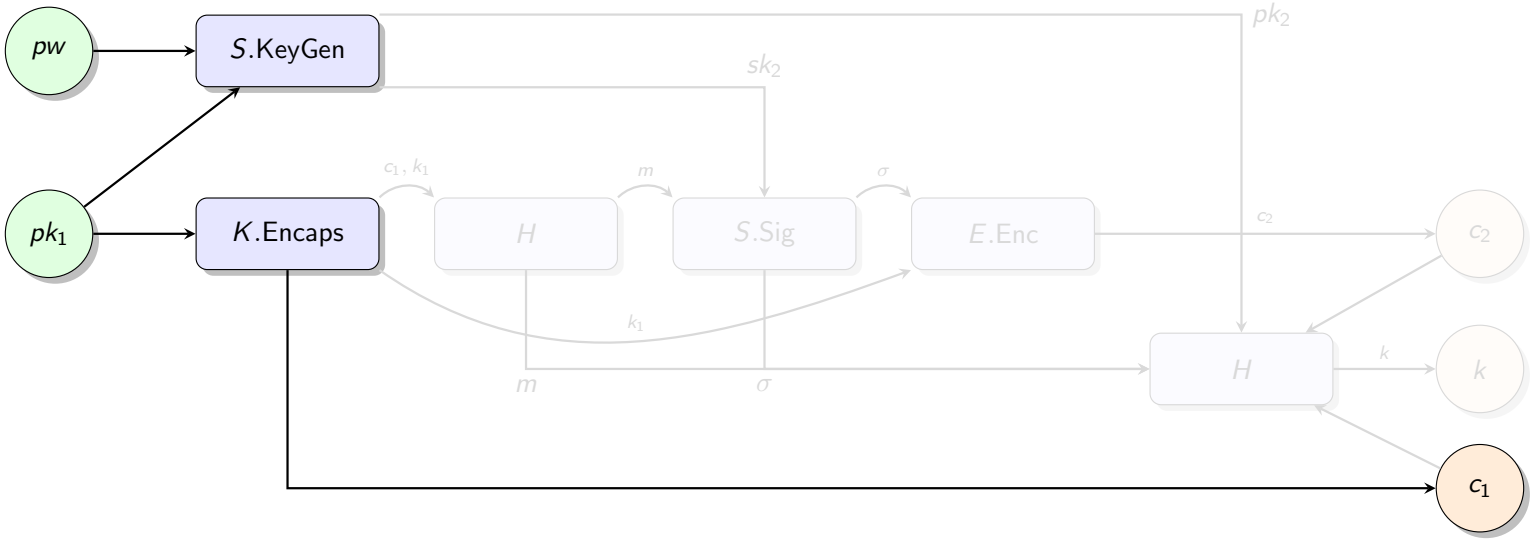
Generic Construction



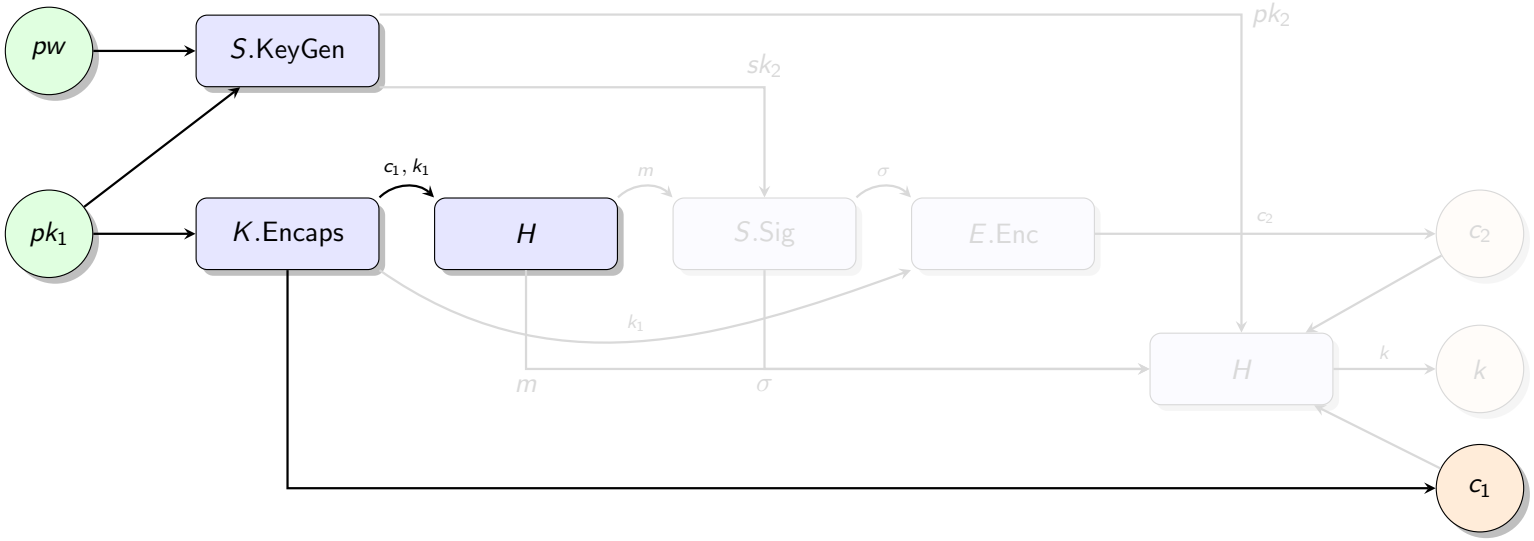
Generic Construction



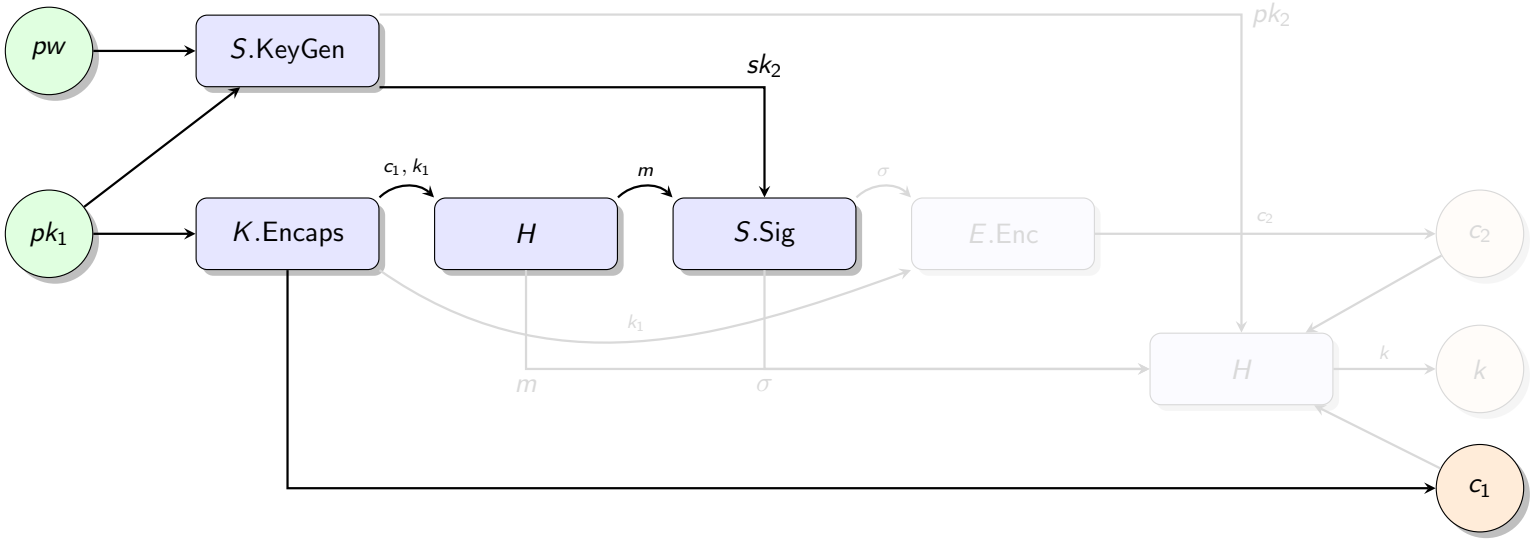
Generic Construction



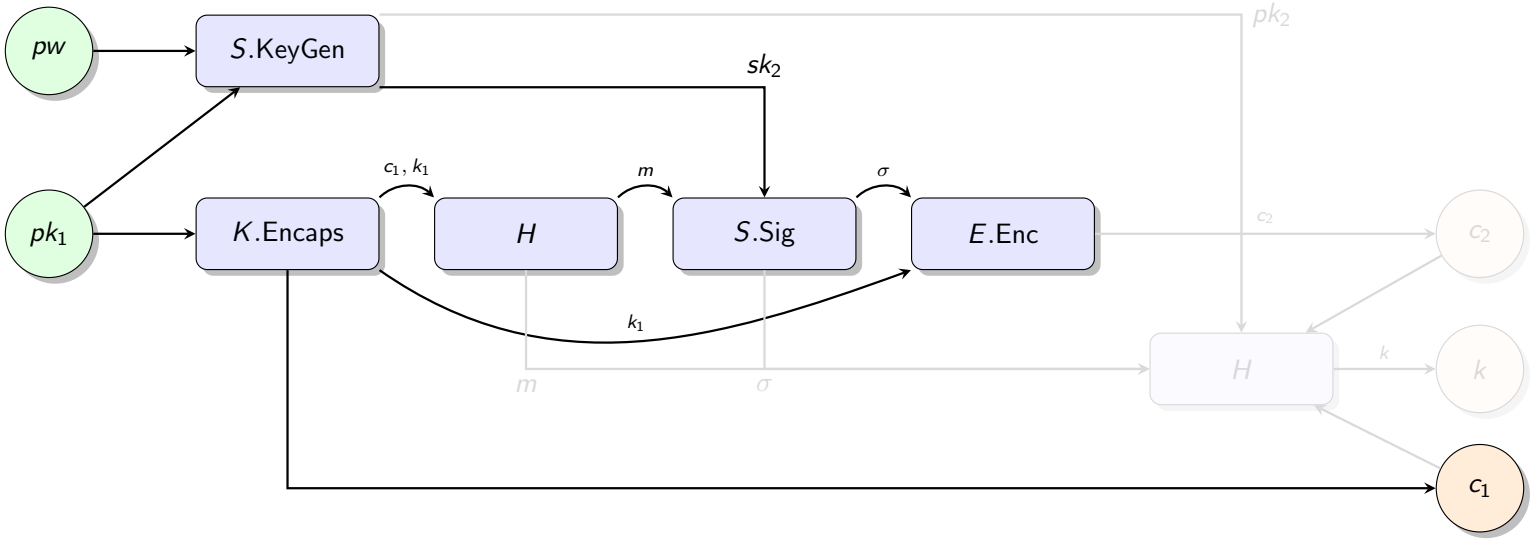
Generic Construction



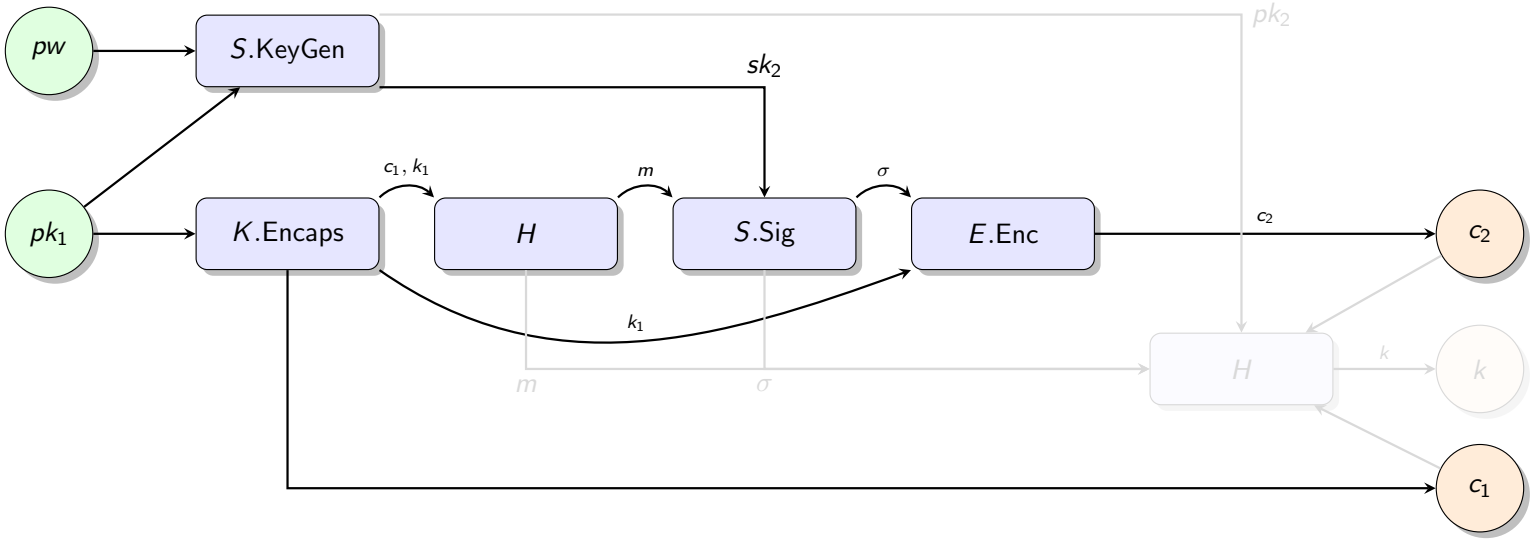
Generic Construction



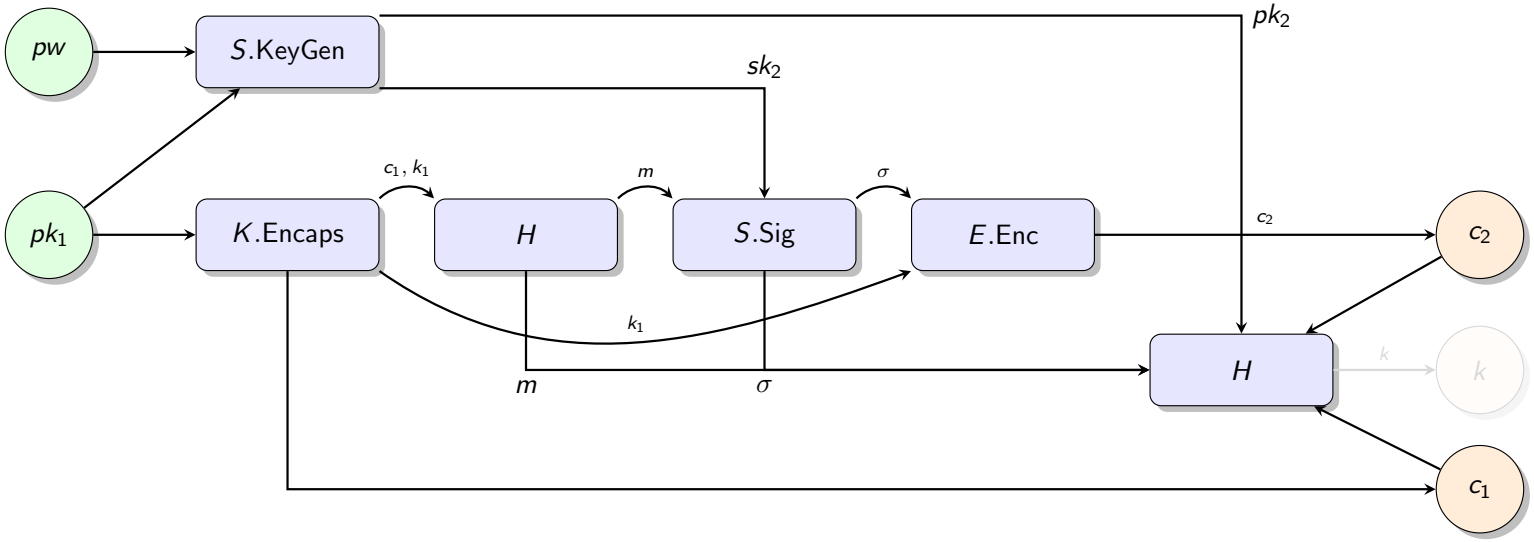
Generic Construction



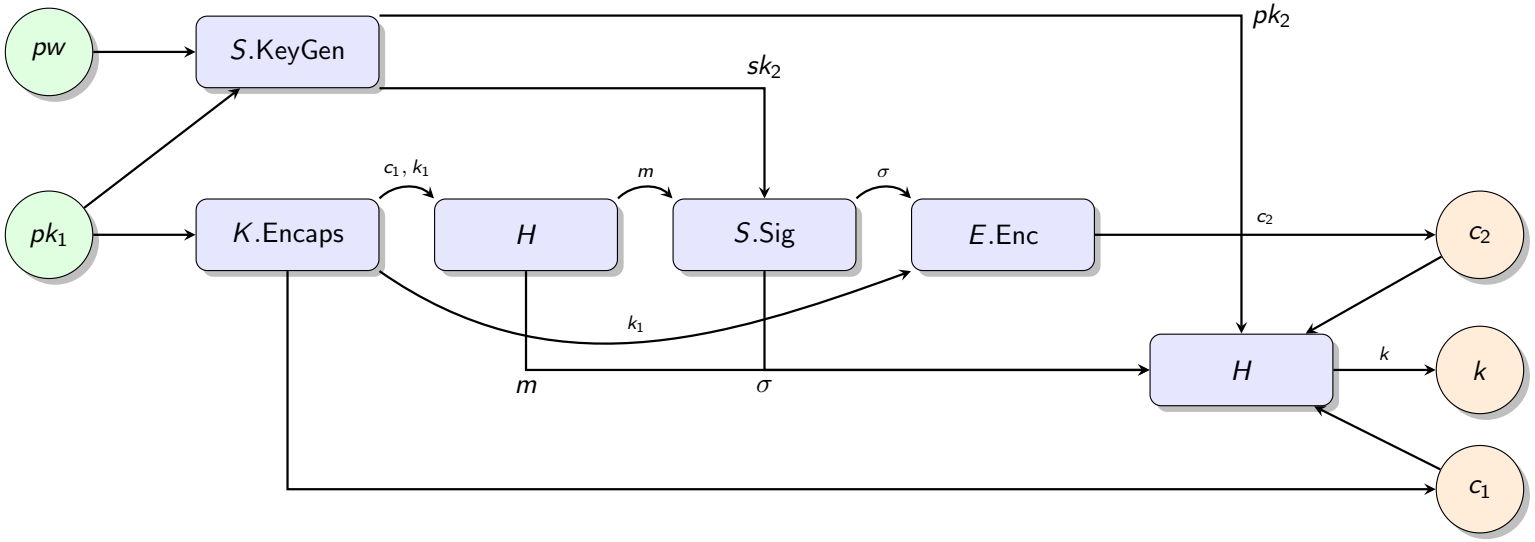
Generic Construction



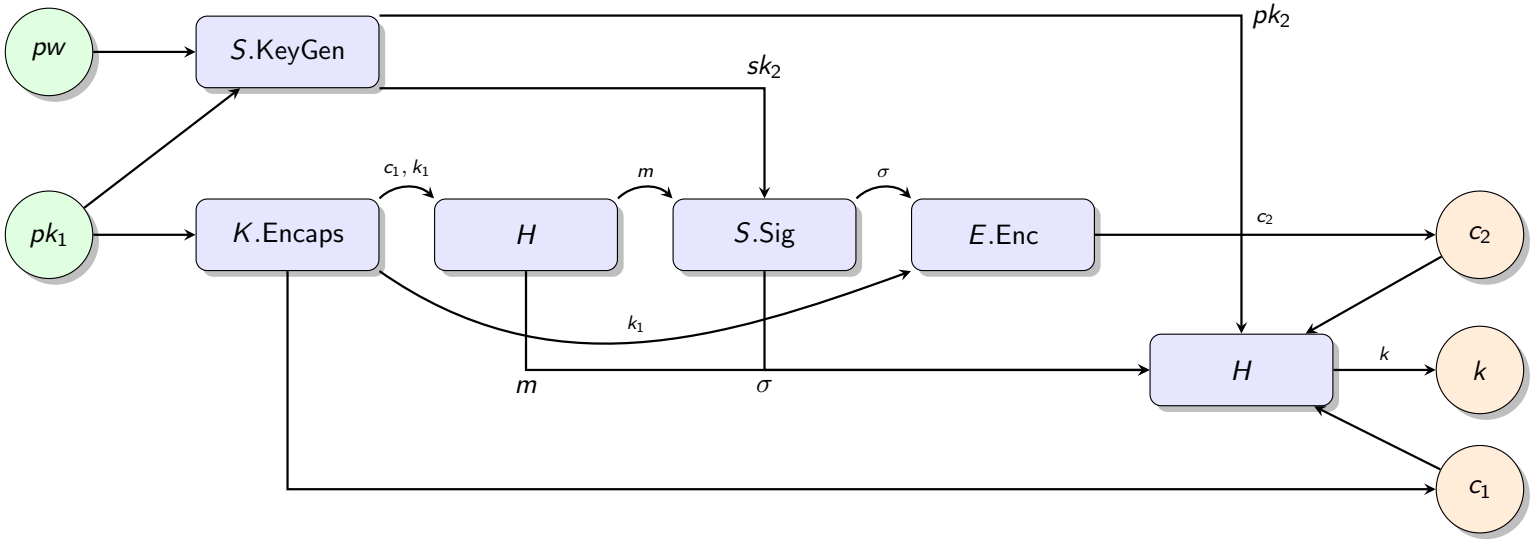
Generic Construction

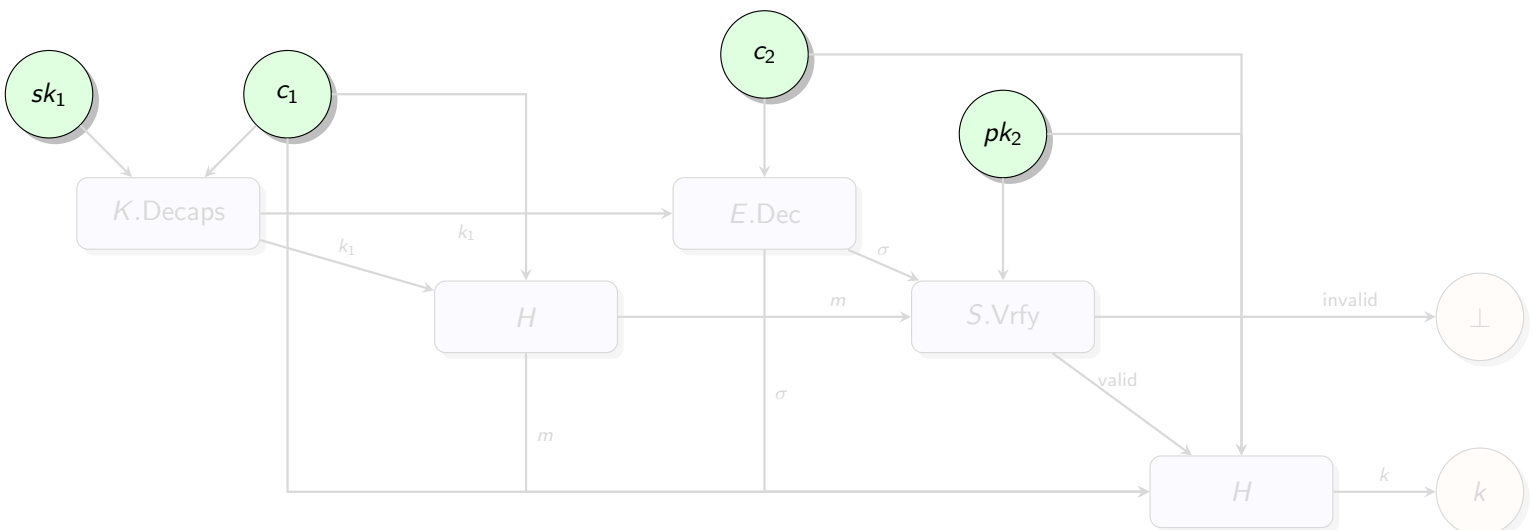


Generic Construction

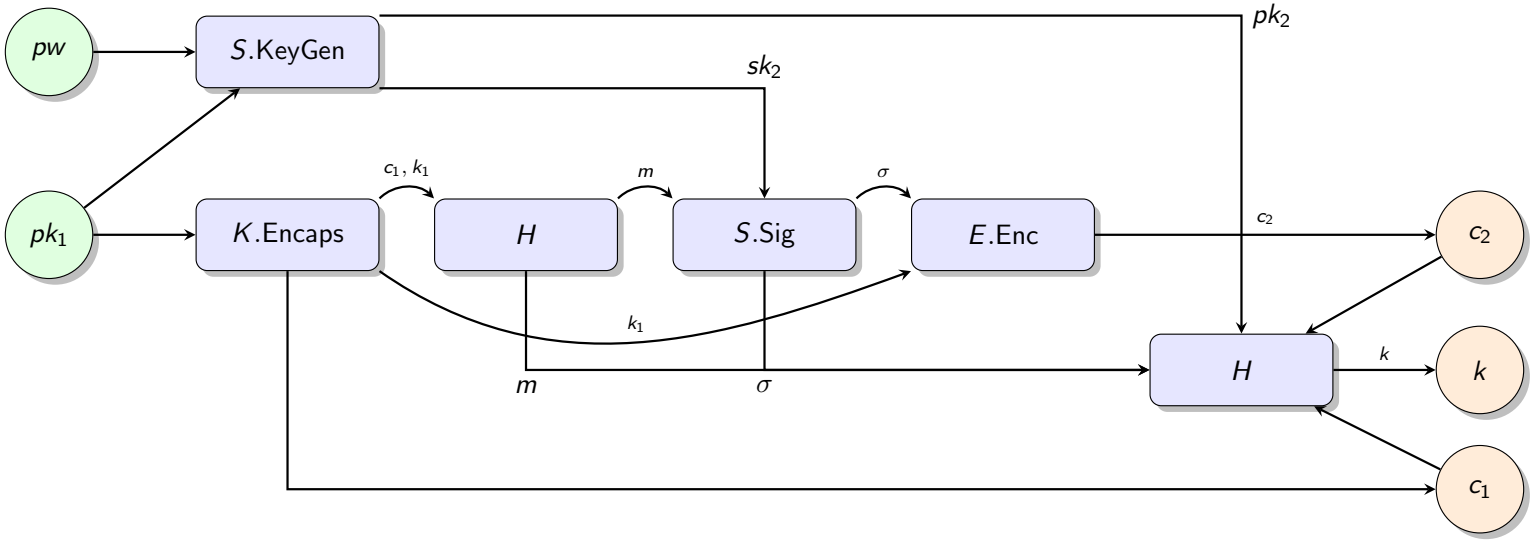


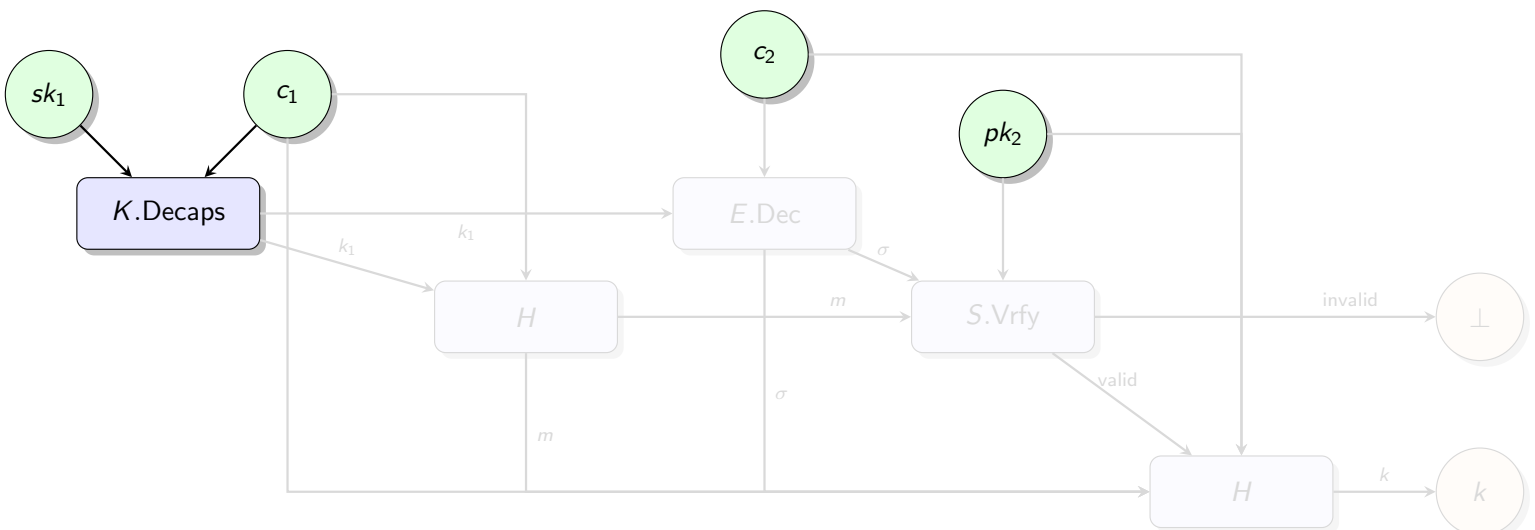
Generic Construction



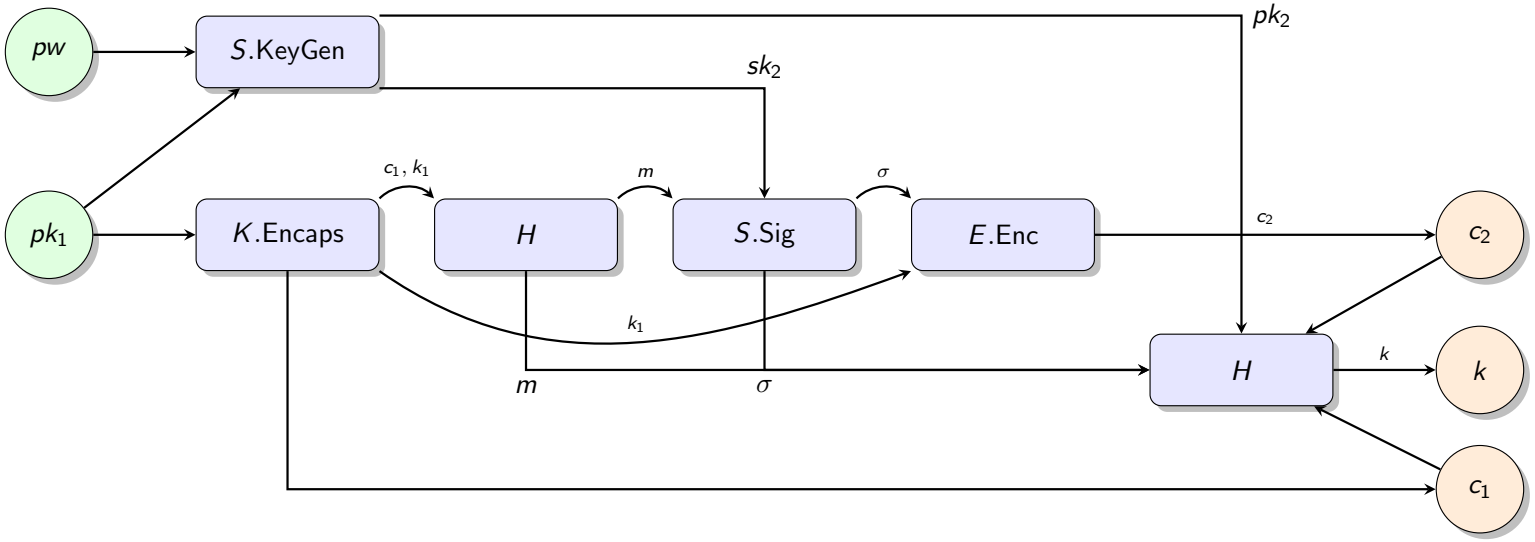


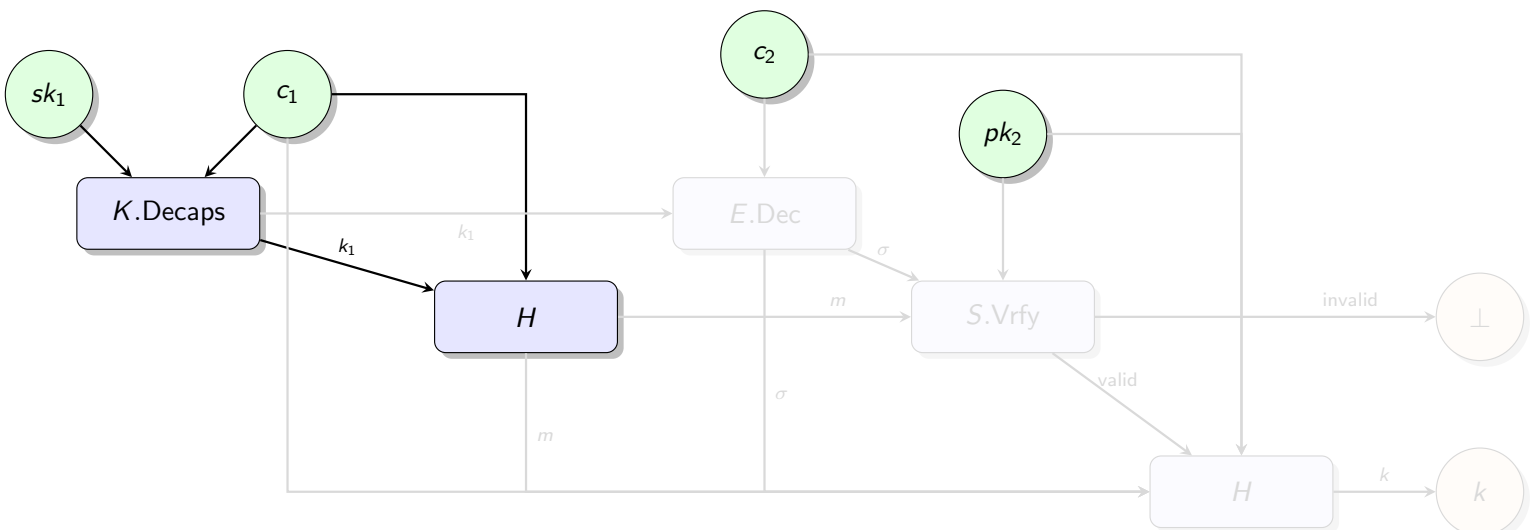
Generic Construction



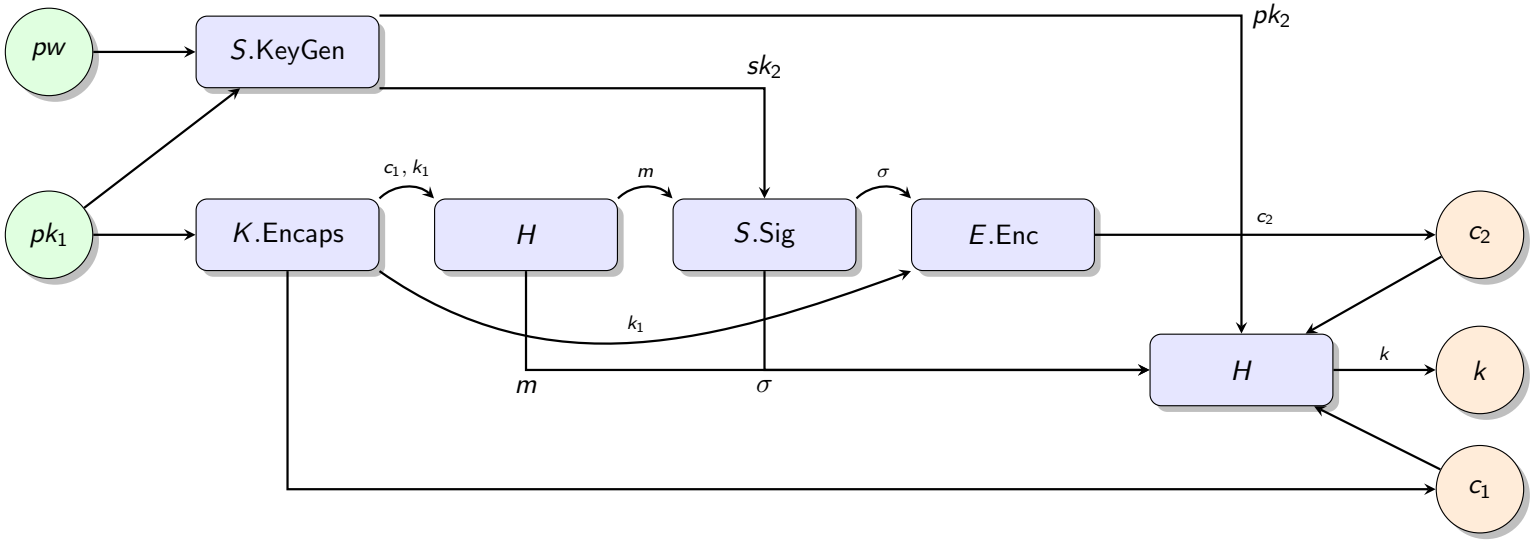


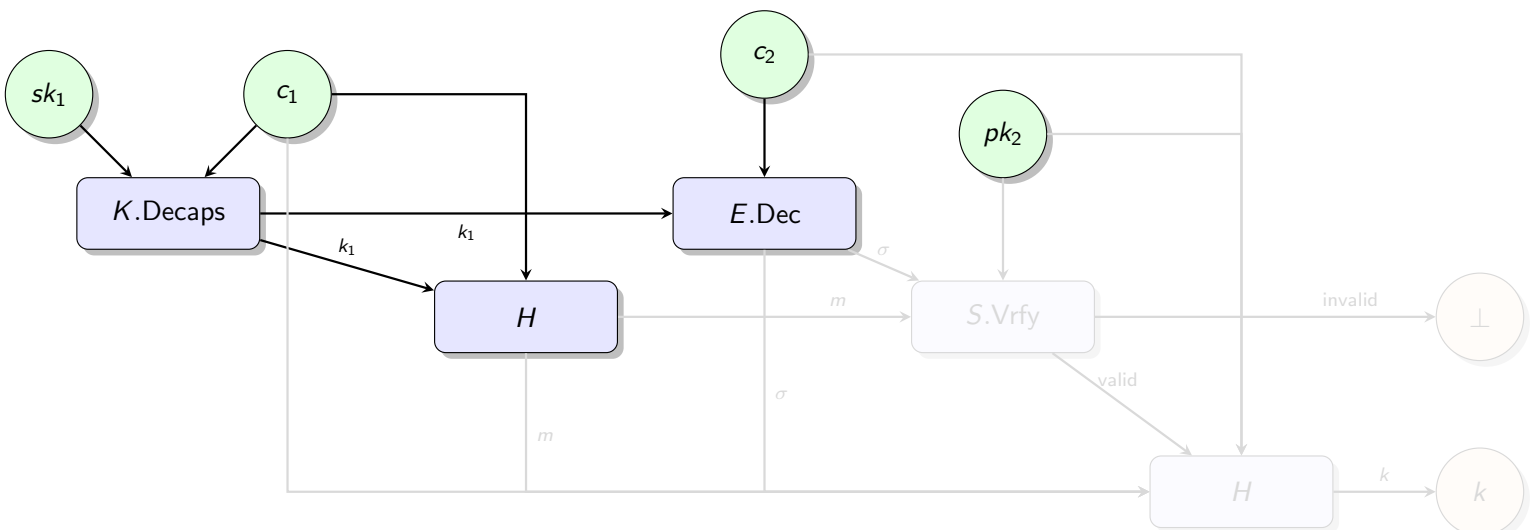
Generic Construction



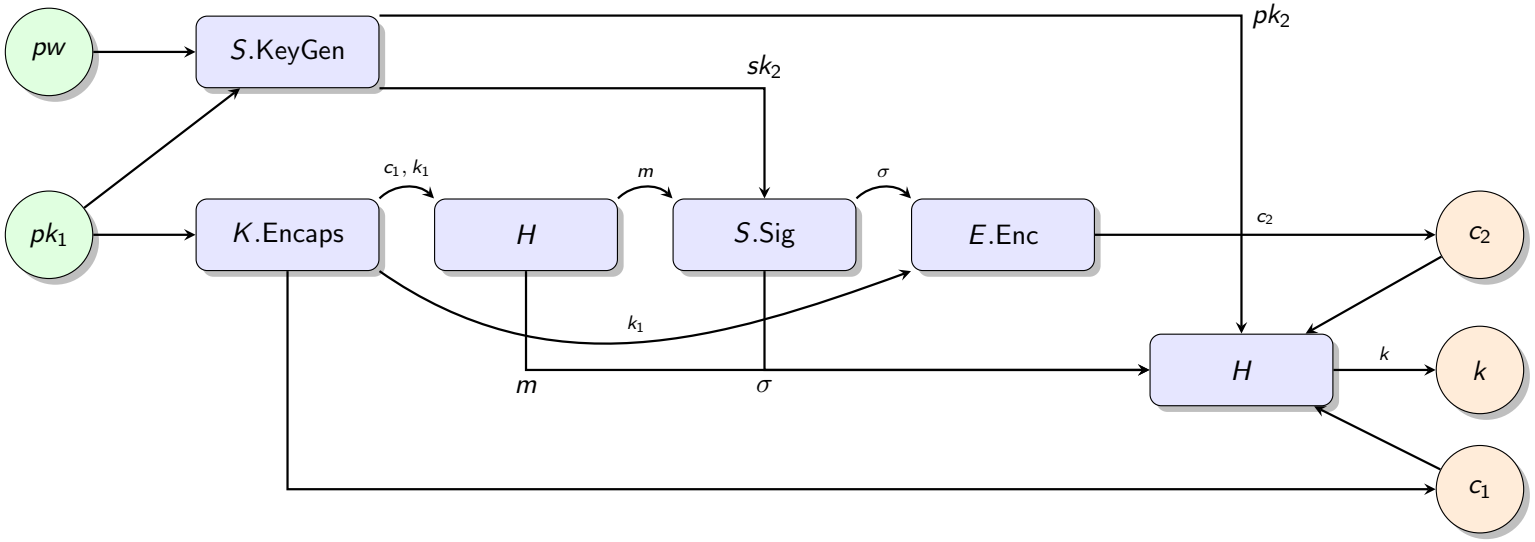


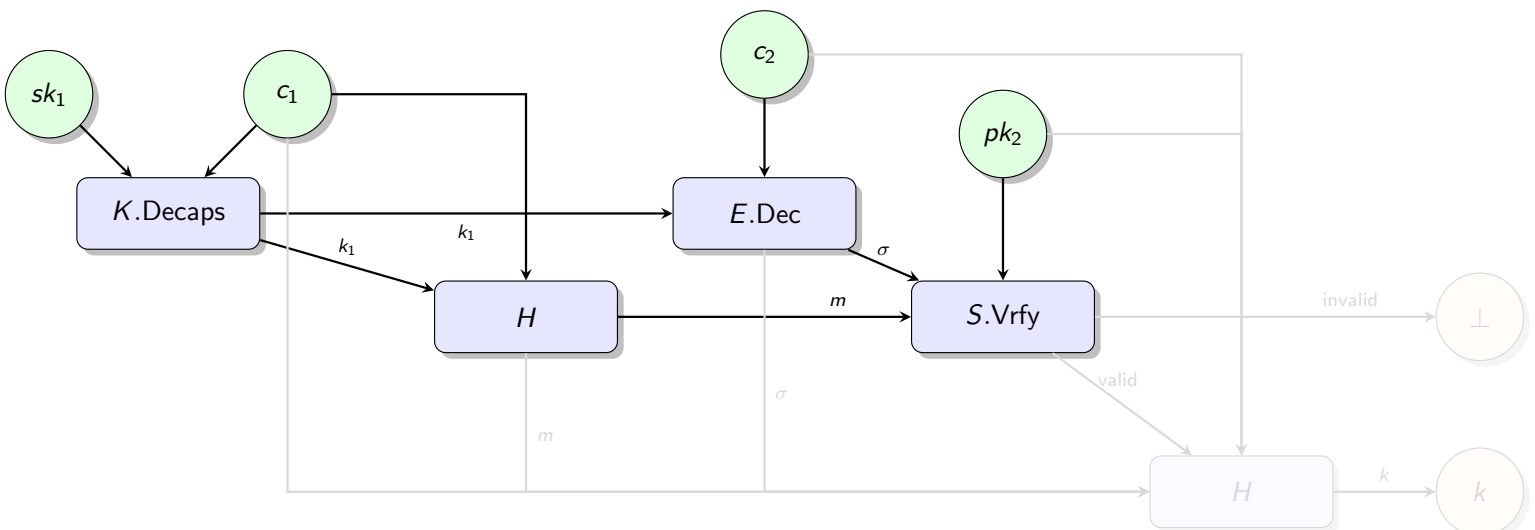
Generic Construction



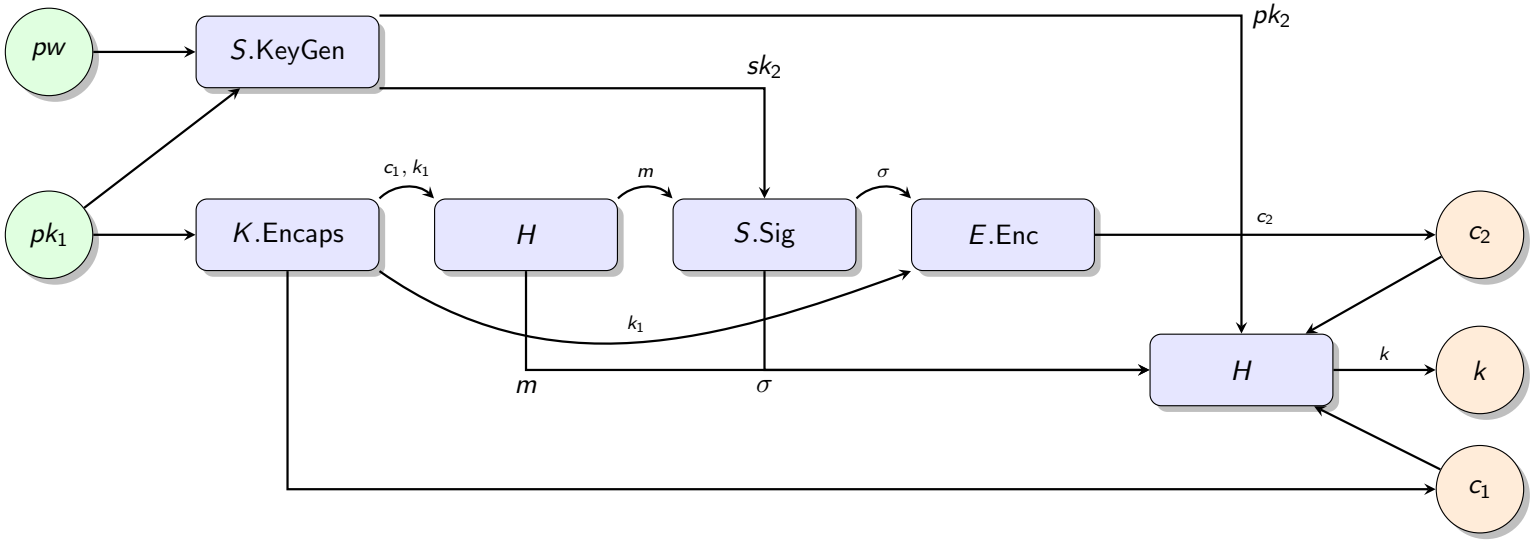


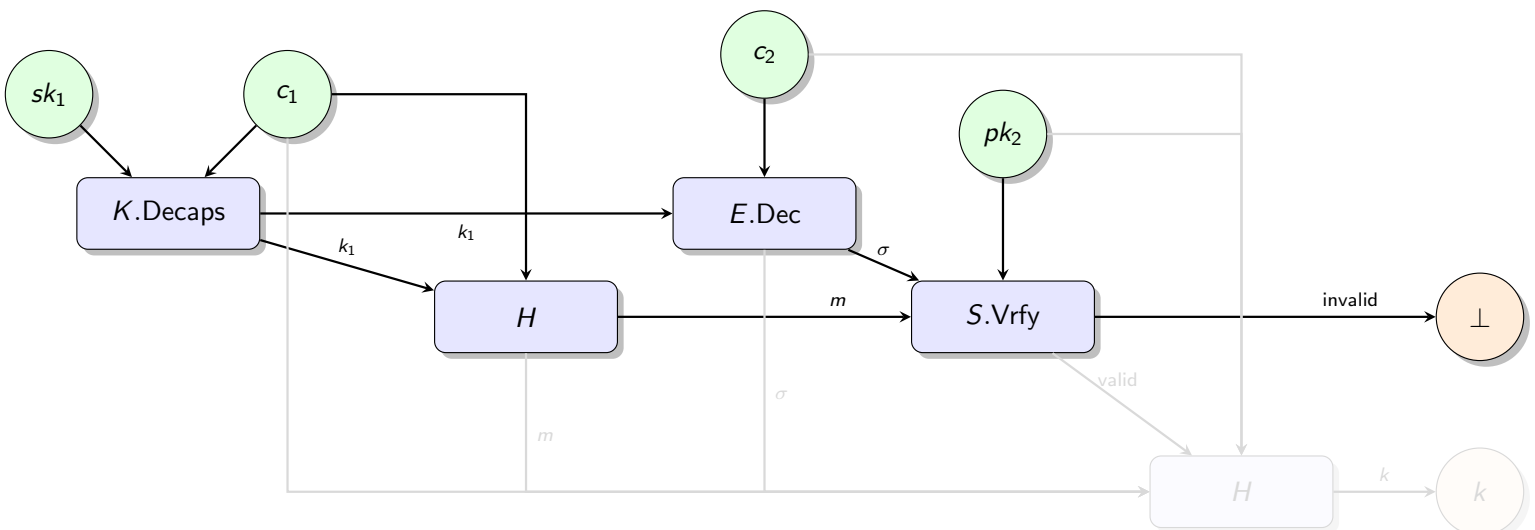
Generic Construction



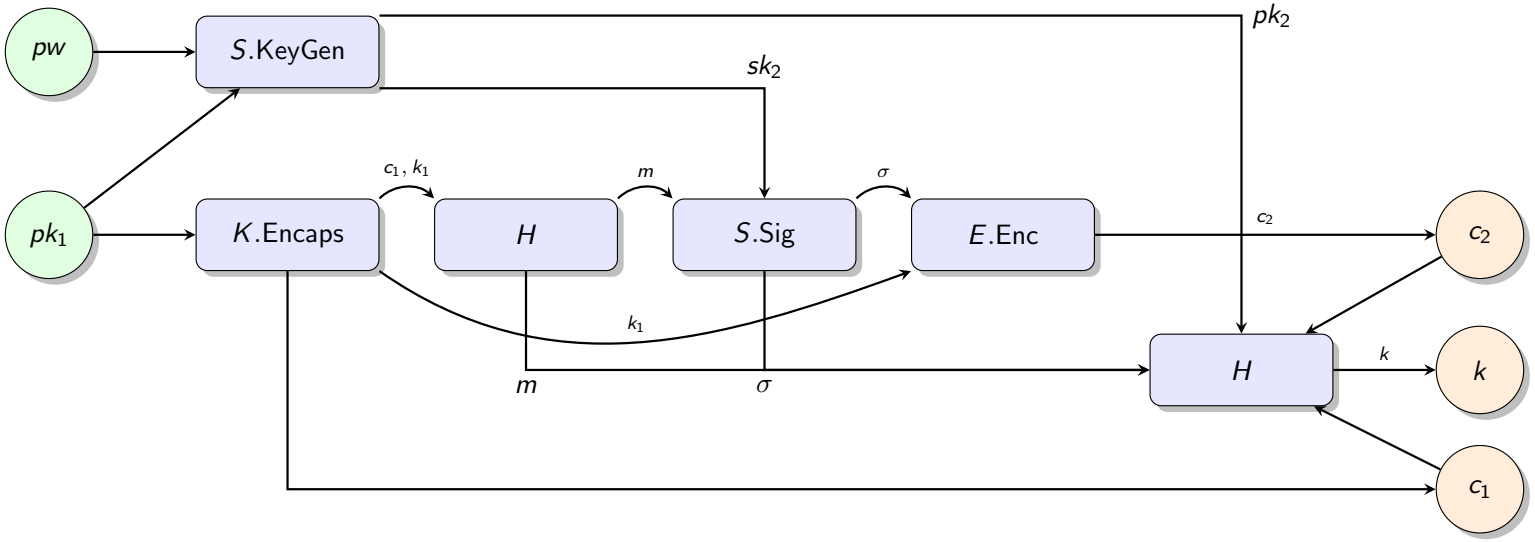


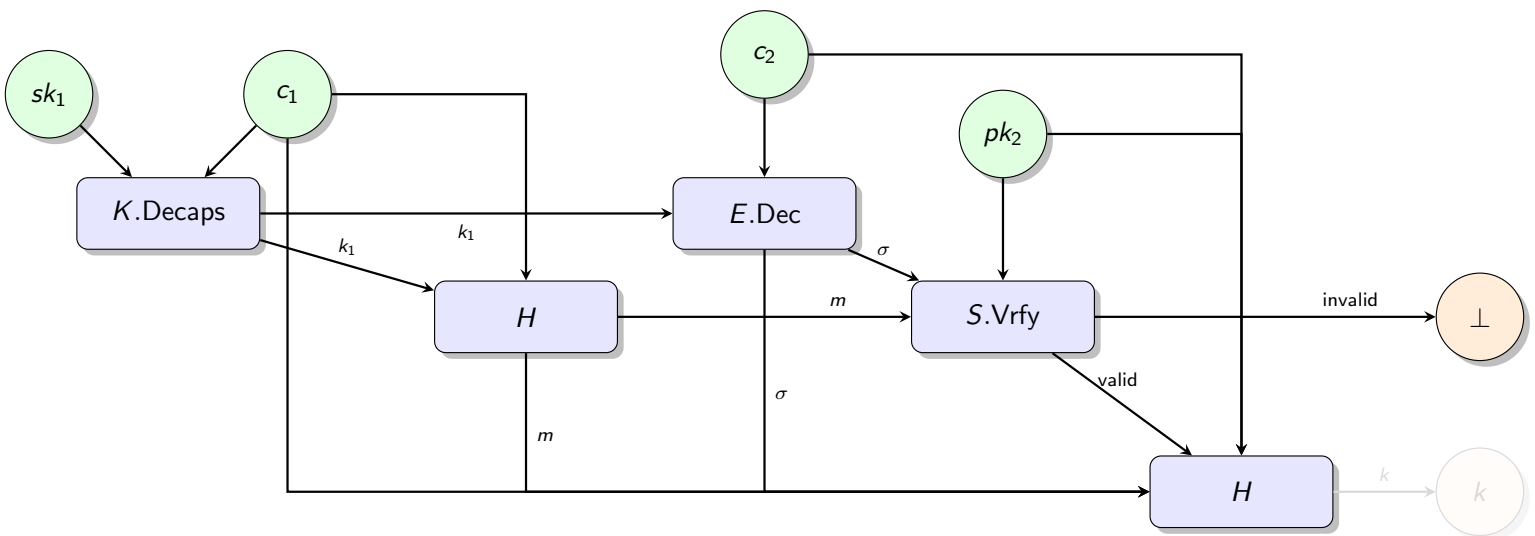
Generic Construction



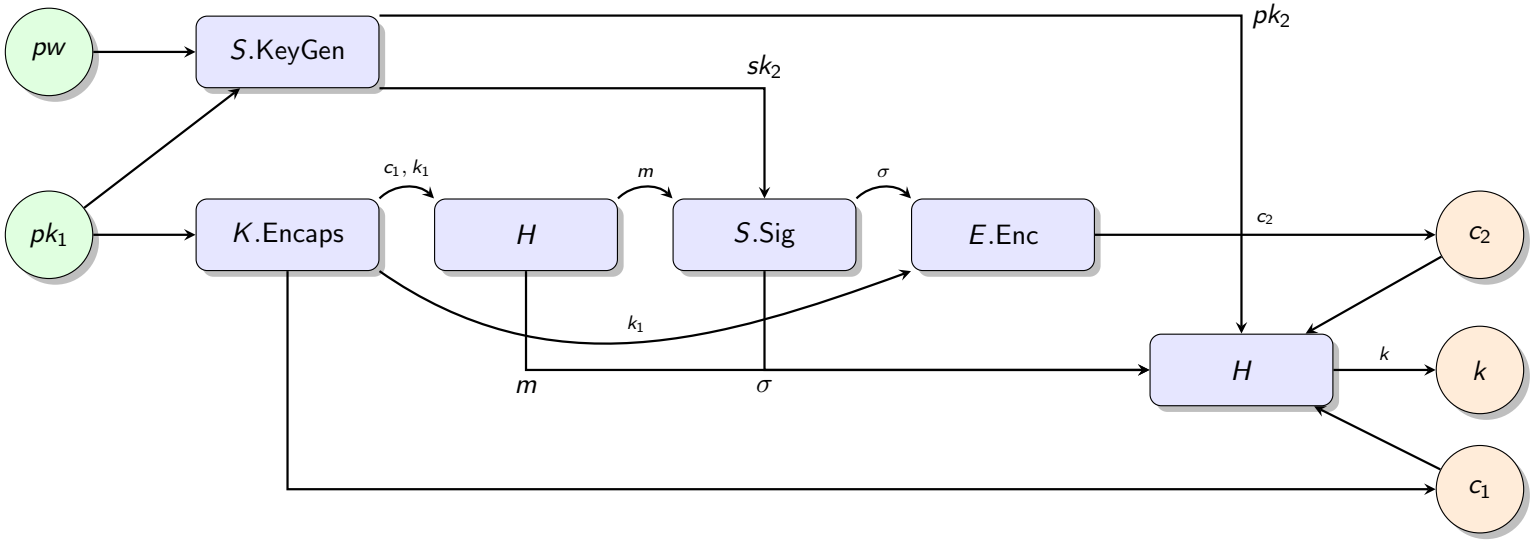


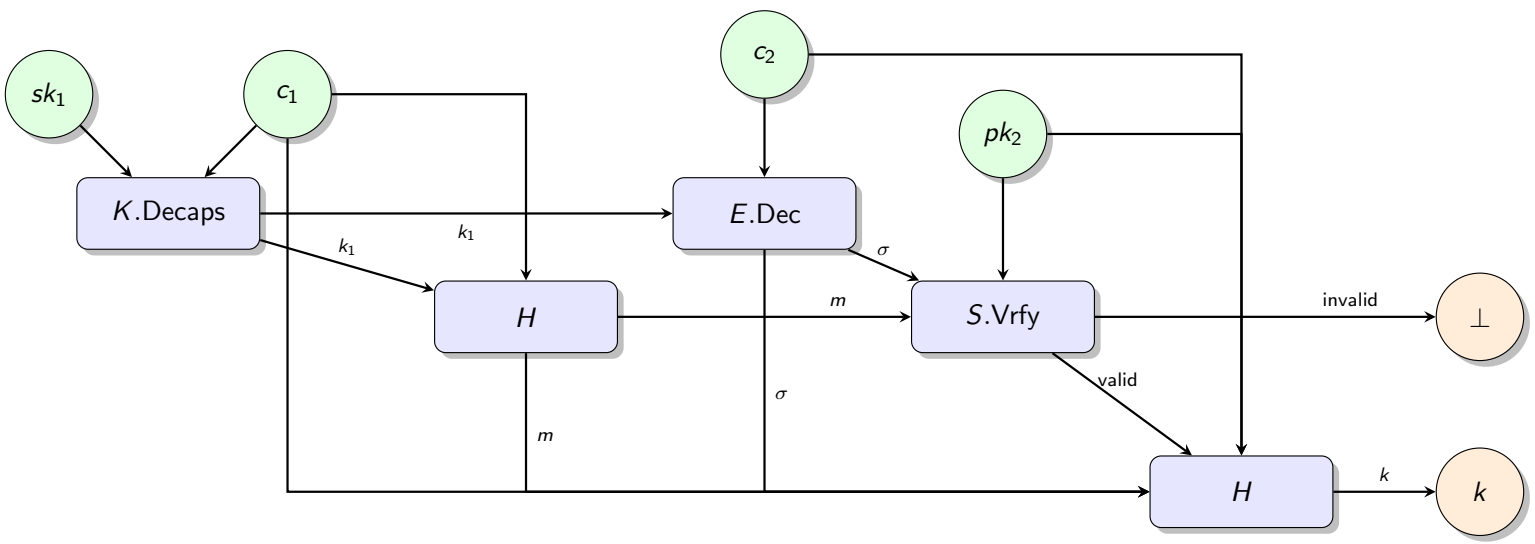
Generic Construction





Generic Construction





Proved Security

KEM	Signature	DEM	Hash	PAKEM
IND-CCA	-	IND-CCA	RO	Insider-CCA
-	EUF-CMA	-	RO	Insider-Auth

Table: Security assumptions of the construction and corresponding security.

PAKEM Property	Reduction
Insider-CCA	$\mathrm{Adv}^{(q_d,q_c)\text{-Insider-CCA}}_{\mathcal{A},\mathrm{PAKEM}} \leq \mathrm{Adv}^{\text{IND-CCA}}_{\mathcal{B},K} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathcal{C},E} + \mathsf{negl}(\lambda)$
Insider-Auth	$\mathrm{Adv}^{(q_d,q_c)\text{-Insider-Auth}}_{\mathcal{A},\mathrm{PAKEM}} \leq \mathrm{Adv}^{\text{EUF}-\text{CMA}}_{\mathcal{A},S} + \mathsf{negl}(\lambda)$

Table: Reduction bound for PAKEM corresponding security property.

Building Blocks

Our instantiation uses the following four algorithms:

- **CRYSTALS-Kyber**: IND-CCA-secure KEM
- **CRYSTALS-Dilithium**: EUF-CMA-secure signature scheme
- **AES-256 GCM**: IND-CCA-secure DEM
- **SHA3-512**: Hash function modeled as a random oracle

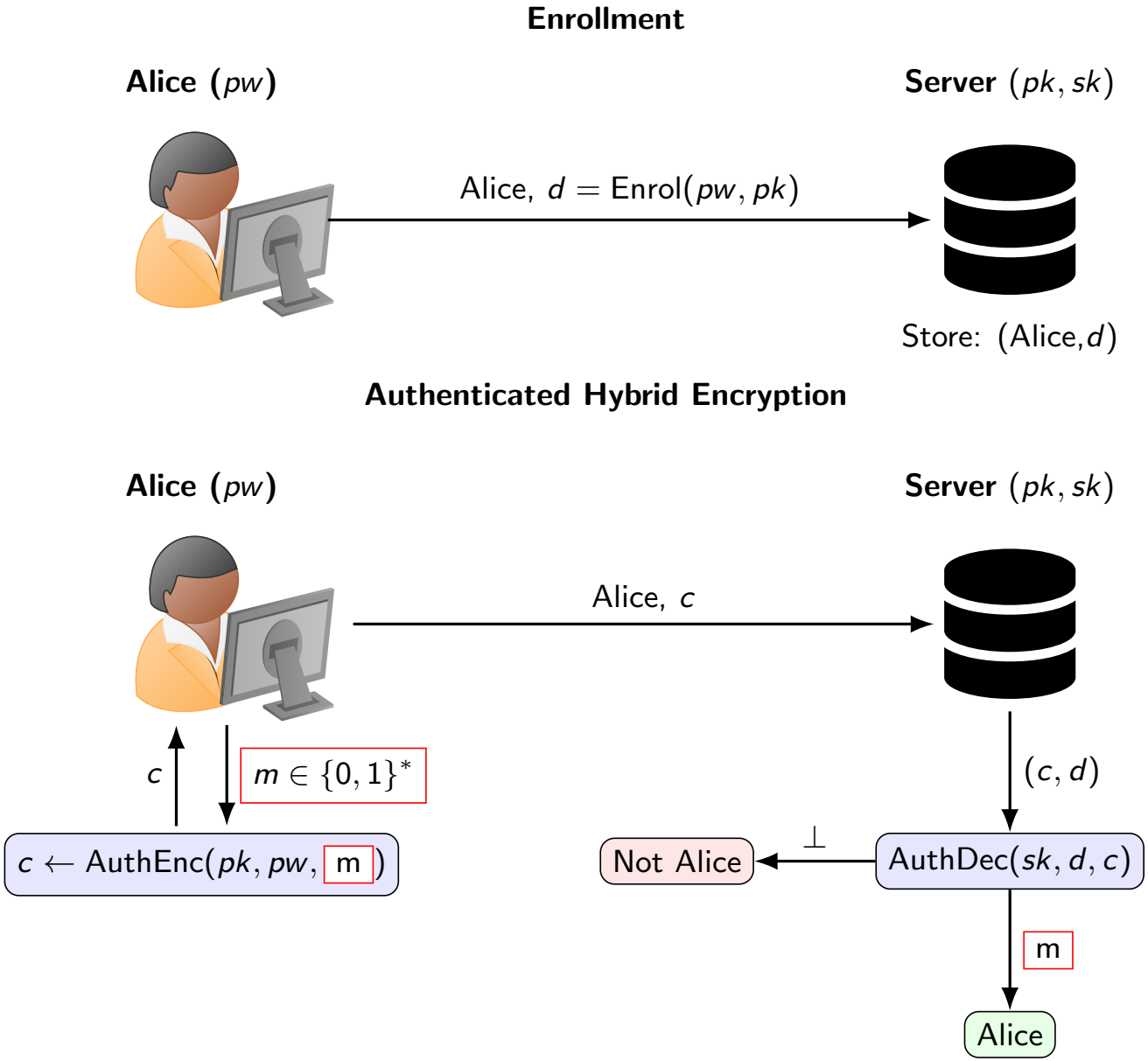
Performance

NIST Level	Scheme	Bandwidth (bytes)	Storage (bytes)	Runtime (μs)		
				Setup	Encaps/Sign	Decaps/Verify
Level 1	PAKEM	3,228	3,744	50.84	126.44	70.65
	Kyber-512	780	2,432	12.81	16.21	11.39
	Dilithium2	2,432	1,312	28.92	79.34	29.04
Level 3	PAKEM	4,422	5,536	80.31	193.02	105.20
	Kyber-768	1,100	3,584	17.48	21.85	16.01
	Dilithium3	3,305	1,952	48.81	128.93	47.73
Level 5	PAKEM	6,204	7,328	119.51	244.42	155.30
	Kyber-1024	1,580	4,736	23.99	30.06	22.52
	Dilithium5	4,607	2,592	76.31	156.66	75.29

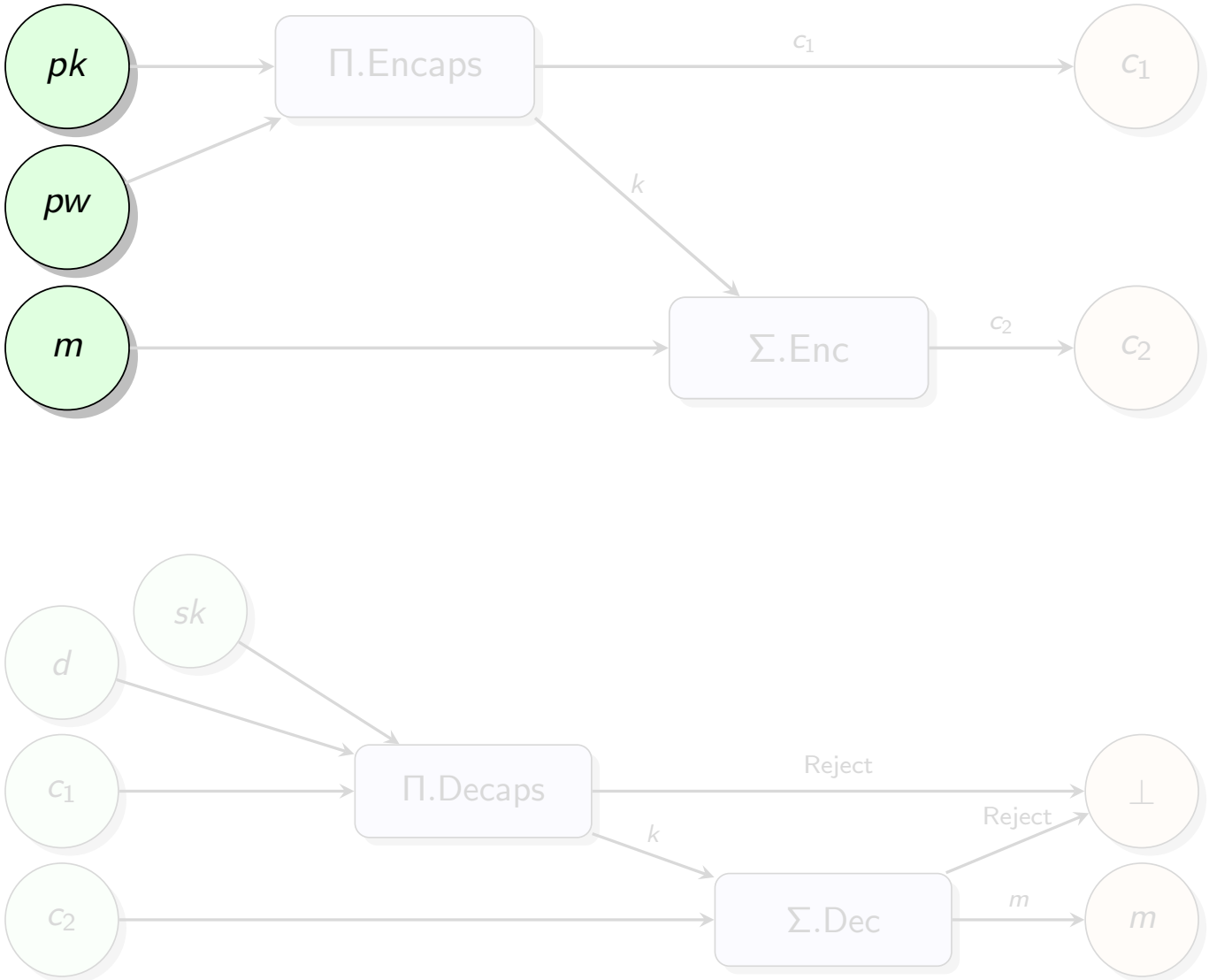
Table: Bandwidth, storage, and runtime (mean over 10,000 iterations) comparison between PAKEM, Kyber and Dilithium.

Table of Contents

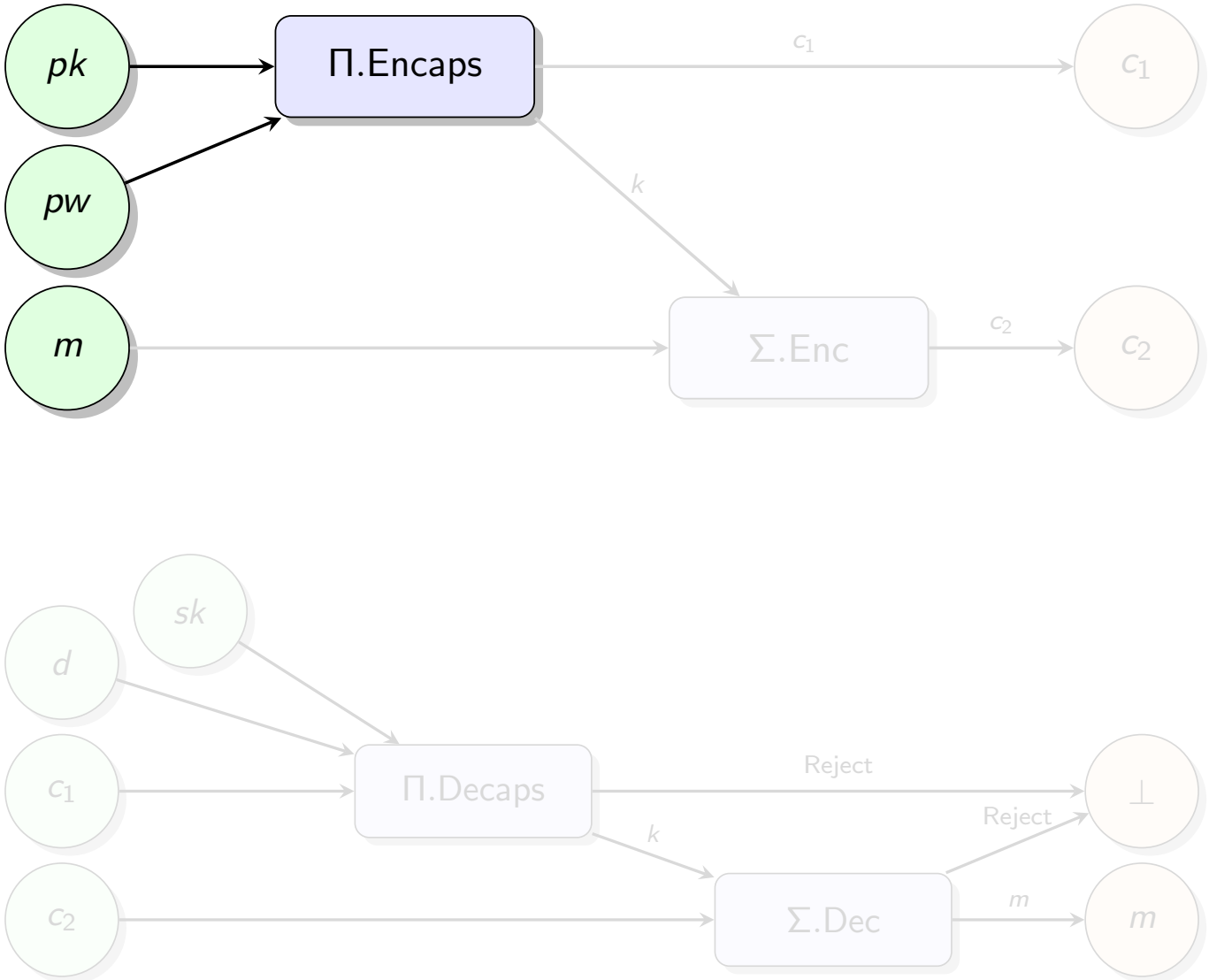
PAHE Overview



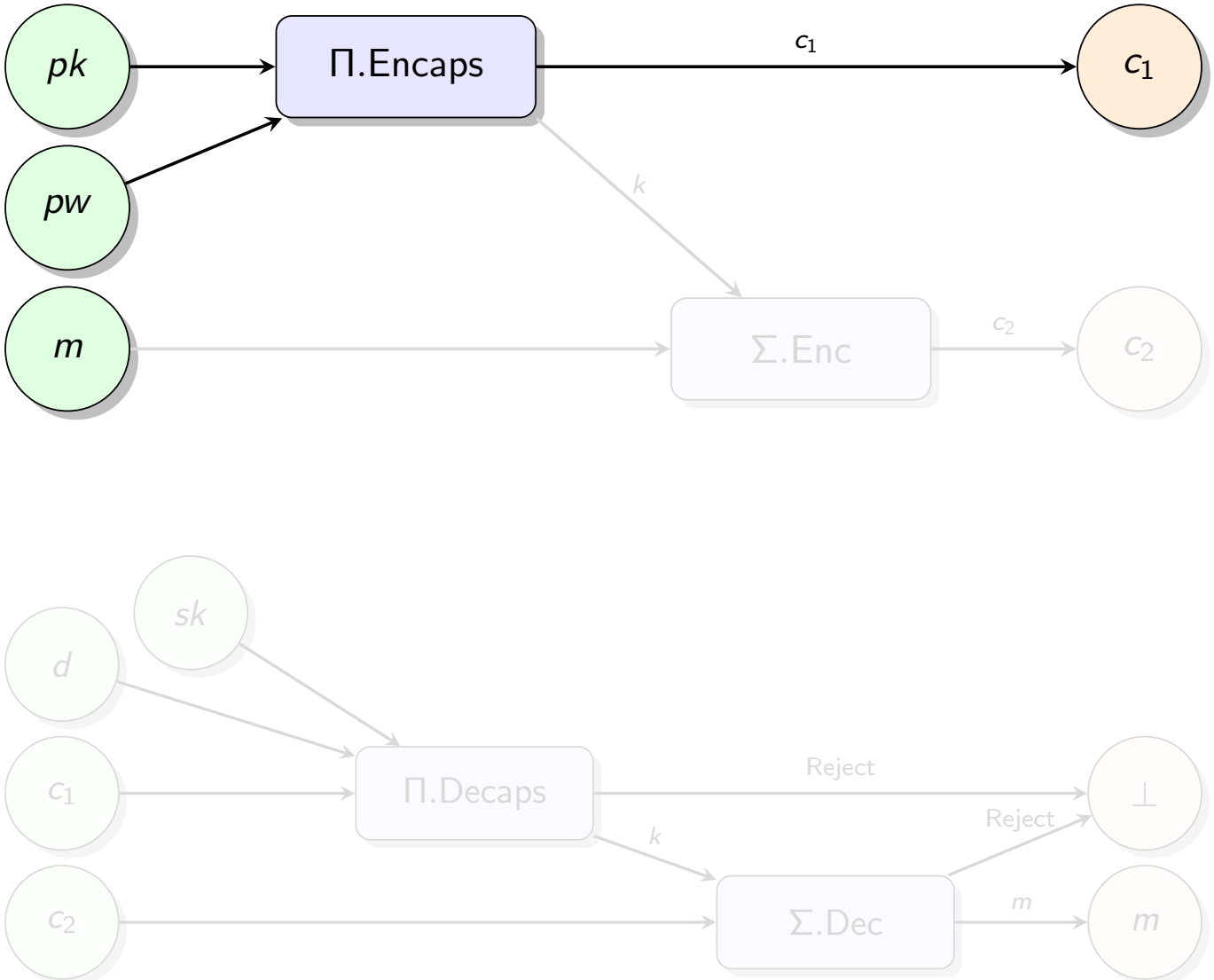
Generic Construction From PAKEM and DEM



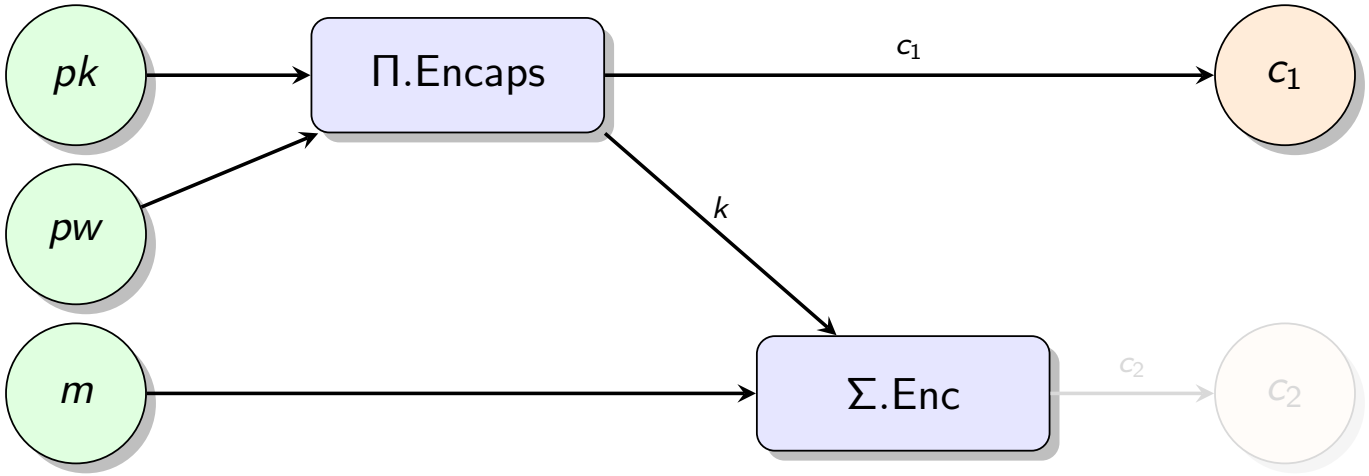
Generic Construction From PAKEM and DEM



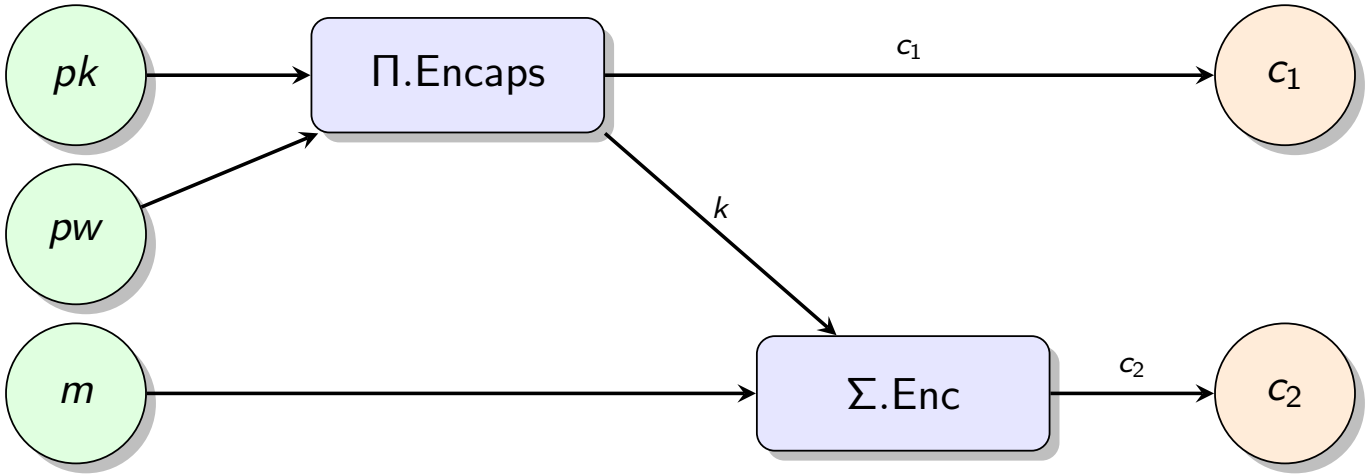
Generic Construction From PAKEM and DEM



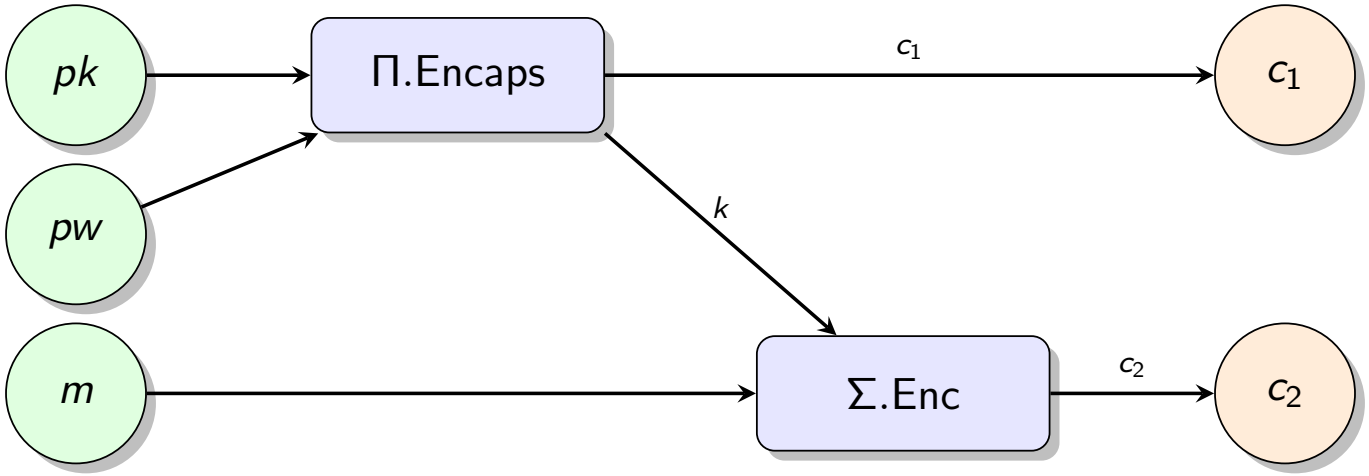
Generic Construction From PAKEM and DEM

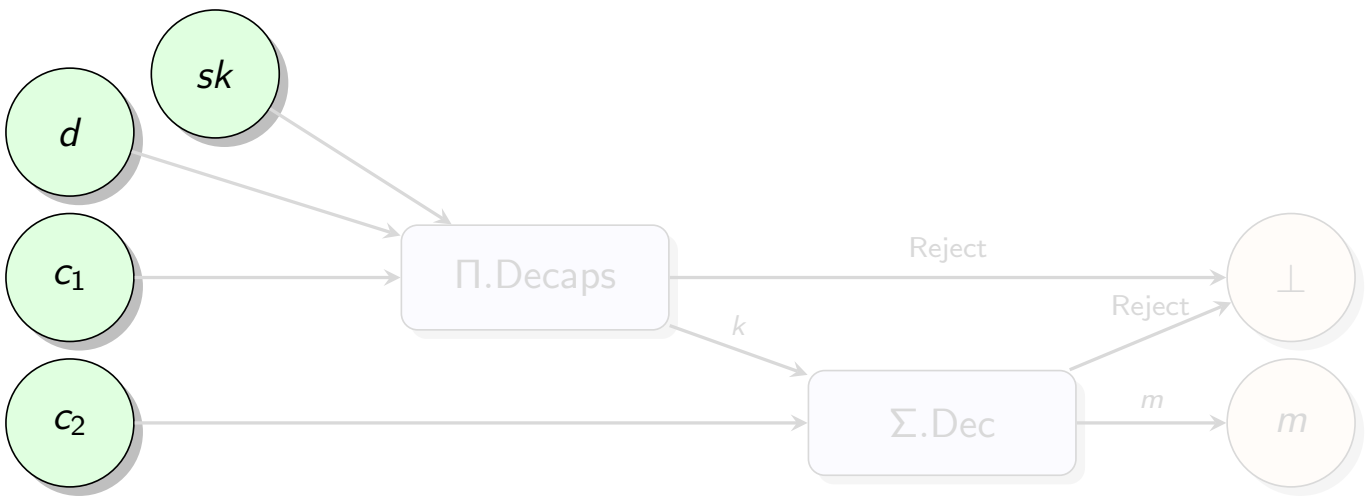


Generic Construction From PAKEM and DEM

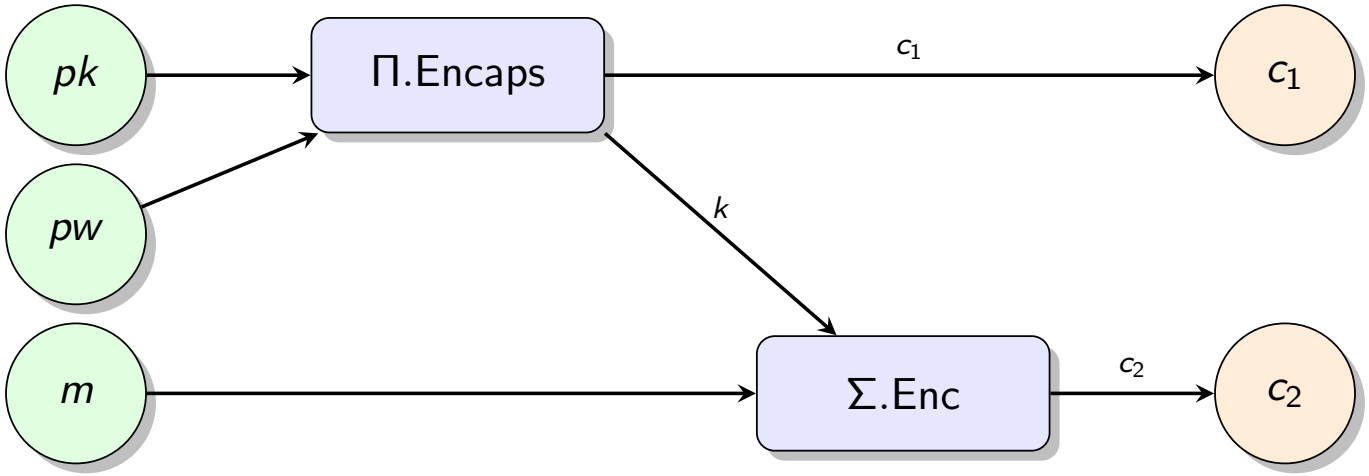


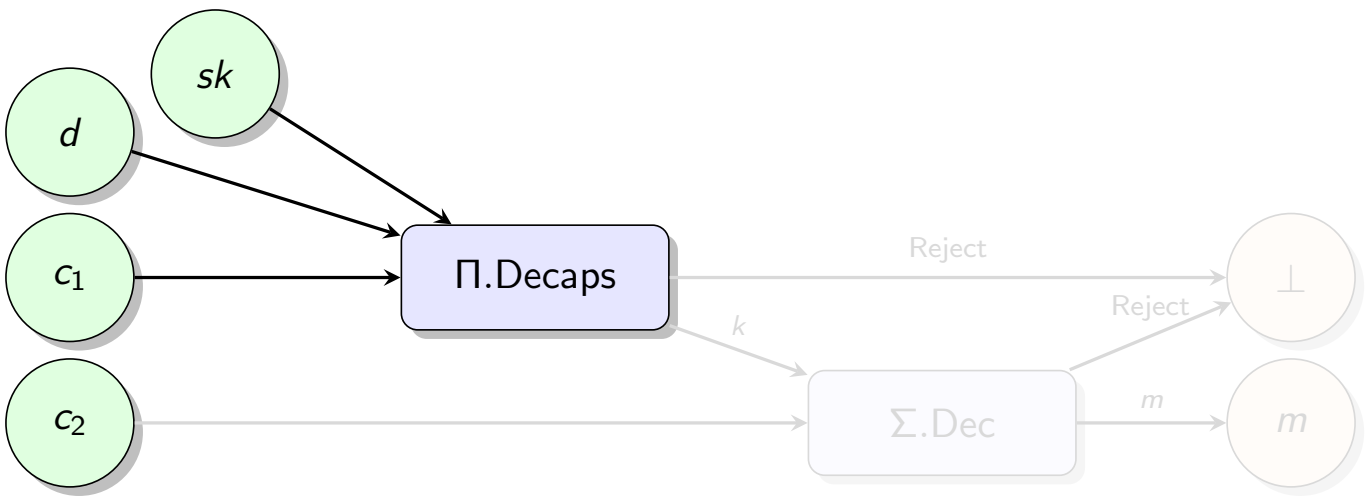
Generic Construction From PAKEM and DEM



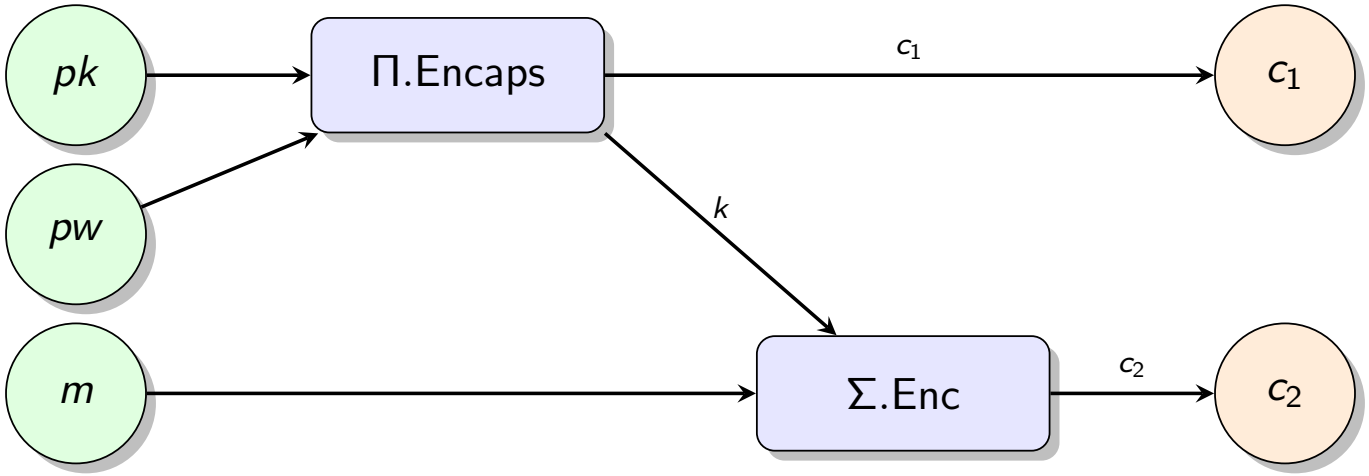


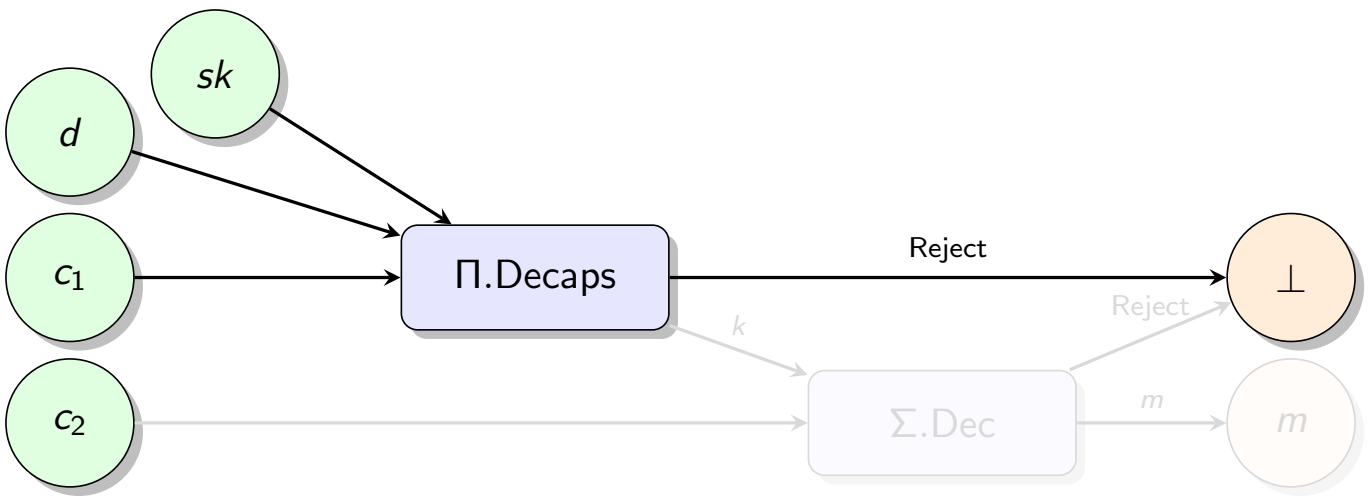
Generic Construction From PAKEM and DEM



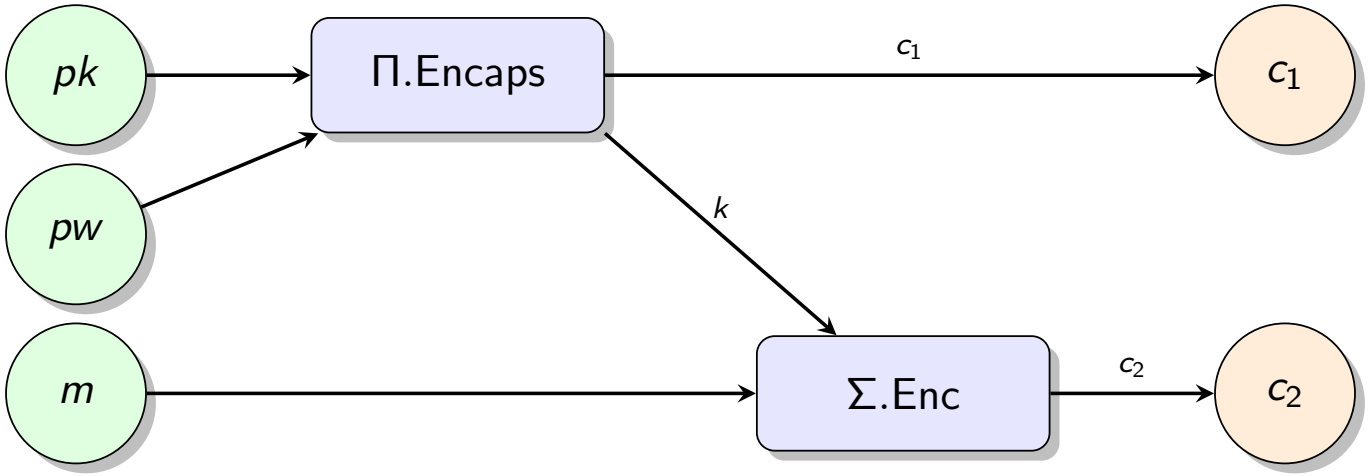


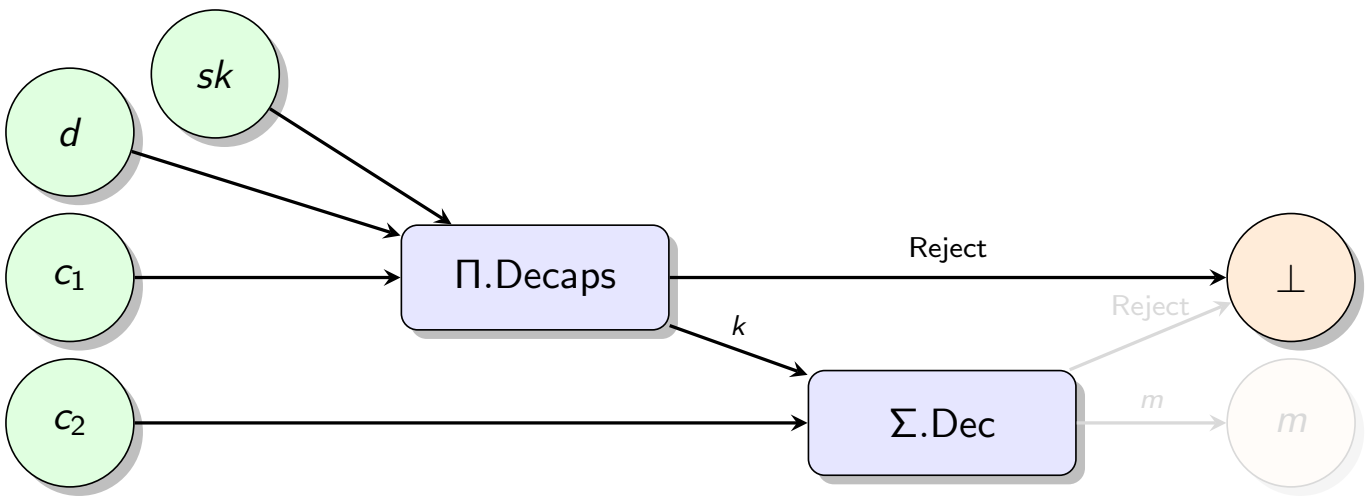
Generic Construction From PAKEM and DEM



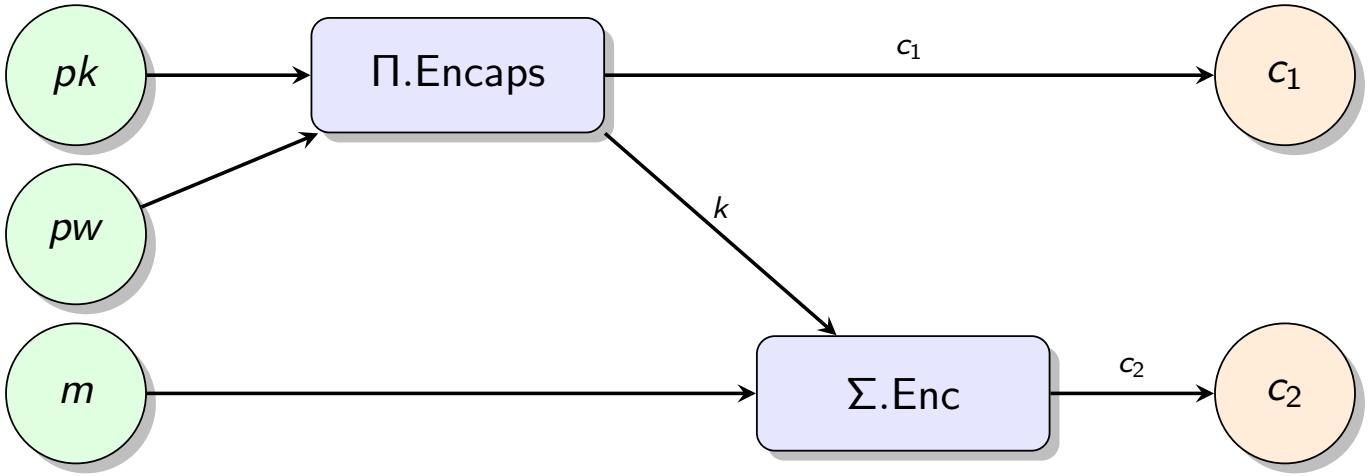


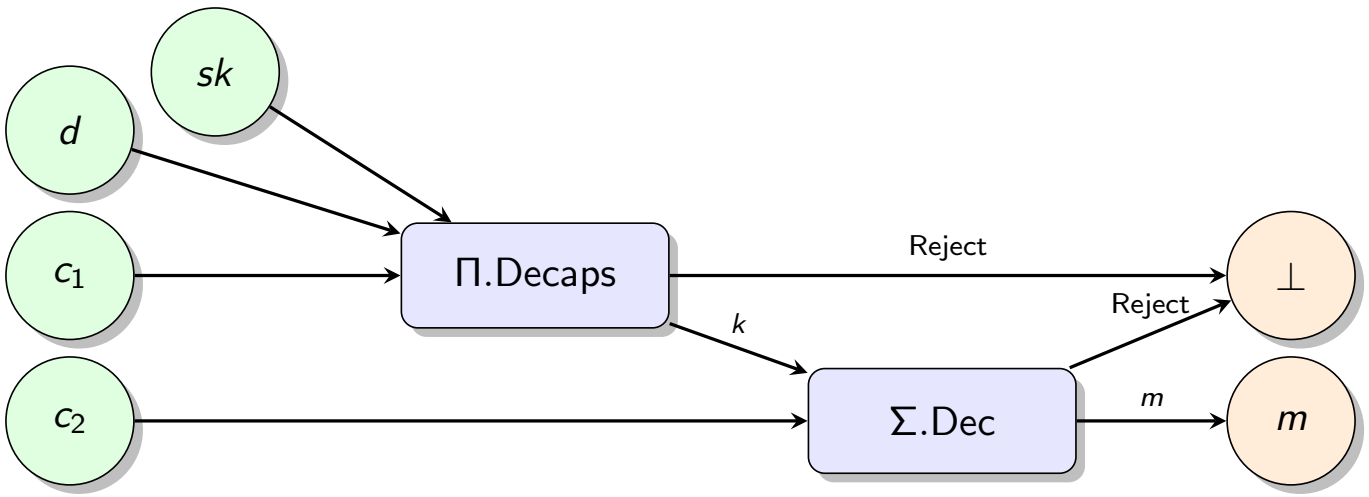
Generic Construction From PAKEM and DEM





Generic Construction From PAKEM and DEM





Proved Security

PAKEM	**DEM**	**PAHE**
Insider-CCA	IND-CCA	Insider-CCA
Outsider/Insider-Auth	INT-CTXT	Outsider-Auth

Table: Security assumptions of the construction and corresponding security.

PAHE Property	Reduction
Insider-CCA	$\mathrm{Adv}^{(q_d,q_c)\text{-Insider-CCA}}_{\mathcal{A},\mathrm{PAHE}} \leq \mathrm{Adv}^{(q_d,q_c)\text{-Insider-CCA}}_{\mathcal{B},\Pi}(\lambda) + q_c \cdot \mathrm{Adv}^{\text{IND-CCA}}_{\mathcal{C},\Sigma}$
Outsider-Auth	$\mathrm{Adv}^{(q_e,q_d,q_c)\text{-Outsider-Auth}}_{\mathcal{A},\mathrm{PAHE}} \leq \mathrm{Adv}^{(q_e,q_c)\text{-Insider-Auth}}_{\mathcal{B},\Pi} + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathcal{C},\Sigma}$

Table: Security reduction bounds for PAHE security.

Why Not Insider-Auth?

Achieving **Insider-Auth security** by directly composing a PAKEM and a DEM as **black boxes is impossible**.

- The adversary is given both sk (PAKEM) and d.
- It obtains a valid ciphertext (c_1, c_2) from the encryption oracle.
- Using sk, it decapsulates c_1 and recovers the symmetric key k.
- It uses k to compute a new ciphertext c_2' for a message of its choice.
- The resulting (c_1, c_2') is a fresh valid ciphertext.

Why Not Insider-Auth?

Achieving **Insider-Auth security** by directly composing a PAKEM and a DEM as **black boxes is impossible**.

- The adversary is given both sk (PAKEM) and d.
- It obtains a valid ciphertext (c_1, c_2) from the encryption oracle.
- Using sk, it decapsulates c_1 and recovers the symmetric key k.
- It uses k to compute a new ciphertext c_2' for a message of its choice.
- The resulting (c_1, c_2') is a fresh valid ciphertext.

Problem

The PAKEM ciphertext c_1 is **independent of the message**.

Table of Contents

Future Work Direction

Limitations:

- Security is define for two user setting
- PAHE is not Insider-Auth secure

Future Work:

- Replace the password by a biometric template
- Propose a direct construction of PAHE that is Insider-Auth secure
- Define the multi-user security and prove the security