SYNACKTIV

Retour d'expérience — Cryptographie post-quantique

Antoine Gicquel & Alexandre Brenner

Plan

SYNACKTIV

- **Pourquoi parler de cryptographie post-quantique ?**
- **Comment on s'est formés ?**
- **Et alors, c'est si compliqué que ça ?**
  - Signatures sur les fonctions de hachage
  - Échanges de clés sur les lattices
  - Hybridation

Pourquoi parler de cryptographie post-quantique ?

Pourquoi ?

La « menace quantique »

SYNACKTIV

- **Développement d'ordinateurs quantiques**
  - 10 ans ? 20 ans ?
  - Mieux vaut prévenir que guérir
- **Algorithme de Grover**
  - Recherche exhaustive en $\sqrt{n}$ (plutôt que n avec un bruteforce « classique »)
  - Affaiblit la cryptographie symétrique (facteur 2 sur la taille des clés et des hashs)
- **Algorithme de Shor**
  - Trouve « rapidement » des périodes dans des fonctions
  - Résout efficacement le logarithme discret & la factorisation

Voir SSTIC 2024 - La transition à la cryptographie post quantique, fardeau ou aubaine? de Carlos Aguilar Melchor

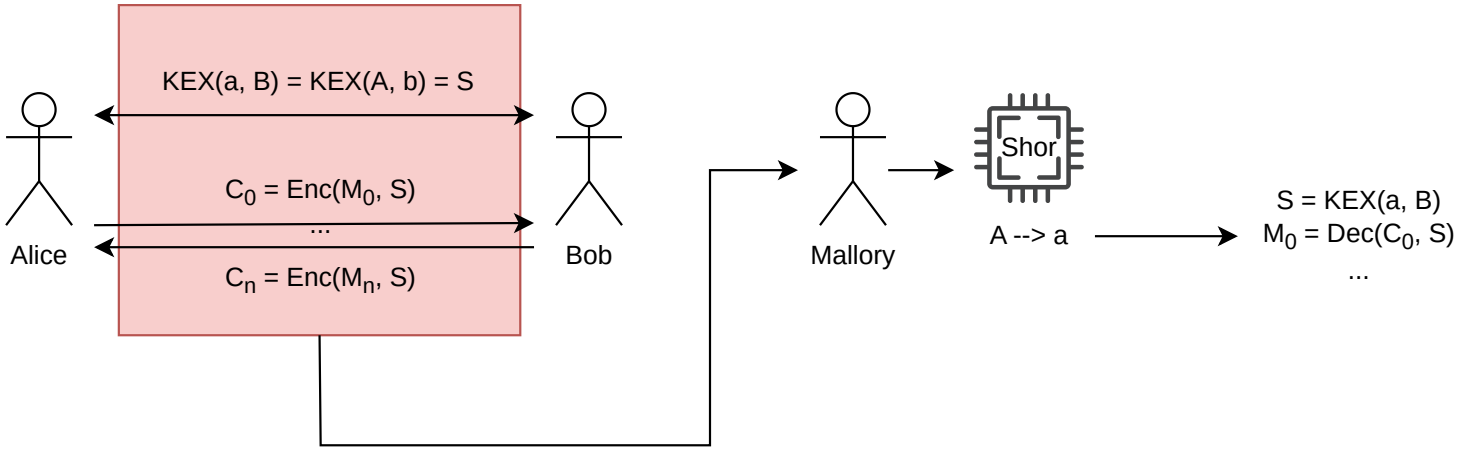
Pourquoi ?

La « menace quantique »

Pourquoi ?

... dès aujourd'hui

- **"Store now & decrypt later"**
  - Capture et stockage d'un échange de clés & des communications associées aujourd'hui
  - Cassage du problème mathématique avec un ordi quantique → récupération de la clé
  - Déchiffrement des communications



Pourquoi ?

... parce qu'on est des auditeurs !

SYNACKTIV

- Vous vous dites peut-être : *« OK, mais en quoi ça me concerne ? »*
  → En tant qu'auditeur·rice, vous tomberez tôt ou tard sur du post-quantique : mieux vaut être prêt·e !
- Pas de magie noire : on garde les **mêmes réflexes** que sur RSA ou ECC (timings, aléa, canaux auxiliaires, ...).
- Bref, pour des auditeur·rices comme **vous et nous**, la PQC, c'est la **même cuisine** avec de nouveaux ingrédients.

Pourquoi ?

... parce que l'ANSSI le recommande fortement !

SYNACKTIV

> L'utilisation d'une protection **hybride post-quantique** est recommandée en particulier pour les produits de sécurité destinés à offrir une protection durable des informations (jusqu'à après 2030) ou qui seront potentiellement utilisés après 2030 sans mise à jour.

Avis de l'ANSSI sur la migration vers la cryptographie post-quantique

Comment on s'est formés ?

Comment ?

Point de départ

- **Sondage initial en interne**
  - ~10 consultants curieux & intéressés
  - Connaissances en cryptographie classique
  - Pas/peu habitués à lire des papiers scientifiques
- **Travaux préliminaires**
  - Semaines de « formation » en crypto post-quantique en solo, à base de recherches Google
  - Lecture de publications scientifiques, complexe, parfois fastidieux
  - *Is it post quantum time yet?* par Gaëtan Ferry sur le blog Synacktiv en 2021

Comment ?

Formation – QSI Spring School

SYNACKTIV

- **Quantum-Safe Internet**
  - Réseau doctoral européen « *formant une cohorte de chercheurs en cybersécurité de classe mondiale à adresser le défi de proposer un internet Quantum-Safe* »
- **QSI Spring School** [1]
  - Semaine de conférences / cours d'introduction à la cryptographie PQ
  - Public cible : doctorants
  - Présentations détaillées sur divers sous-domaines
  - 5 auditeurs sur place

1: https://pqc-spring-school.nl/

Comment ?

Formation – QSI Spring School

SYNACKTIV

	Tuesday	Wednesday	Thursday	Friday
9:00-10:30	**Crypto 101** *Kathrin Hövelmanns*	**Codes** *Alexander May*	**Lattices** *Wessel van Woerden*	**Multivariate quadratic crypto** *Monika Trimoska*
11:00-12:30	**Quantum random oracles** *Christian Schaffner*			**Isogenies** *Lorenz Panny*
12:30-14:00	**Lunch**			
14:00-15:30	**Symmetric crypto** *André Schrottenloher*	**Hash-based crypto** *Andreas Hülsing*	**Free afternoon** Optional: walking tour Porto	**Post-quantum implementations** *Bas Westerbaan*
16:00-17:30	**Multi-party computation in the head** *Andreas Hülsing*			
19:00 - ?			**Dinner** Restaurante Escondidinho R. de Passos Manuel 144	

Comment ?

Après la formation

SYNACKTIV

- **Reprise de la lecture de papiers**
  - Notamment sur les attaques
  - Papiers plus abordables maintenant que nous avons davantage de connaissances sur les problèmes et les notations
- **Publications sur le blog Synacktiv**
  - *Quantum readiness: Introduction to Modern Cryptography*
  - *Quantum readiness: Hash-based signatures*
  - *Quantum readiness: Lattice-based PQC*
  - *... et d'autres à venir*

Et alors, c'est si compliqué que ça ?

De nouveaux algorithmes

SYNACKTIV

- **Menace principalement sur la crypto asymétrique (Échanges de clés / Signatures)**
- **Algorithmes classiques cassés par Shor**
  - (Elliptic-Curve)-Diffie-Hellman, (EC)DSA, RSA, …
- **« Nouveaux » algorithmes utilisant d'autres problèmes mathématiques** [1]
  - **Hashs** : SPHINCS+ (Signature), …
  - **Lattices** : Dilithium (Signature), Falcon (Signature), Kyber (Échange de clés), …
  - **Codes** : BIKE (Échange de clés), HQC (Échange de clés), …
  - **…**

1: Standardisés par le NIST en 2024 pour la plupart

Les signatures

SLH-DSA

SLH-DSA

Un exemple de signature post-quantique

SYNACKTIV

- **Hypothèse** : il est difficile pour un attaquant (même quantique[1]) de trouver une pré-image ou une collision sur les fonctions de hachage utilisées
- **Avantage** : pas de « nouveaux » problèmes mathématiques, bonne confiance
- ⚠️ **Nous avons peu de temps, beaucoup de raccourcis seront pris[2] !**

1: L'algorithme de Grover (et ses dérivés) donnent ***au pire*** une division par 2 de la robustesse des fonctions de hachage → SHA-384 toujours solide face aux collisions

2: Voir les actes pour davantage de détails

SLH-DSA

Étape 1 – Les chaînes de hashs

SYNACKTIV

- **Hachage répété du même blob privé pour obtenir un blob public**
- **Calcul à sens unique**
- **Révéler un « parent » de P prouve qu'on est bien l'émetteur**

SLH-DSA

Étape 1 – Les chaînes de hashs

SYNACKTIV

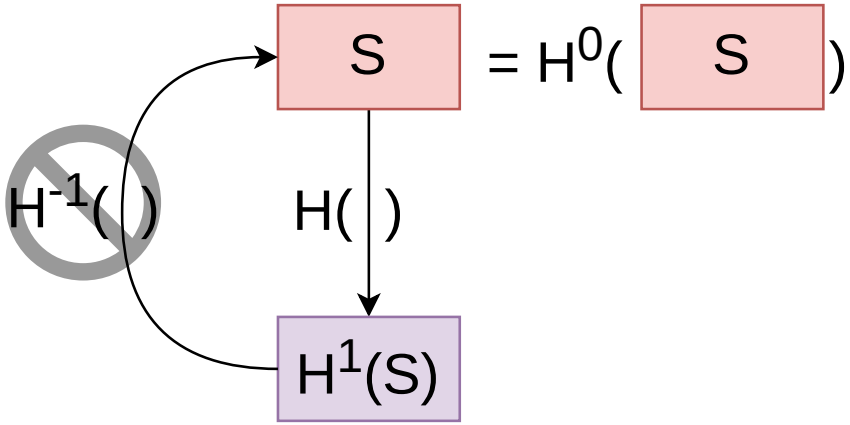
- **Hachage répété du même blob privé pour obtenir un blob public**
- **Calcul à sens unique**
- **Révéler un « parent » de P prouve qu'on est bien l'émetteur**

$$\boxed{S} = H^0(\boxed{S})$$

SLH-DSA

Étape 1 – Les chaînes de hashs

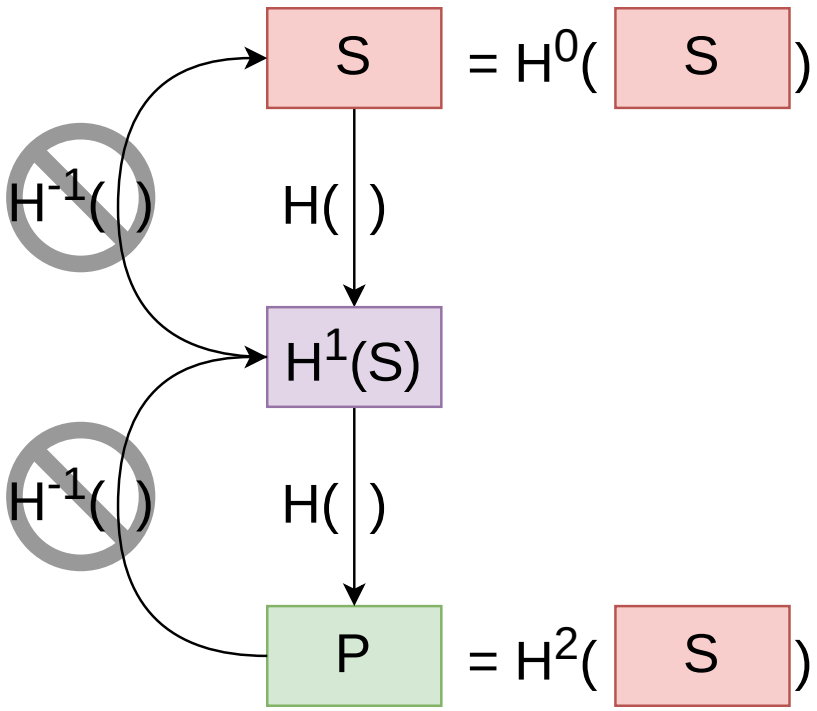
- **Hachage répété du même blob privé pour obtenir un blob public**
- **Calcul à sens unique**
- **Révéler un « parent » de P prouve qu'on est bien l'émetteur**



SLH-DSA

Étape 1 – Les chaînes de hashs

- **Hachage répété du même blob privé pour obtenir un blob public**
- **Calcul à sens unique**
- **Révéler un « parent » de P prouve qu'on est bien l'émetteur**

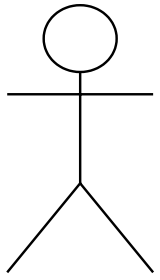


SLH-DSA

Étape 1 – Les chaînes de hashs

SLH-DSA

Étape 1 – Les chaînes de hashs



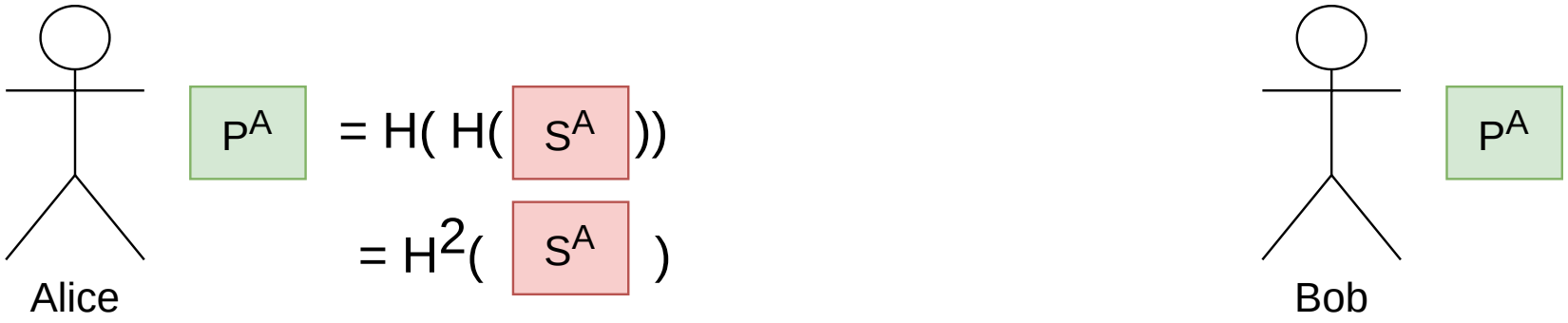
Alice

$$P^A = H(\,H(\,S^A\,))$$

$$= H^2(\,S^A\,)$$

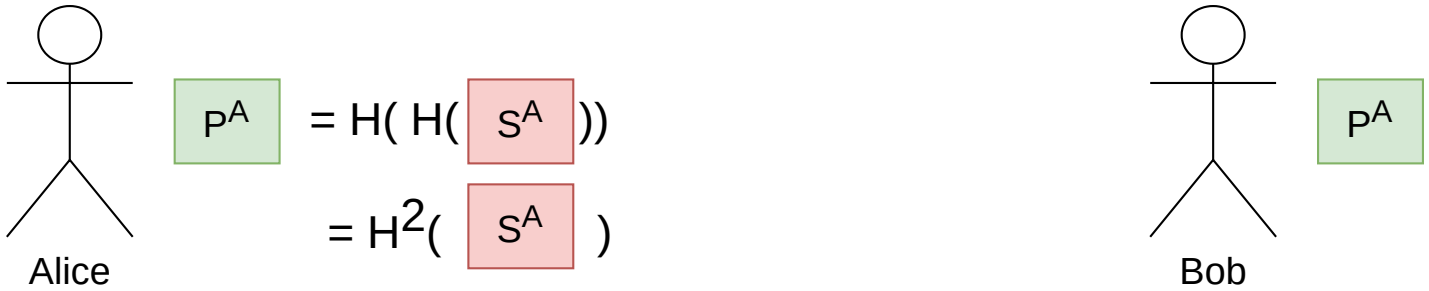
SLH-DSA

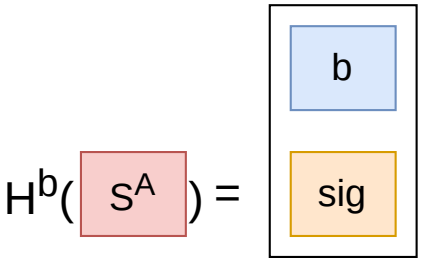
Étape 1 – Les chaînes de hashs



SLH-DSA

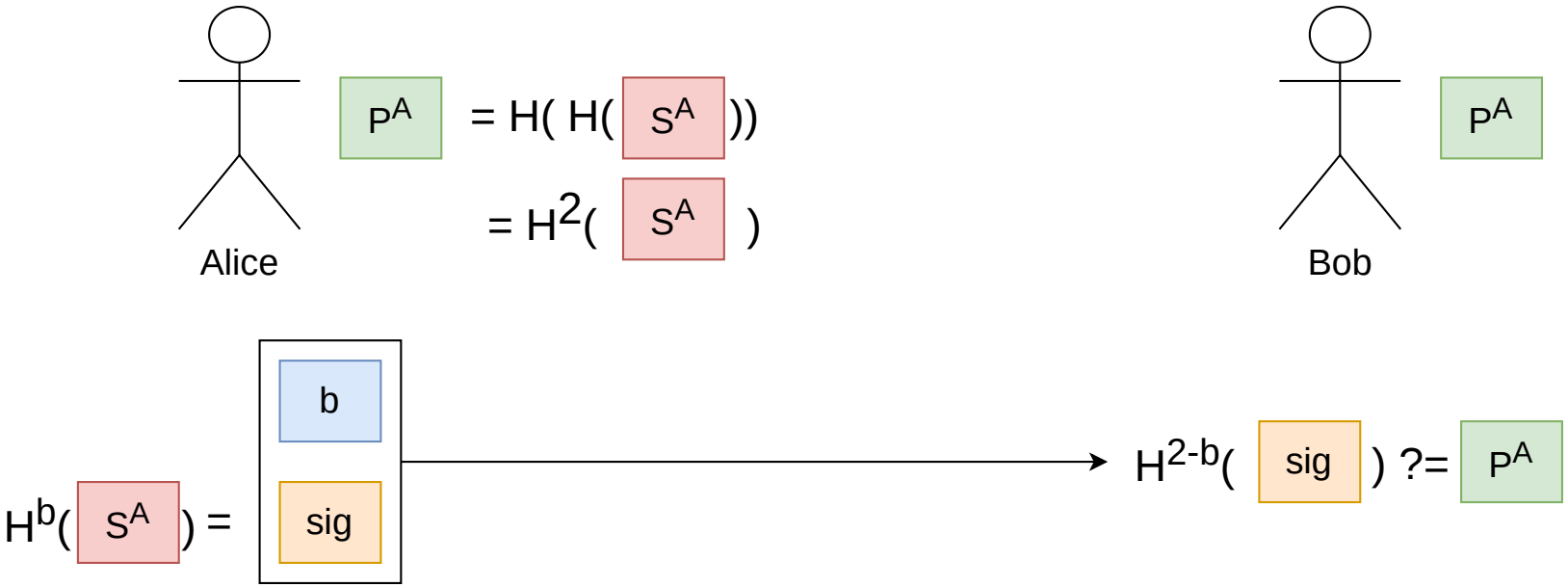
Étape 1 – Les chaînes de hashs





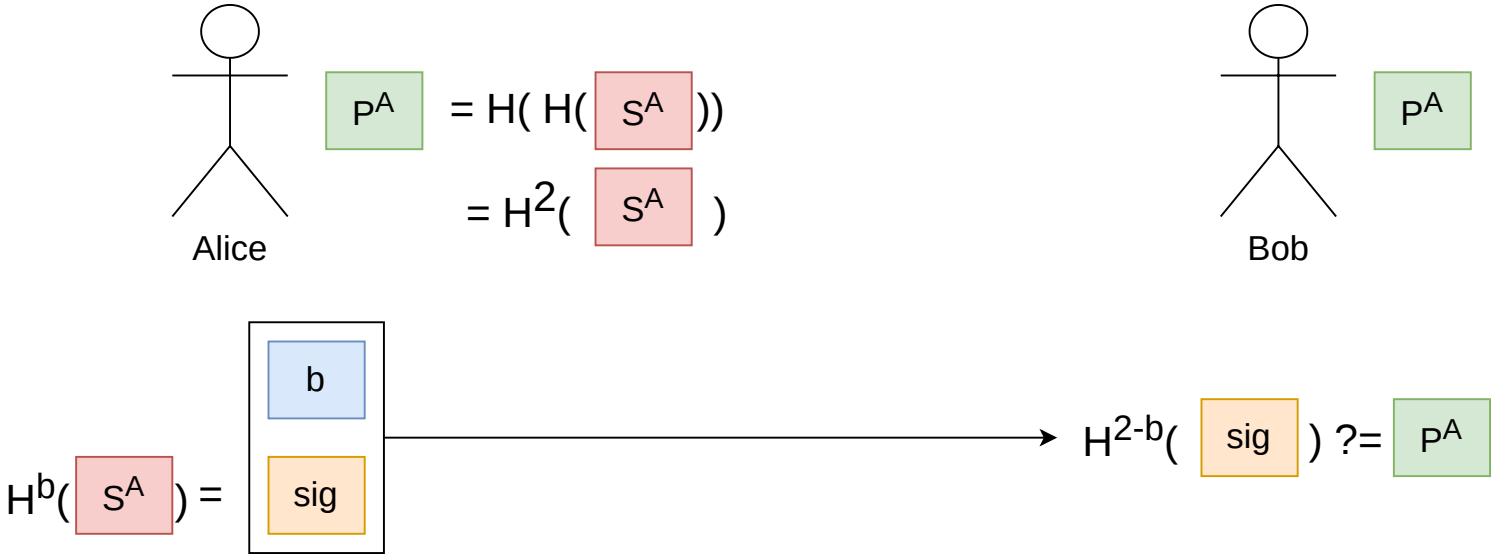
SLH-DSA

Étape 1 – Les chaînes de hashs



SLH-DSA

Étape 1 – Les chaînes de hashs



- **Problème : si Alice signe b = 0, alors un attaquant pourra signer b = 1**
  - Sign(1, S) = H(Sign(0, S))
  - → Il faut une seconde chaîne, avancée de n-b pas
- **Complexité exponentielle** en la taille du message

SLH-DSA

Chaîne de contrôle

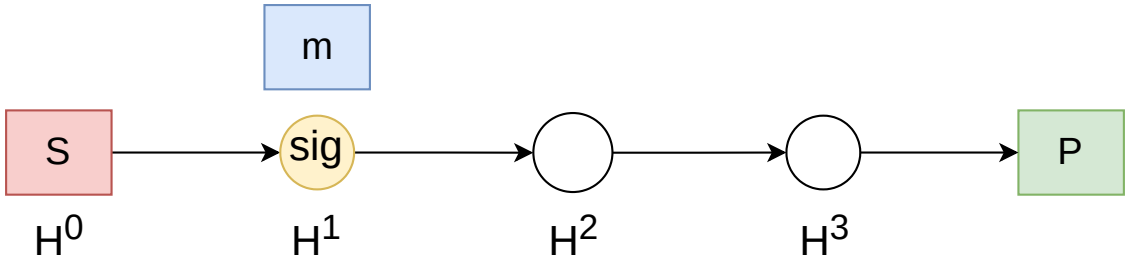
SYNACKTIV

- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**

SLH-DSA

Chaîne de contrôle

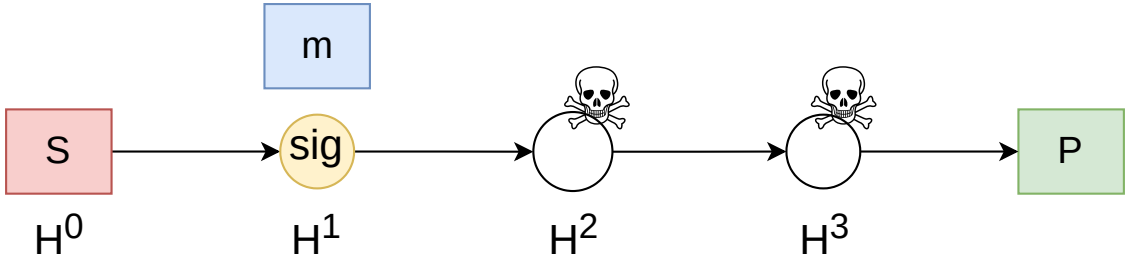
- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**



SLH-DSA

Chaîne de contrôle

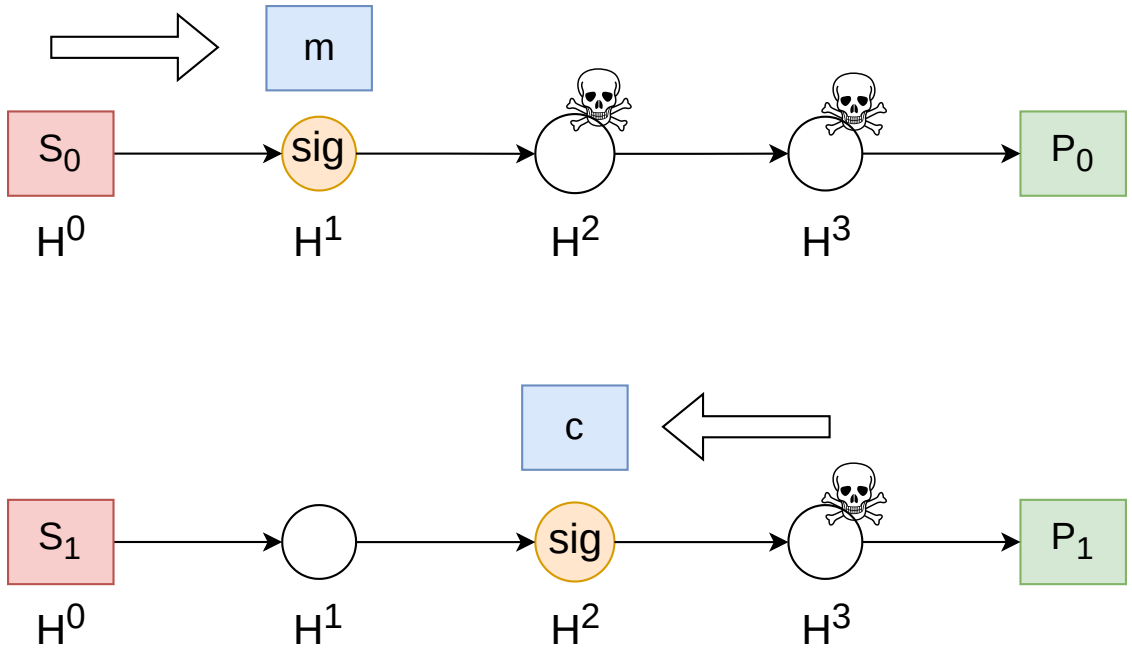
- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**



SLH-DSA

Chaîne de contrôle

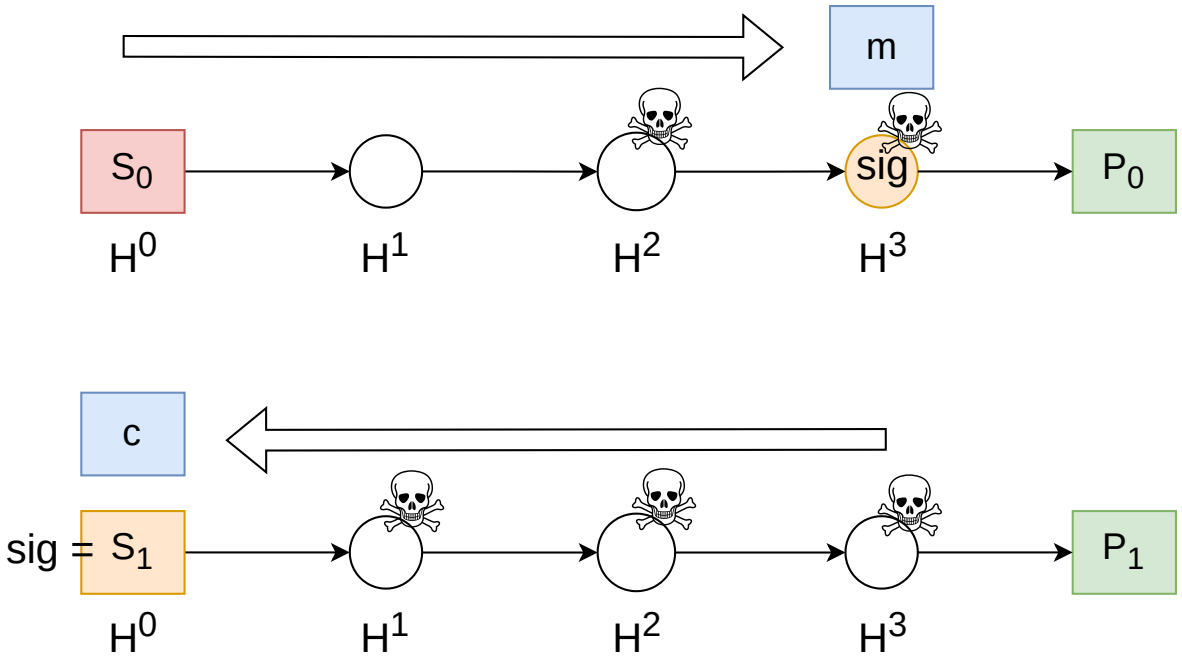
- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**



SLH-DSA

Chaîne de contrôle

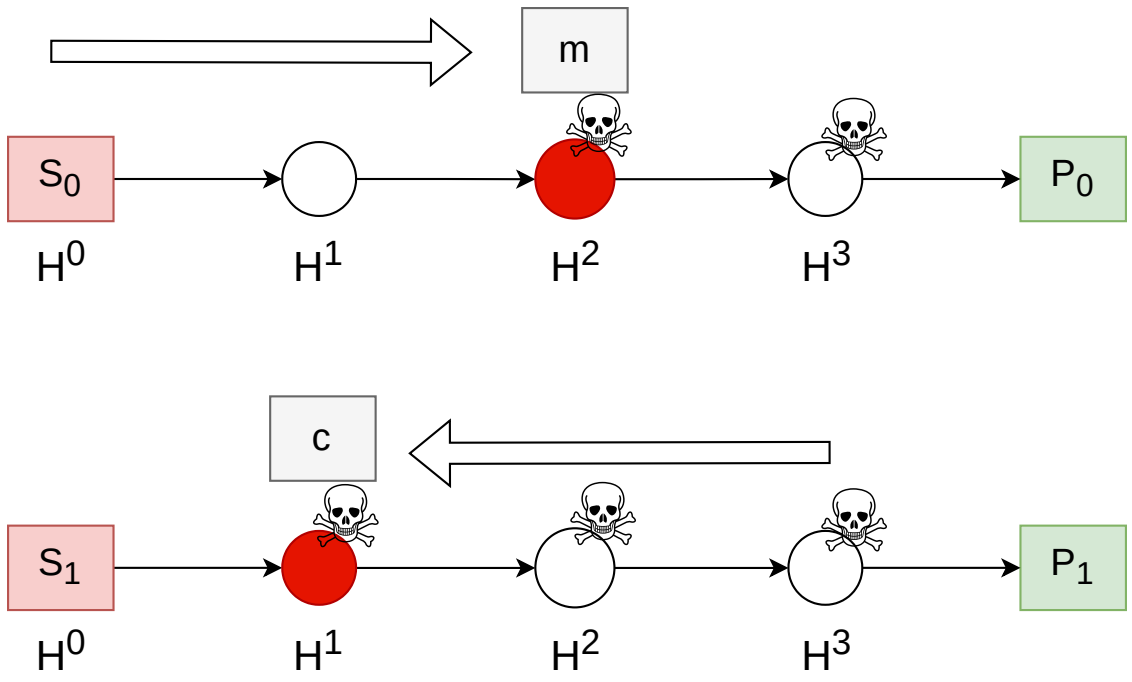
- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**



SLH-DSA

Chaîne de contrôle

- Signature de $m = 1$: **compromission** des signatures de $m \geq 2$
- Introduction d'une **seconde chaîne**
- **Avancer m = reculer c**
- ⚠️ **Usage unique !**



SLH-DSA

SYNACKTIV

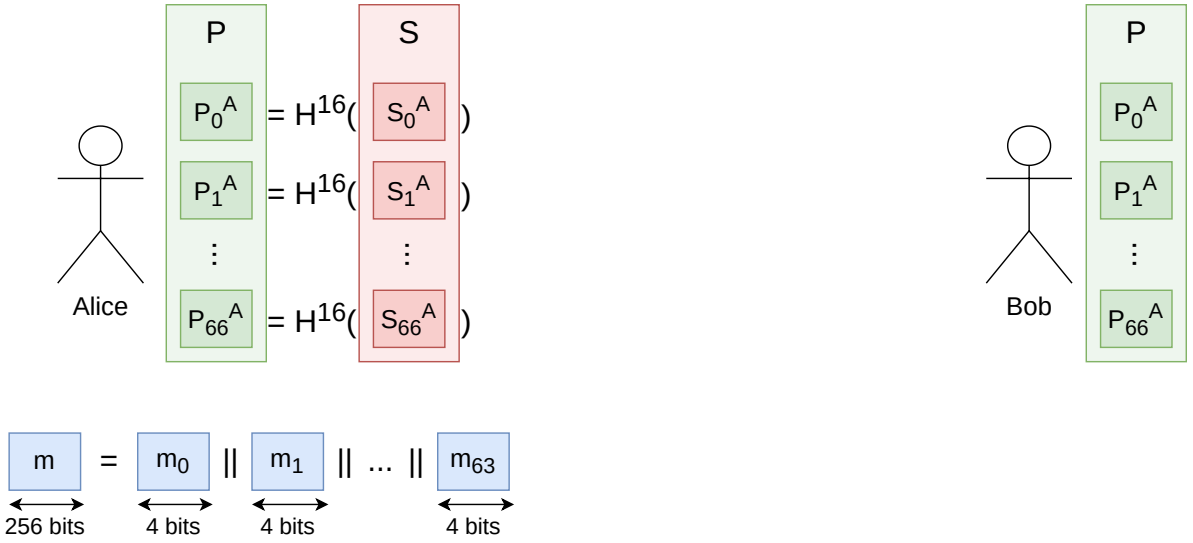
Étape 1 – Les chaînes de hashs & Winternitz One-Time Signature

- Découpage du message en blocs
  → plusieurs chaînes en parallèle
- Les chaînes sont plus longues (16 maillons en général) pour avoir des blocs de plusieurs bits (4 bits si 16 maillons)
- **WOTS** : Winternitz One-Time Signature

SLH-DSA

Étape 1 – Les chaînes de hashs & Winternitz One-Time Signature

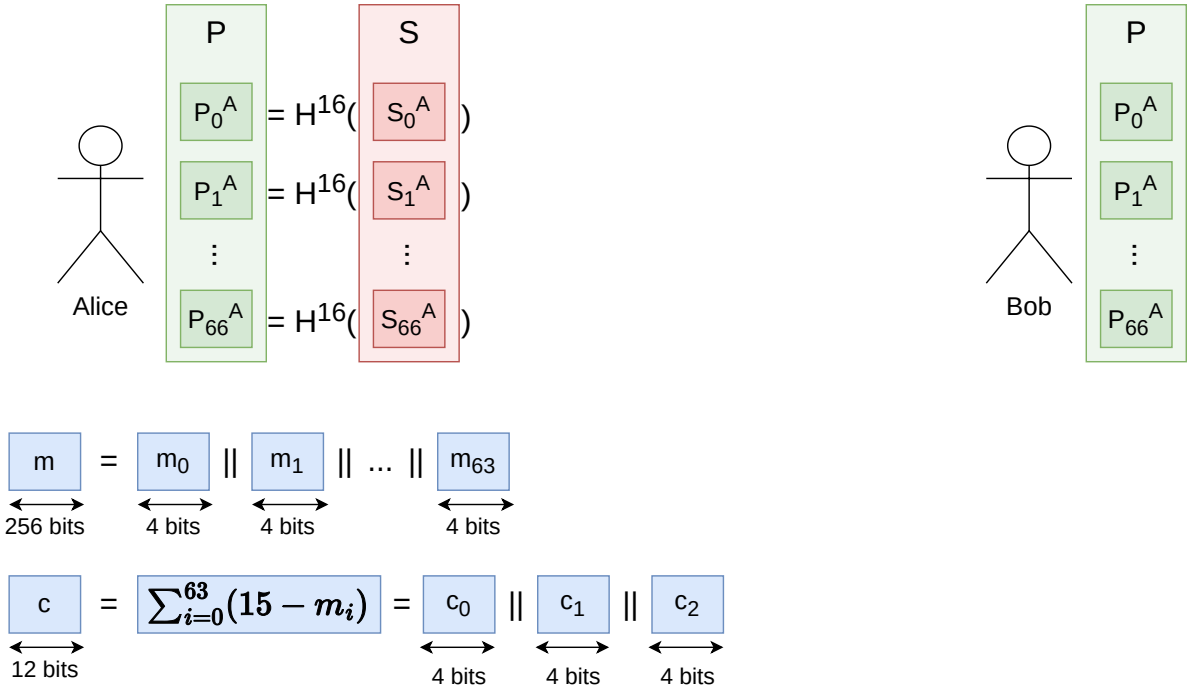
- Découpage du message en blocs → plusieurs chaînes en parallèle
- Les chaînes sont plus longues (16 maillons en général) pour avoir des blocs de plusieurs bits (4 bits si 16 maillons)
- **WOTS** : Winternitz One-Time Signature



SLH-DSA

Étape 1 – Les chaînes de hashs & Winternitz One-Time Signature

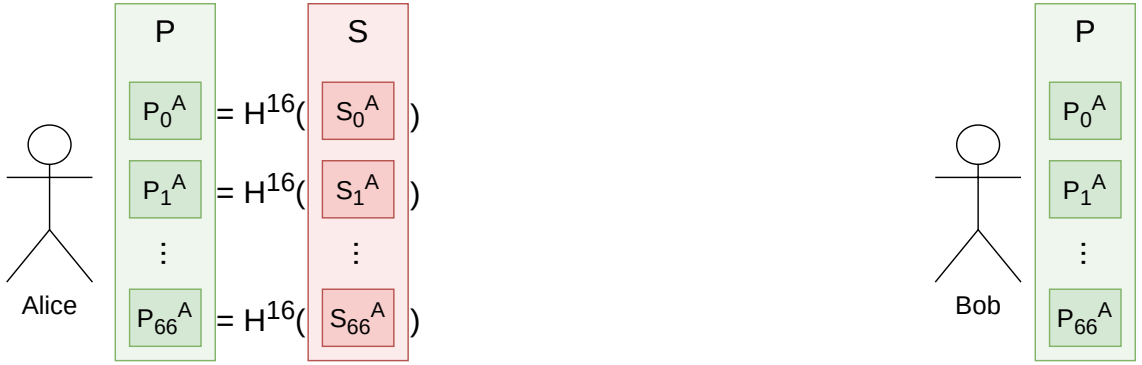
- Découpage du message en blocs → plusieurs chaînes en parallèle
- Les chaînes sont plus longues (16 maillons en général) pour avoir des blocs de plusieurs bits (4 bits si 16 maillons)
- **WOTS** : Winternitz One-Time Signature



SLH-DSA

Étape 1 – Les chaînes de hashs & Winternitz One-Time Signature

- Découpage du message en blocs → plusieurs chaînes en parallèle
- Les chaînes sont plus longues (16 maillons en général) pour avoir des blocs de plusieurs bits (4 bits si 16 maillons)
- **WOTS** : Winternitz One-Time Signature



$$\underbrace{m}_{256\text{ bits}} = \underbrace{m_0}_{4\text{ bits}} \,\|\, \underbrace{m_1}_{4\text{ bits}} \,\|\, \dots \,\|\, \underbrace{m_{63}}_{4\text{ bits}}$$

$$\underbrace{c}_{12\text{ bits}} = \sum_{i=0}^{63}(15 - m_i) = \underbrace{c_0}_{4\text{ bits}} \,\|\, \underbrace{c_1}_{4\text{ bits}} \,\|\, \underbrace{c_2}_{4\text{ bits}}$$

$$sig = H^{m_0}(S_0^A) \,\|\, H^{m_1}(S_1^A) \,\|\, \dots \,\|\, H^{m_{63}}(S_{63}^A) \,\|\, H^{c_0}(S_{64}^A) \,\|\, H^{c_1}(S_{65}^A) \,\|\, H^{c_2}(S_{66}^A)$$

SLH-DSA

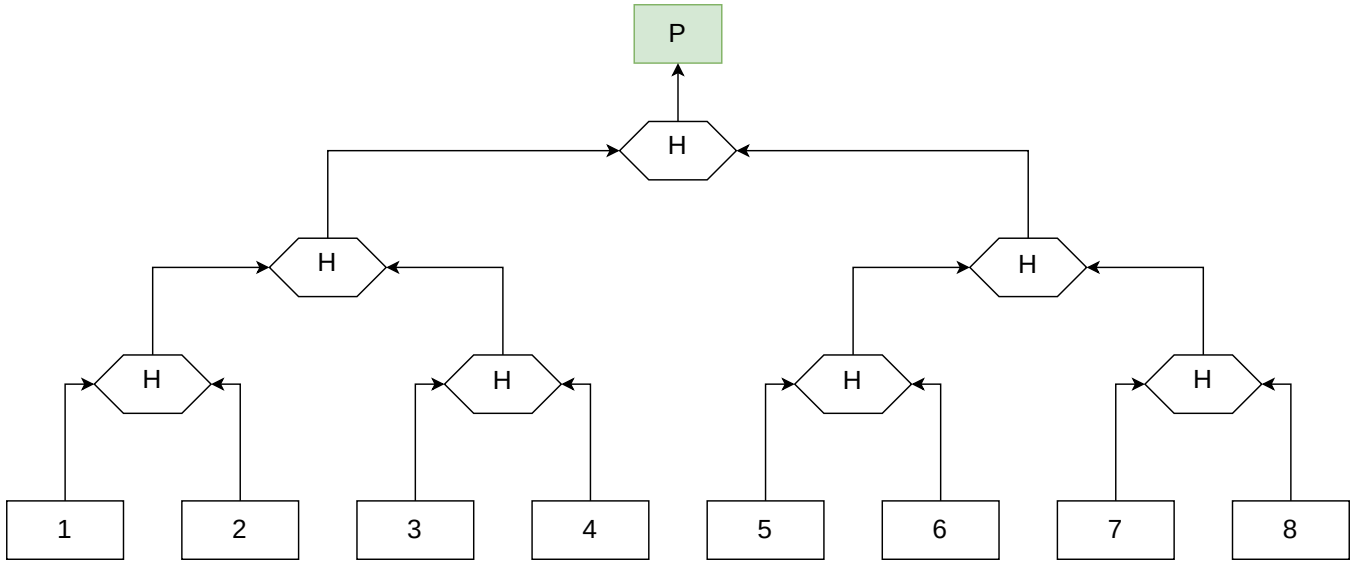
Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**

SLH-DSA

Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

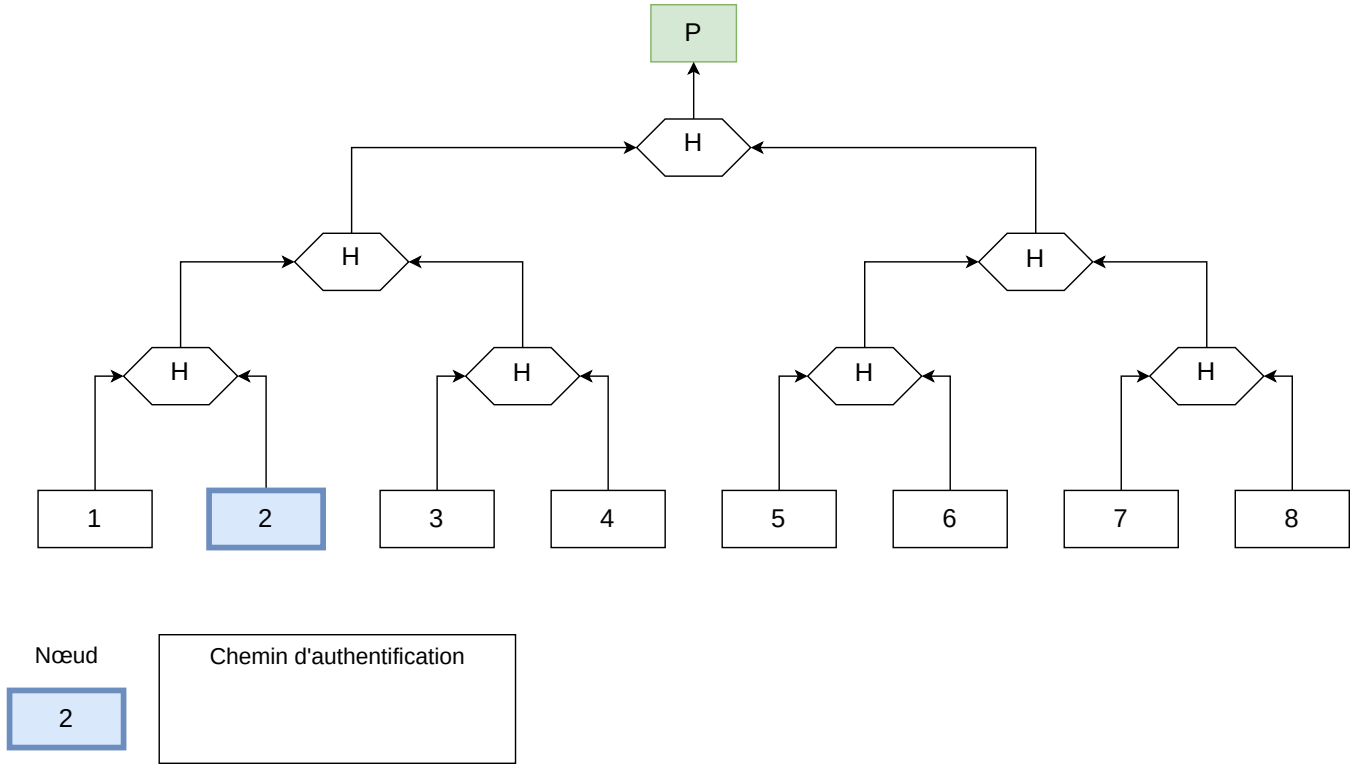
- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**



SLH-DSA

Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

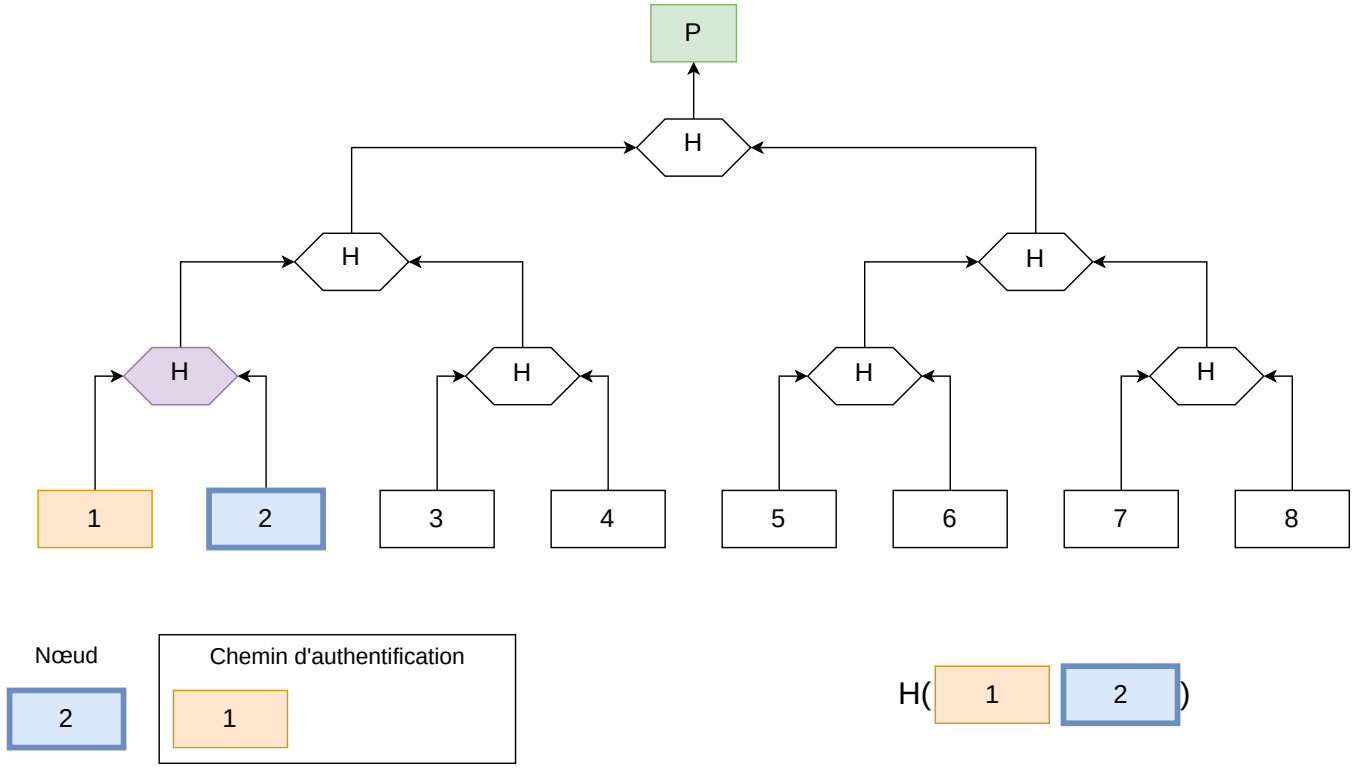
- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**



SLH-DSA

Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

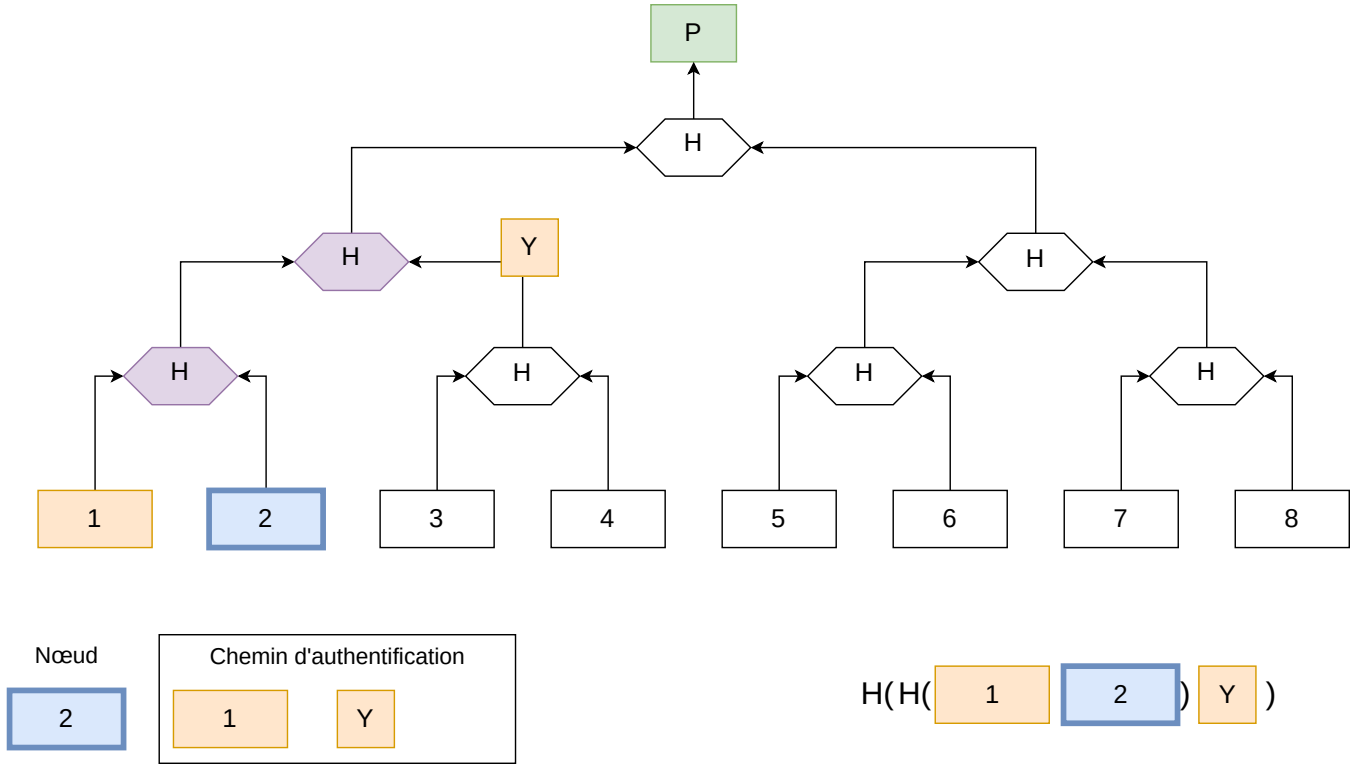
- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**



SLH-DSA

Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

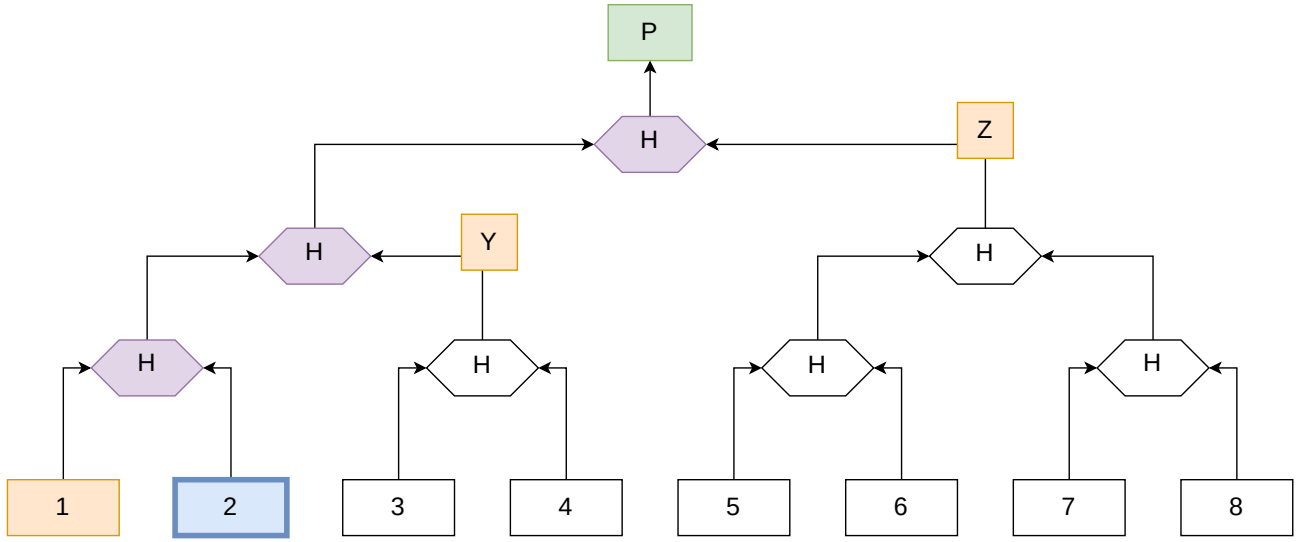
- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**

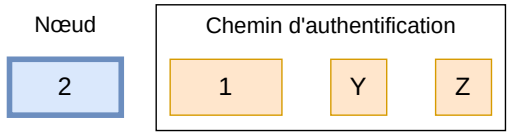


SLH-DSA

Étape 2 – Les arbres de Merkle & le Merkle Signature Scheme

- **Arbre binaire**, chaque nœud est le haché des deux enfants concaténés
- Utilisé pour condenser plusieurs valeurs en une seule
- Pour prouver qu'une valeur appartient à l'arbre, on publie le **chemin d'authentification**



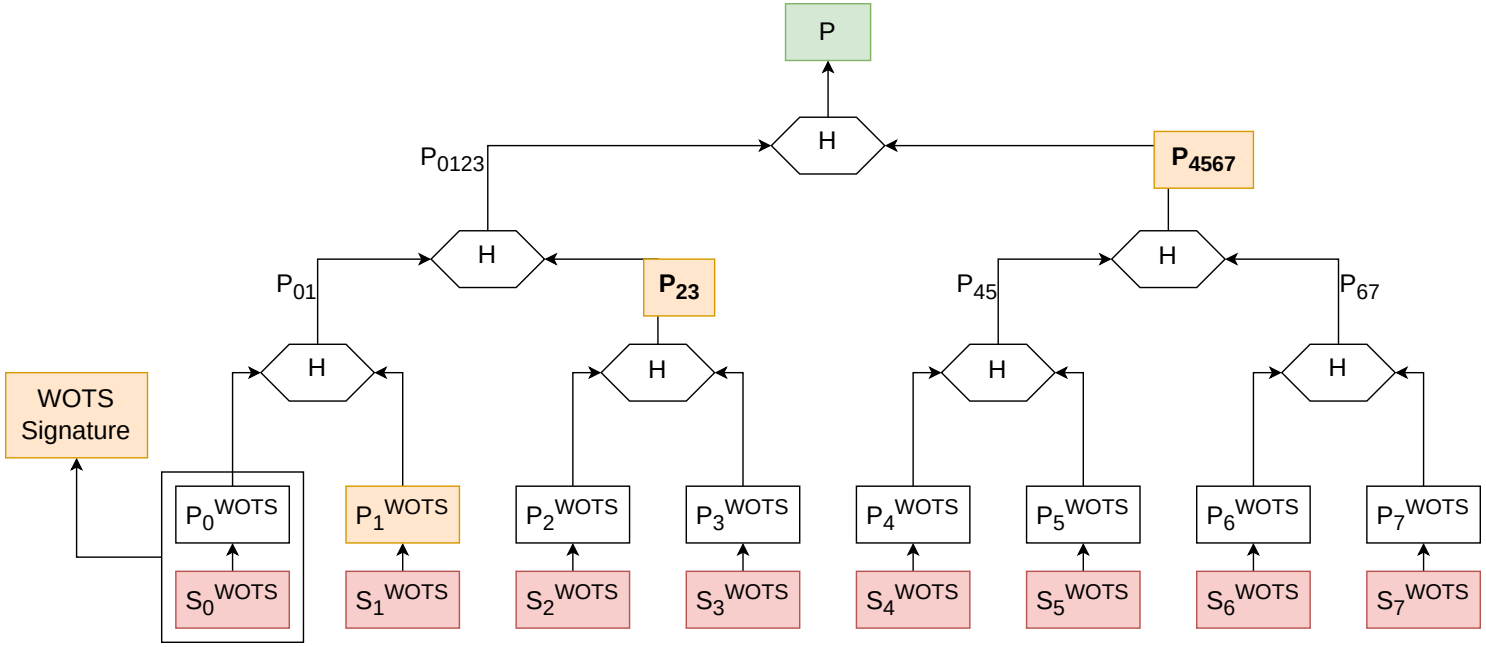


$$P \stackrel{?}{=} H(H(H(1\ 2)\ Y)\ Z)$$

SLH-DSA

Étape 2 – Les arbres de Merkle & XMSS

- **Utile pour compresser plusieurs clés WOTS**
  - Pour signer, on publie la **signature WOTS** et le **chemin d'authentification** de la clé
  - → Signature à usage **multiple**



SLH-DSA

Étape 2.5 – XMSS^{MT}

SYNACKTIV

- **Pourquoi s'arrêter là ?**

SLH-DSA

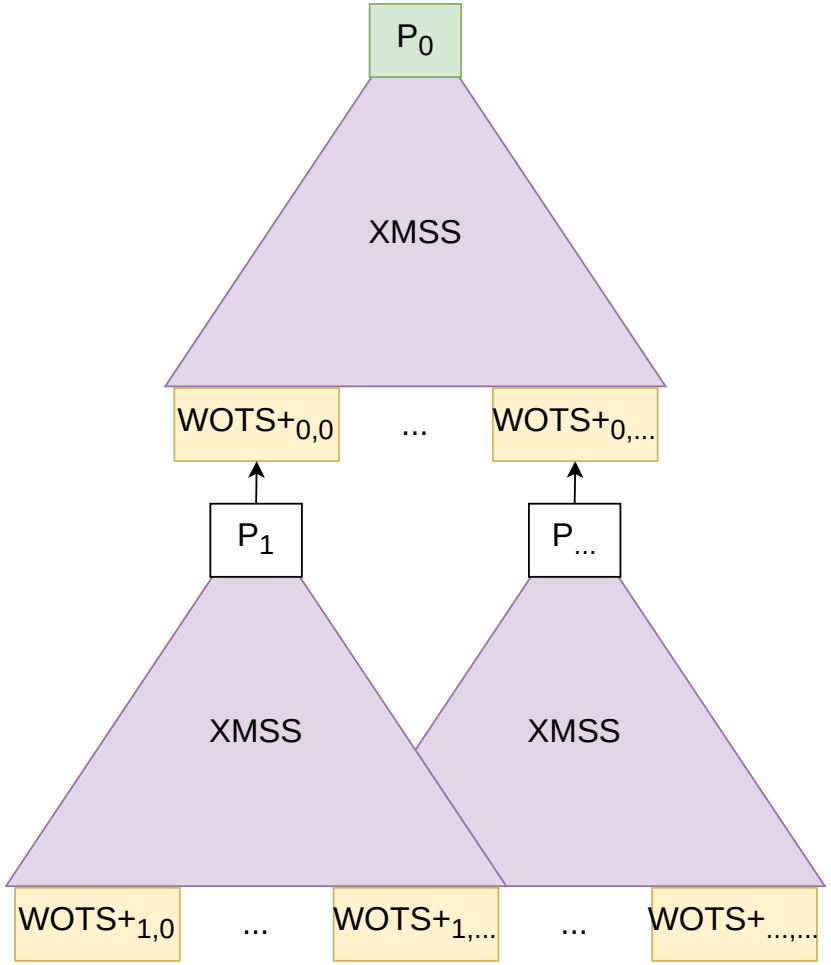
Étape 2.5 – XMSS^{MT}

- **Pourquoi s'arrêter là ?**

SLH-DSA

Étape 2.5 – XMSS^{MT}

- Empliement d'arbres pour stocker encore plus de clés : **XMSS Multi-Tree**
- Toujours besoin de maintenir un **état** (clés WOTS consommées)
- **→ Et si on trouvait une primitive de signature utilisable plusieurs fois ?**



SLH-DSA

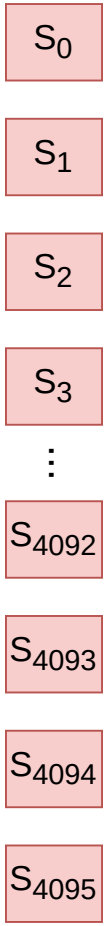
Étape 3 – HORS, Hash to Obtain a Random Subset

- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**

SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

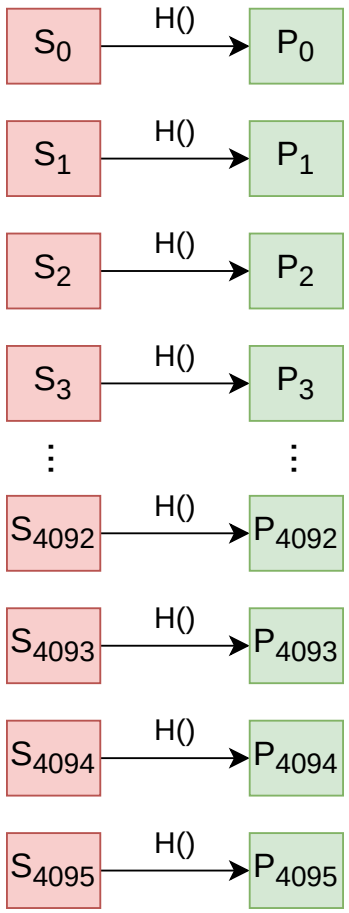
- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**



SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

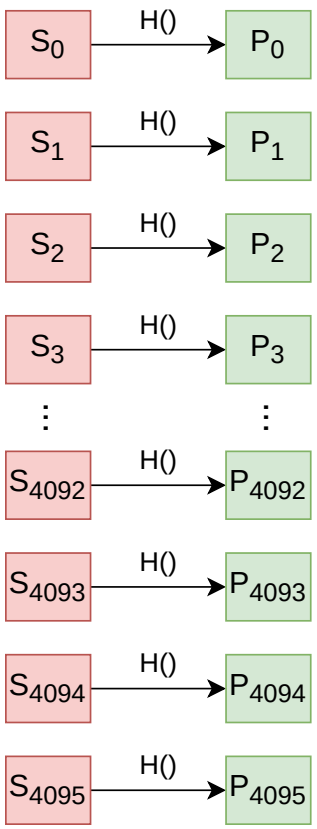
- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**



SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**

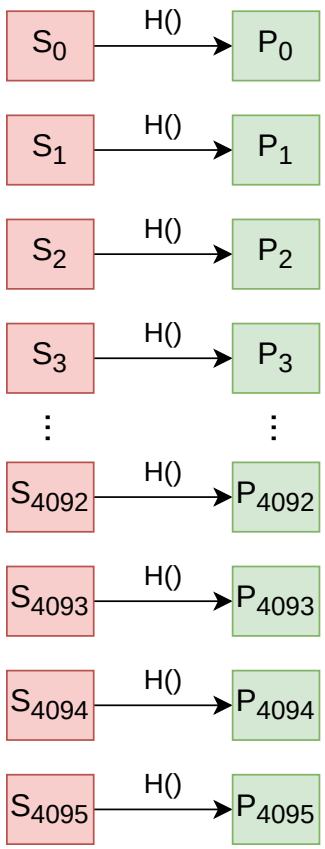


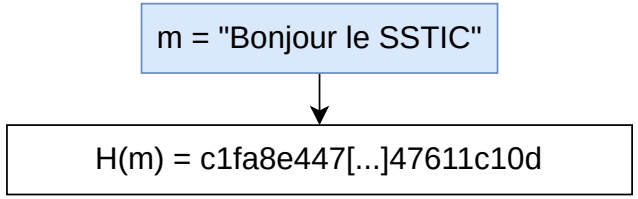
m = "Bonjour le SSTIC"

SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**

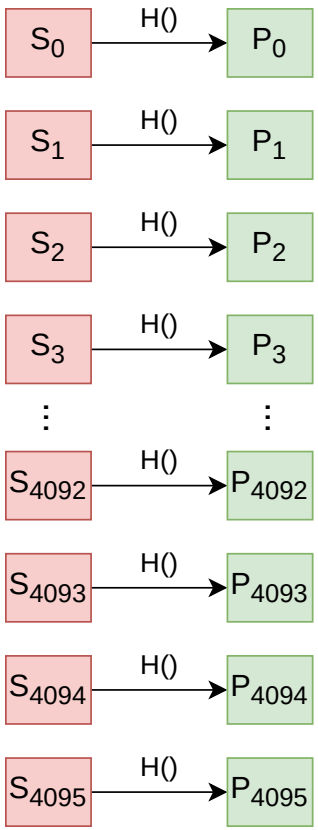


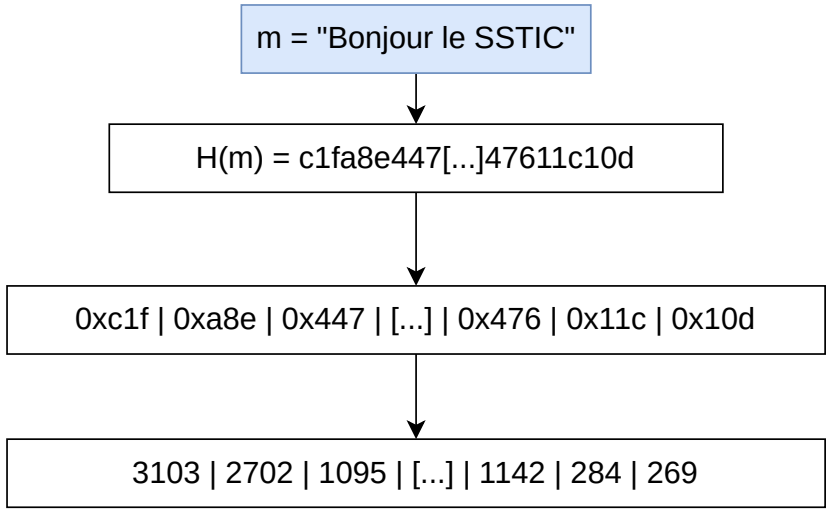


SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**

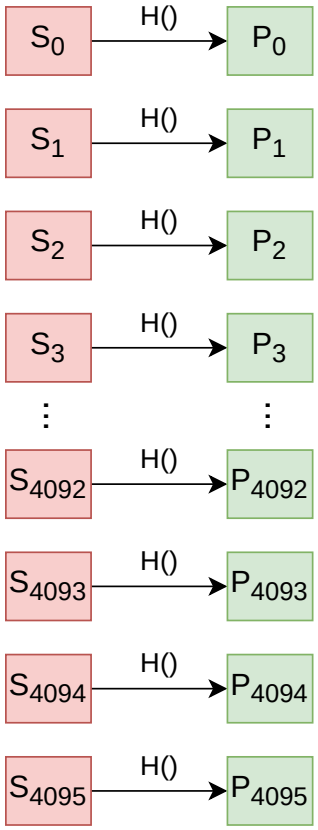


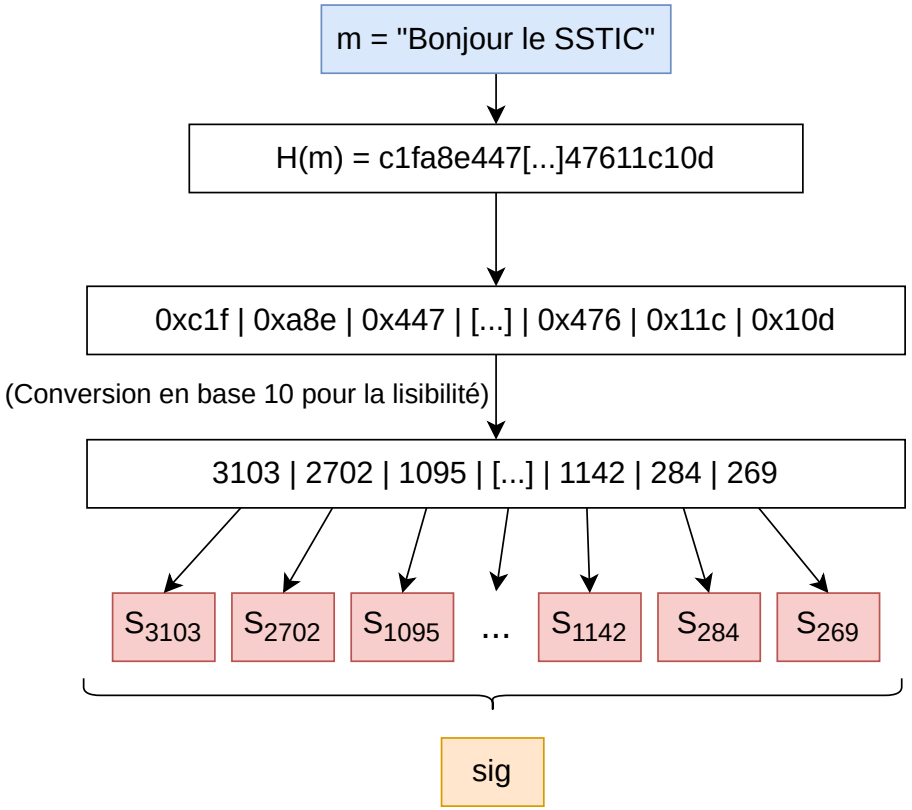


SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

- **Beaucoup** de paires « secret / public »
- Révélation d'**une petite partie** des secrets pour signer
- **Forger** une signature = trouver un message engendrant **le même subset**

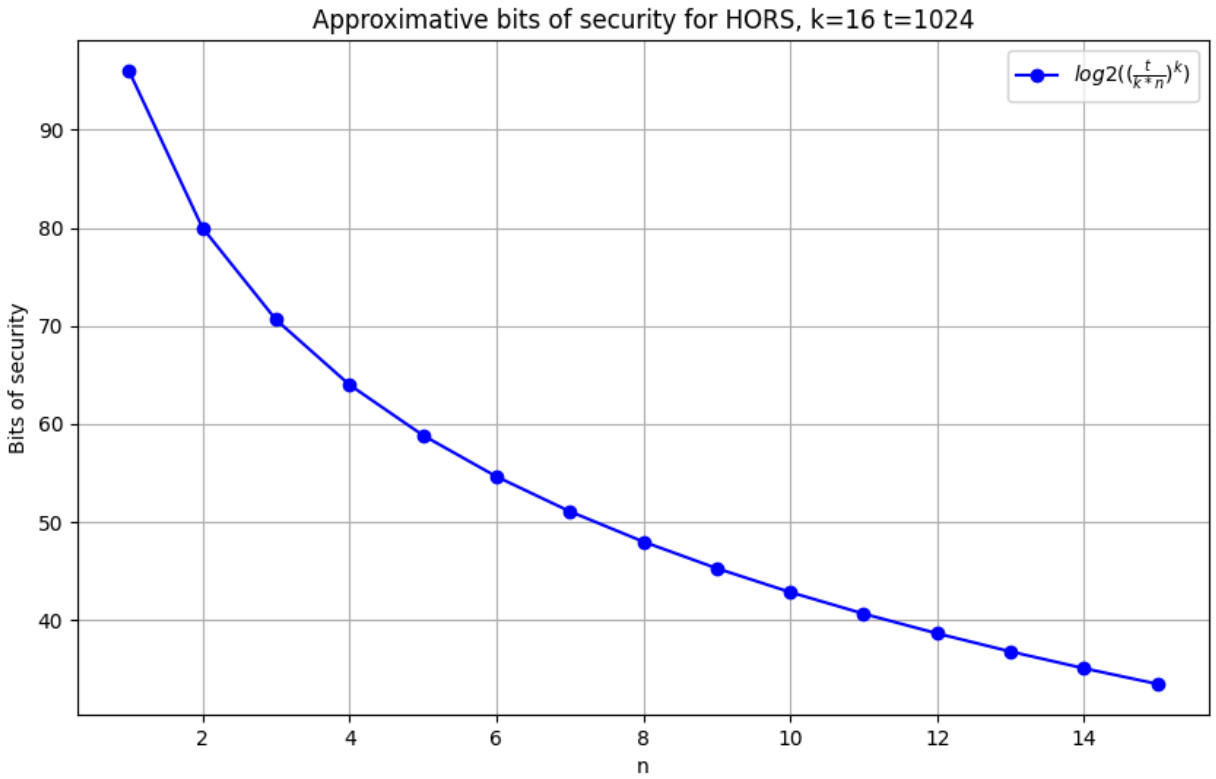




SLH-DSA

Étape 3 – HORS, Hash to Obtain a Random Subset

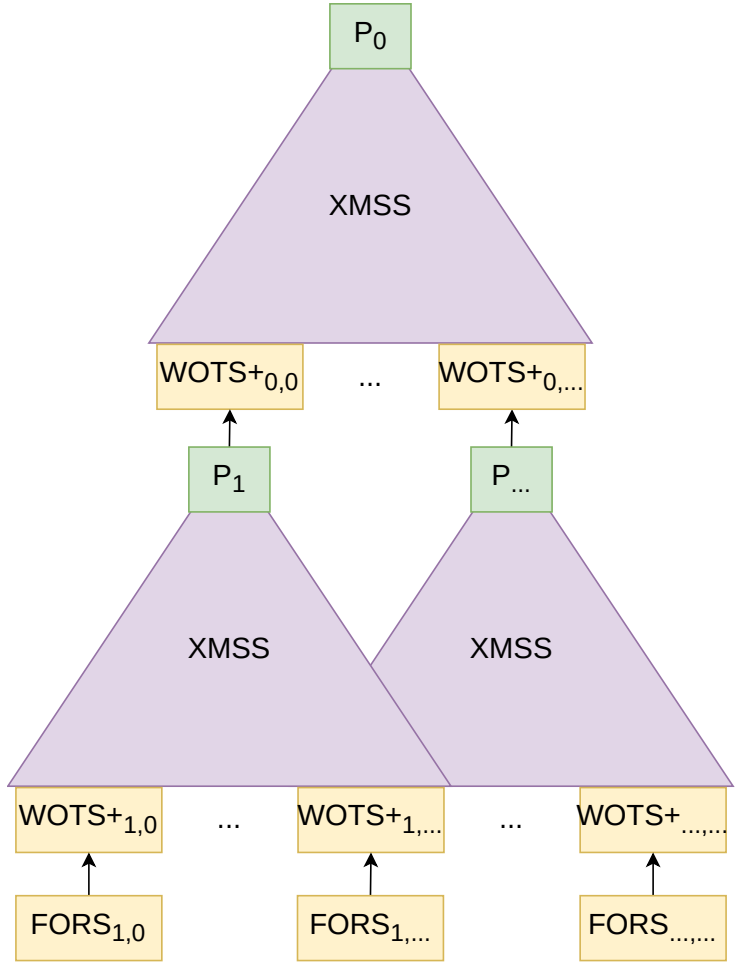
- **Après une signature, il reste « difficile » d'en forger une autre**



SLH-DSA

Ensemble

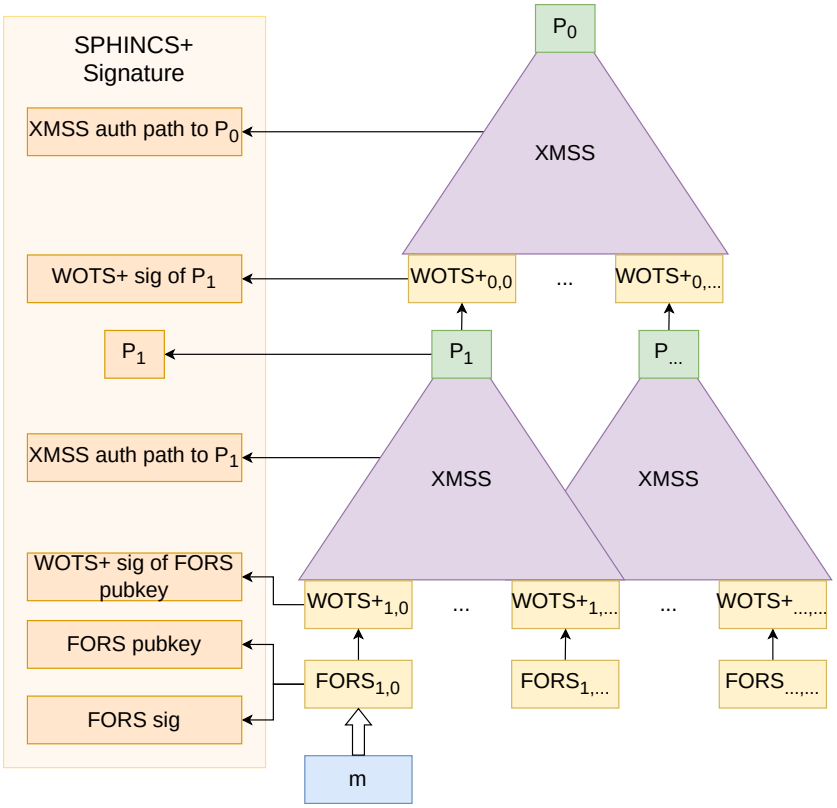
- Arbre d'arbres de Merkle
- Tellement de clés FORS (~2^{64}) que c'est considéré « stateless »



SLH-DSA

Ensemble

- **Signature**
  - Signature FORS du message
  - Signature WOTS de la clé publique FORS
  - Signature XMSS de la clé publique WOTS précédente
  - Signature WOTS de la clé publique XMSS précédente
  - Signature XMSS de la clé publique WOTS précédente
  - … jusqu'à retomber sur la signature donnant P_0



SLH-DSA

Conclusion

- **Algorithme *pas si complexe***
- **Quantique ? Pas vu passer…**
- **Empilement de briques connues**
- **Audit classique**
  - Vecteurs de test
  - Générateur d'aléa
  - Canaux auxiliaires

Les échanges de clés

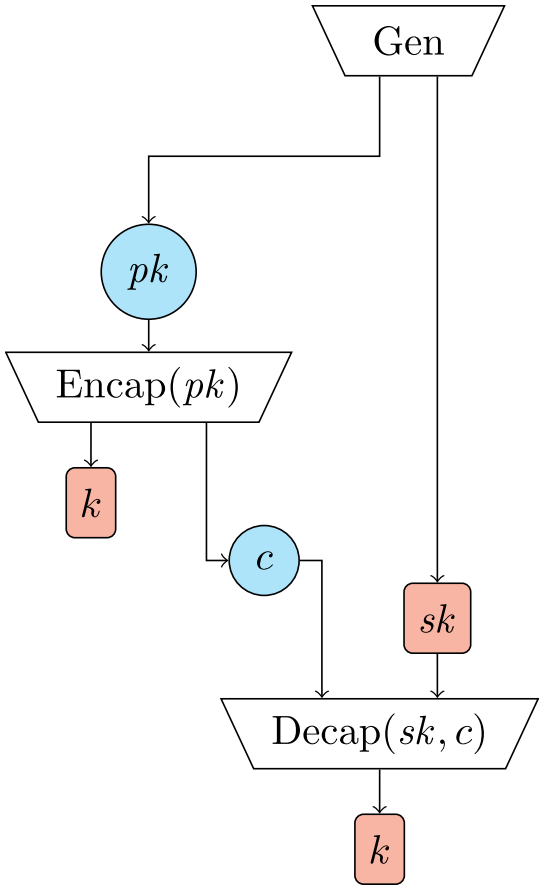
Kyber

Échanges de clés

Mécanisme d'encapsulation de clés

SYNACKTIV

- **"KEM"** : Key Encapsulation Mechanism, généralise l'échange de clés
  - **KeyGen()** $\rightarrow (\boldsymbol{pk}, \boldsymbol{sk})$
  - **Encaps($\boldsymbol{pk}$)** $\rightarrow (\boldsymbol{k}, \boldsymbol{caps})$, $\boldsymbol{k}$: shared secret
  - **Decaps($\boldsymbol{sk}, \boldsymbol{caps}$)** $\rightarrow \boldsymbol{k}$
- **Alice** utilise **Encaps()** avec la clé publique de Bob
- **Bob** « décapsule » (**Decaps()**) la capsule que lui envoie **Alice**
- Assimilable à un chiffrement asymétrique



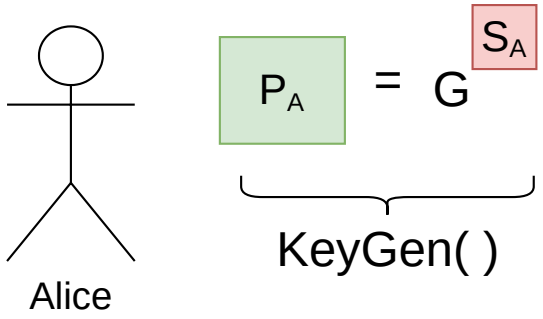
Échanges de clés

Mécanisme d'encapsulation de clés – Diffie-Hellman

Échanges de clés

Mécanisme d'encapsulation de clés – Diffie-Hellman

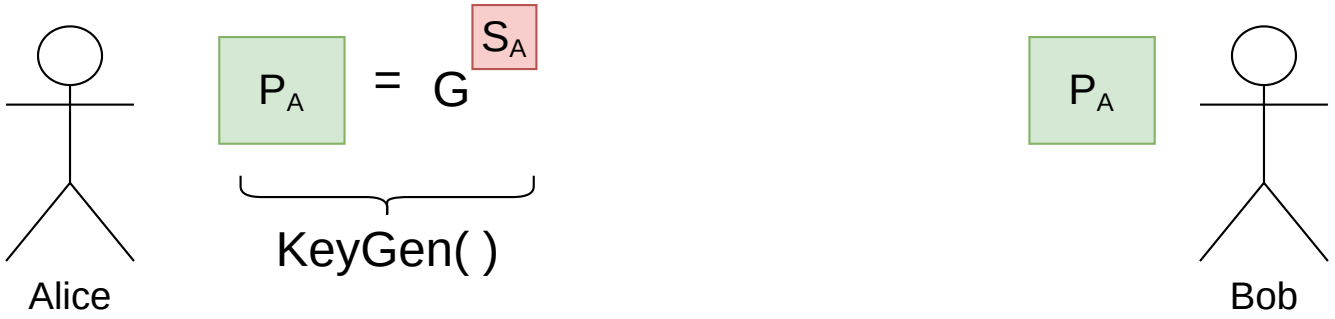
- **Exemple : Diffie-Hellman** (pas post-Q)



Échanges de clés

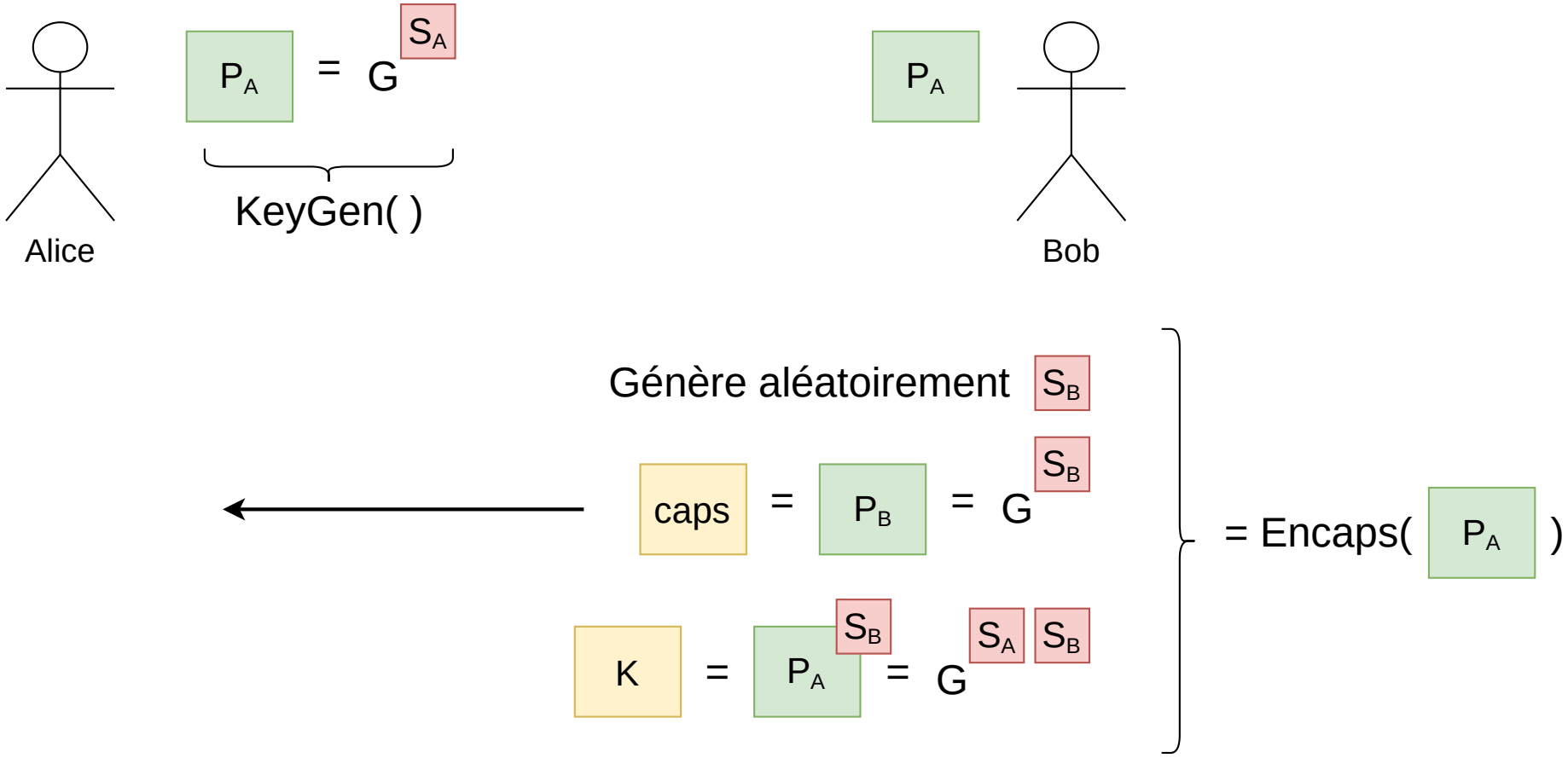
Mécanisme d'encapsulation de clés – Diffie-Hellman

- **Exemple : Diffie-Hellman** (pas post-Q)



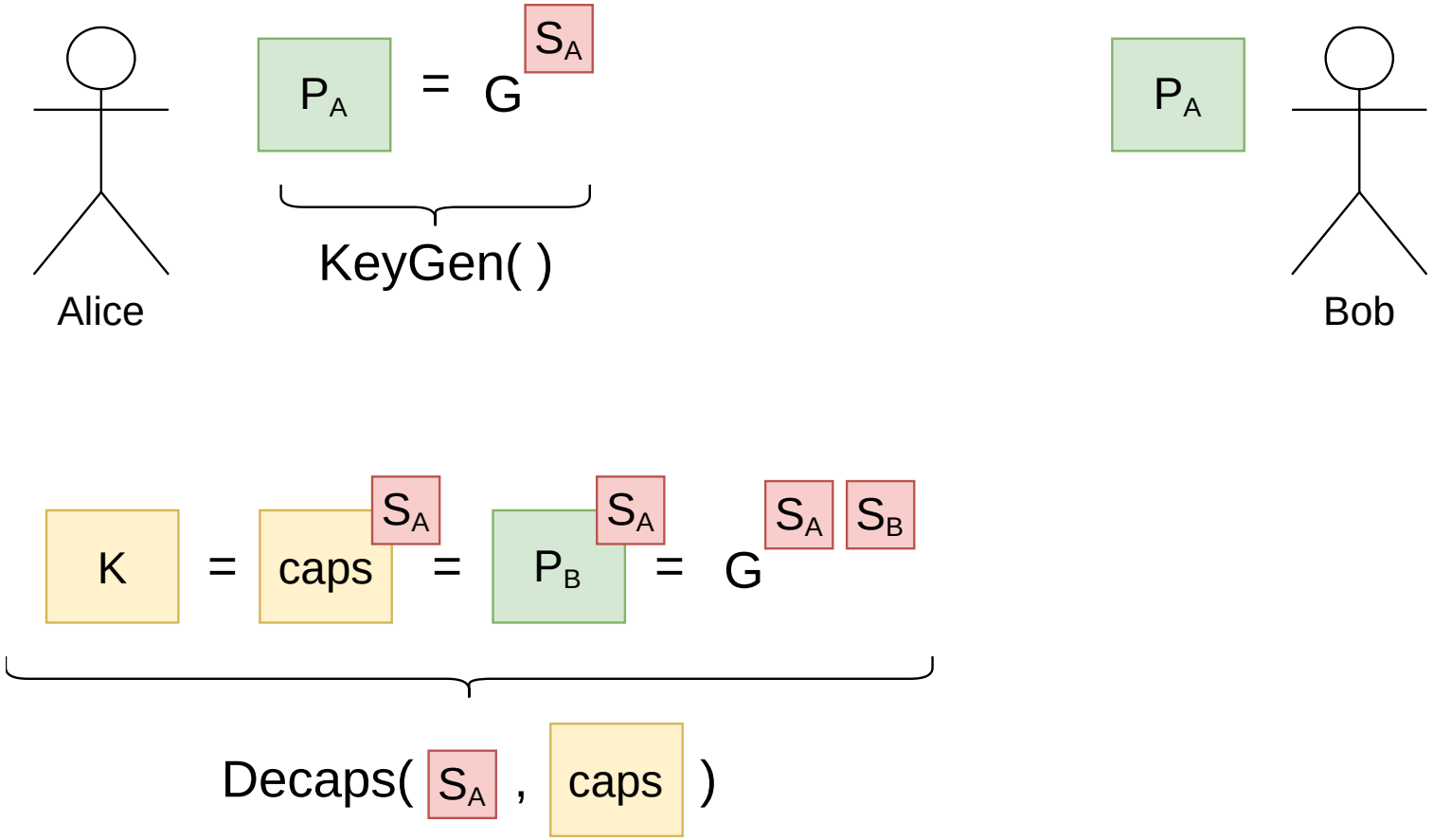
Échanges de clés

Mécanisme d'encapsulation de clés – Diffie-Hellman



Échanges de clés

Mécanisme d'encapsulation de clés – Diffie-Hellman



Kyber en une (une?) phrase

SYNACKTIV

- Problème sous-jacent : **Module-LWE** (« variante » du LWE)
- Paramètres ≈ niveaux NIST 1/3/5
  - Kyber-512 ≈ AES-128
  - Kyber-768 ≈ AES-192
  - Kyber-1024 ≈ AES-256

Implémentation crypto

Rappels

SYNACKTIV

- Pas de branchements dépendants d'un secret
- Pas d'accès mémoire dépendant d'un secret
- Pas de division en temps non-constant
- RNG solide

Implémentation crypto

SYNACKTIV

Case study : KyberSlash

- 2024 : publication de **KyberSlash 1 & KyberSlash 2**
- Cause : divisions dépendantes du secret dans de multiples implémentations
- Les divisions prennent un temps variable en fonction du secret
  - timing side-channel
- Exploit démontré :
  - Récupération de la clef **en quelques minutes** sur un RPI
- Correctif : remplacer les divisions par des opérations constantes ou des tables pré-calculées

```c
void poly_tomsg(uint8_t msg[32], const poly *a)
{
  unsigned int i,j;
  uint16_t t;
  for(i=0; i<KYBER_N/8; i++) {
    msg[i] = 0;
    for(j=0; j<8; j++) {
      t = a->coeffs[8*i+j];
      t += ((int16_t)t >> 15) & KYBER_Q;
      // Division par KYBER_Q (publique)
      // mais 't' est secret
      t = (((t << 1) + KYBER_Q/2)/KYBER_Q) & 1;
      msg[i] |= t << j;
    }
  }
}
```

Implémentation crypto

KyberSlash vs RSA (Kocher 1996)

SYNACKTIV

- **Racine commune :** des opérations dont la durée dépend de données **secrètes** → exposent un canal temporel exploitable.
- **KyberSlash (2024)** : divisions par `t` dans `poly_tomsg()` ; leur temps varie avec les coefficients secrets → clé récupérée en quelques minutes sur un Raspberry Pi.
- **RSA « square-and-multiply » (Kocher, 1996)** : chaque bit du secret décide (ou non) d'une multiplication supplémentaire ; la durée de l'exponentiation révèle la clé privée.

Implémentation crypto

À retenir

SYNACKTIV

- Les maths de Kyber restent solides
- Les vulnérabilités découvertes ressemblent à celles qu'on connaît déjà
- Un auditeur doit :
  1. Vérifier l'absence de dépendances secrètes (branches/mémoire/temps)
  2. Valider la génération aléatoire et la gestion d'erreurs
  3. Tester sur la *vraie* cible pour débusquer les fuites
- En bref : mêmes recettes qu'en crypto classique

L'hybridation

Hybridation

SYNACKTIV

- **Algorithmes post-quantiques « nouveaux », peu étudiés, vulnérables[1] ?**
- **Idéalement : sécurité pré-Q des algos classiques, post-Q des nouveaux algos**
  - *Mélanger* des algorithmes des deux familles pour obtenir une sécurité équivalente au meilleur d'entre eux

Famille	Confiance pré-Q	Confiance post-Q
Classiques (Courbes elliptiques, DH, ...)		
Nouveaux (Lattices, Codes, Hashs, ...)		
Meilleur des mondes		

1: Penser au cas de Supersingular Isogeny Diffie Hellman, cassé récemment (en 2022) après avoir passé les premiers rounds du NIST

Hybridation

Signatures

SYNACKTIV

- Le cas simple, on concatène les signatures
- Signature valide si et seulement si toutes les sous-signatures sont valides
  - KeyGen$_{hyb}$() $\rightarrow S = (S_{class}, S_{new}), P = (P_{class}, P_{new})$
  - Sign$_{hyb}$(m, S) $\rightarrow sig = (sig_{class}, sig_{new})$
  - Verify$_{hyb}$(m, sig, P) $\rightarrow isValid = isValid_{class} \ \& \ isValid_{new}$

Hybridation

Échanges de clés

SYNACKTIV

- **Concaténer les secrets partagés des KEM**
  - Si un KEM est cassé, secret partiellement compromis
  - **→ NOK** ☹️
- **Hacher la concaténation des secrets (avec une KDF)**
  - Si un KEM ne respecte pas certaines propriétés, possibilité d'affaiblir le secret [1]
  - **→ NOK** ☹️
- **Hacher la concaténation des secrets et des capsules (avec une KDF)**
  - **→ Solution prouvée valide** [2] ✅
  - … mais overhead sur la taille des entrées à hacher
- **Beaucoup de solutions *artisanales*, standardisation en cours**[3]

1: KEM Combiners, https://eprint.iacr.org/2018/024
2: CatKDF, https://eprint.iacr.org/2022/065.pdf
3: Composite ML-KEM en draft à l'IETF, probablement des structures similaires pour d'autres KEM

Conclusion

Conclusion

SYNACKTIV

- C'est le moment de s'y mettre
- Allez au-delà du cliché *post-quantique = compliqué* !
- Conférences ET lecture de papiers pour se former simplement [1]

1: Ressources disponibles en bibliographie des actes

Questions ?

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://x.com/synacktiv

https://bsky.app/profile/synacktiv.com

https://synacktiv.com