

論文“Breaking Category Five SPHINCS+ with SHA-256” の紹介

縫田 光司 (NUIDA, Koji)

九州大学 マス・フォア・インダストリ研究所
／産業技術総合研究所

SCAIS 2023 @小倉 2023年1月23日

概要

- Ray Perlner, John Kelsey, and David Cooper:
“Breaking Category Five SPHINCS+ with SHA-256”,
PQCrypto 2022 の紹介
 - ハッシュベース署名SPHINCS+ (w/ SHA-256) の
NIST category 5 (≒256ビット古典安全性) パラメータを
約217.4ビット安全性に低下させた
 - Round 3 official comment (2022.6.10)で対応済みとのこと
- SPHINCS+の構成 (の概要) の説明

目次

- SPHINCS⁺の構成の概要
- 攻撃論文 (@PQCrypto 2022) の紹介

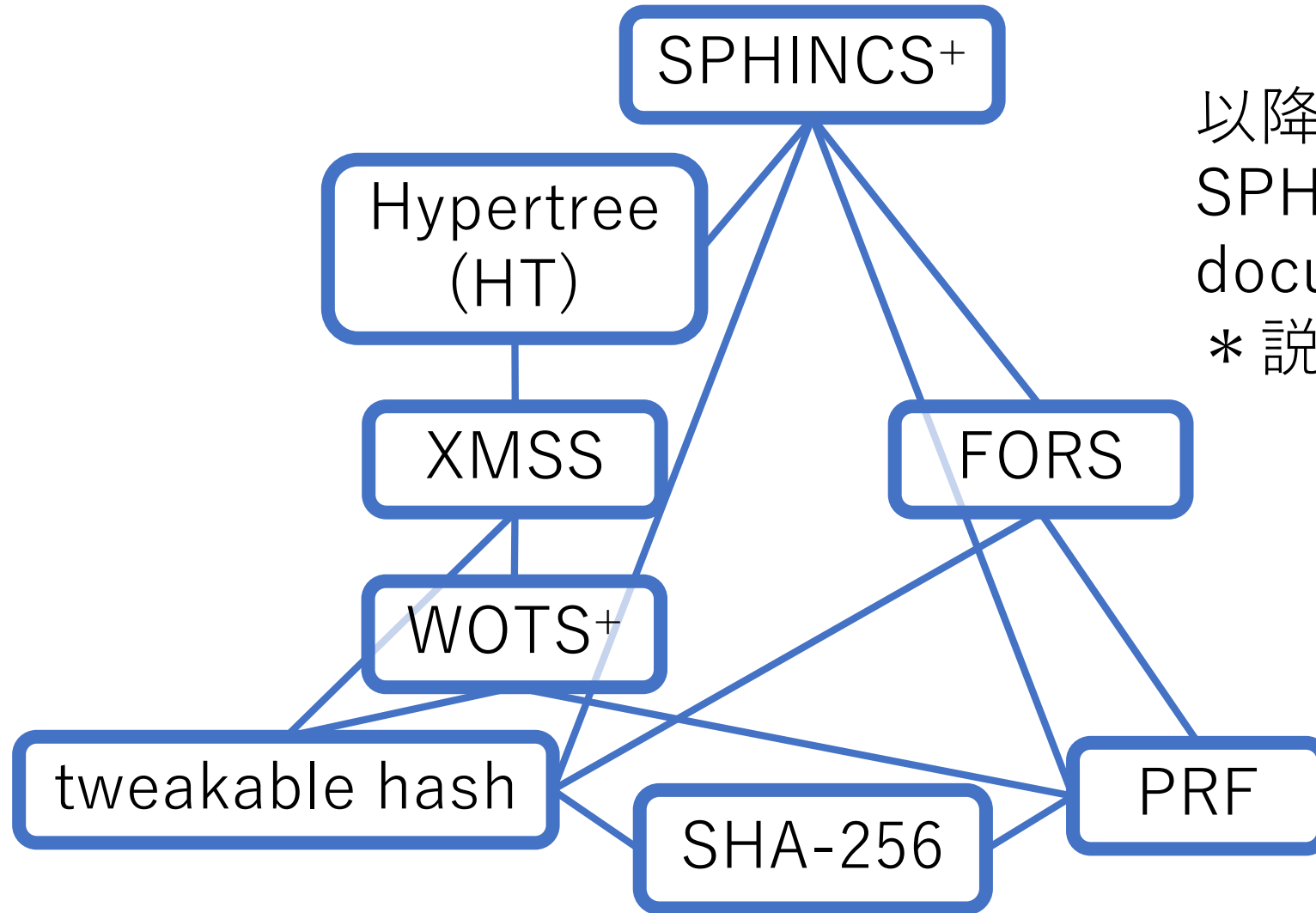
目次

- SPHINCS⁺の構成の概要
- 攻撃論文 (@PQCrypto 2022) の紹介

SPHINCS+

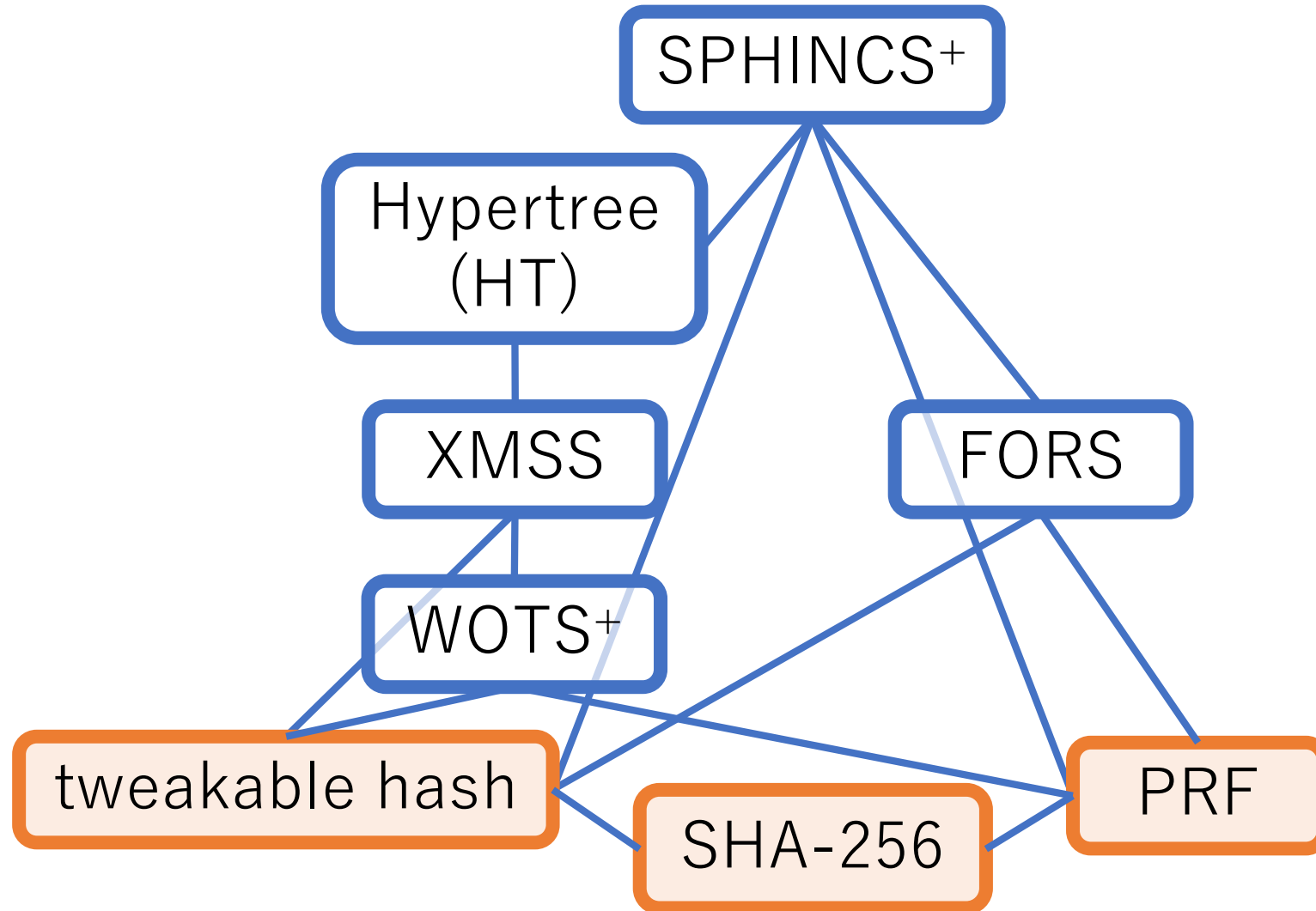
- 署名方式SPHINCS : Bernstein, Hopwood, Hulsing, Lange, Niederhagen, Papachristodoulou, Schneider, Schwabe, Wilcox-O'Hearn, EUROCRYPT 2015
- ハッシュベース署名
 - (Round 3当時は) SHAKE-256, SHA-256, Harakaに対応
 - **今回はSHA-256ベースの構成を扱う (攻撃対象なので)**
- SPHINCS+ : NIST PQC標準に選定 (2022.7) された署名方式の一つ (他の2方式はどちらも格子ベース)

SPHINCS+の構成要素と依存関係



以降の構成の説明は
SPHINCS+のRound 3
documentを参照
* 説明用に一部簡略化

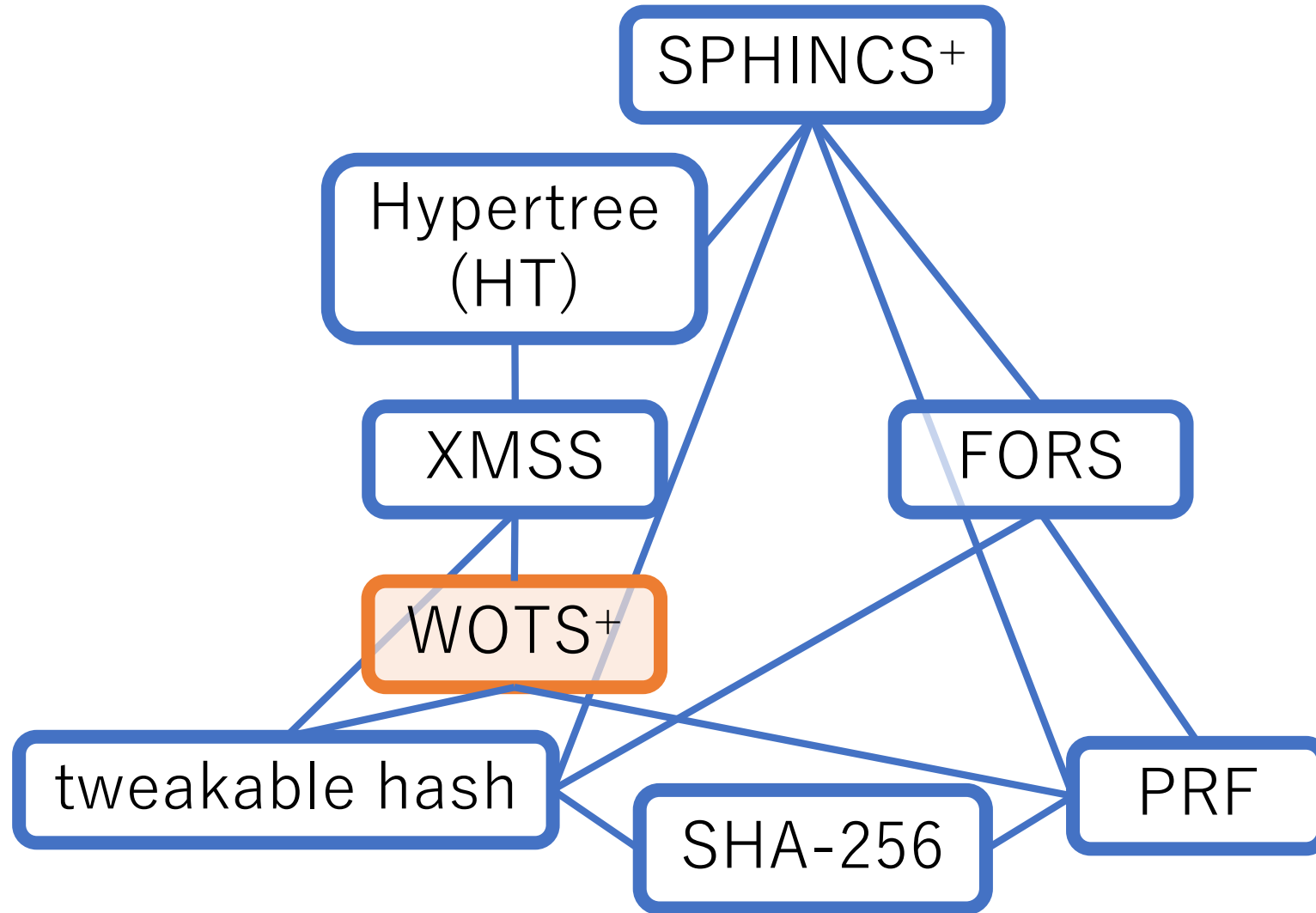
SPHINCS+の構成要素と依存関係



Tweakable HashとPRF

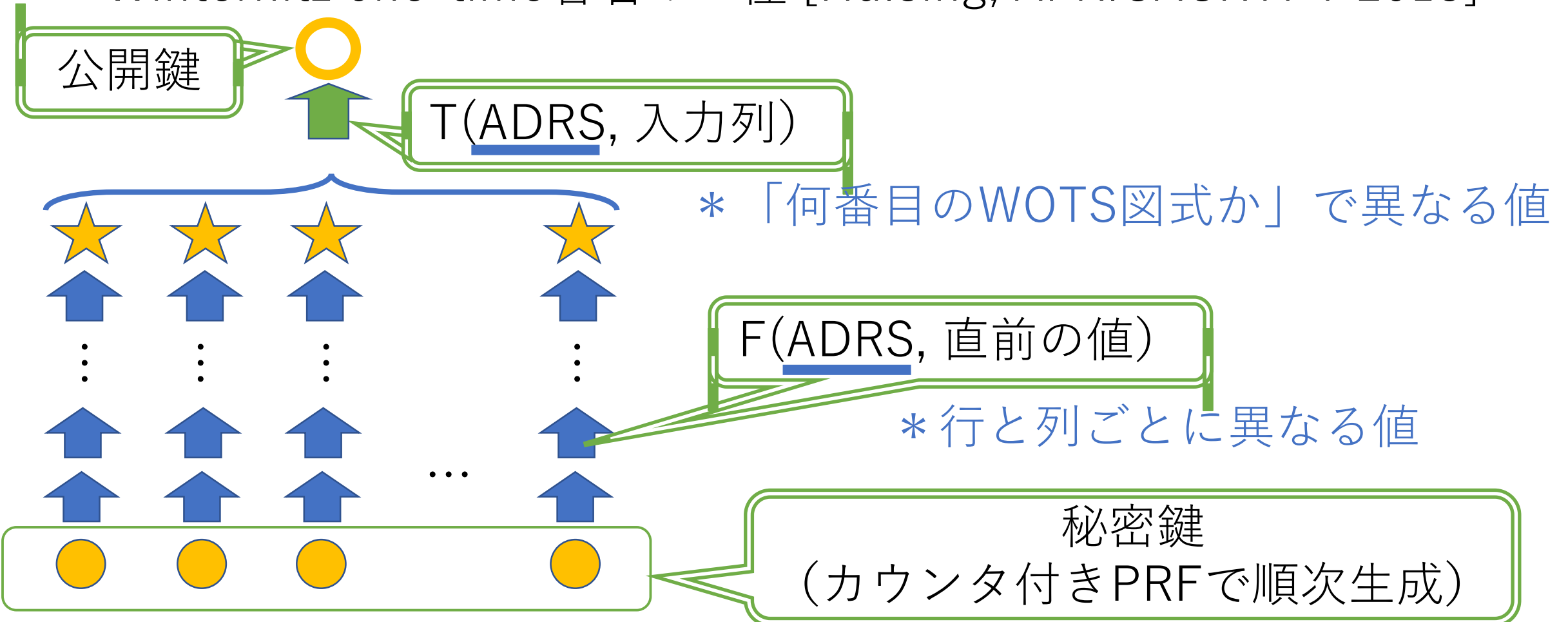
- F, H, T: “tweakable”ハッシュ関数
 - (公開seedと) 固定長メッセージ M と **補助入力 ADRS**
 - ハッシュ関数族とみたとき、量子 (多項式) 攻撃者に対して **distinct-function multi-target second-preimage 耐性**:
一様ランダムなメッセージ M_1, \dots, M_p に対して、
 $M' \neq M_j$ かつ $\text{Hash}_j(M') = \text{Hash}_j(M_j)$ を満たす (j, M') が
見つかる確率が negligible
 - Fについては追加の仮定が必要 (割愛)
- 疑似ランダム関数PRFにも (耐量子的) 仮定 (割愛)
- どれもハッシュ関数 (SHA-256) から作られる

SPHINCS+の構成要素と依存関係



WOTS+

Winternitz one-time署名の一種 [Hulsing, AFRICACRYPT 2013]



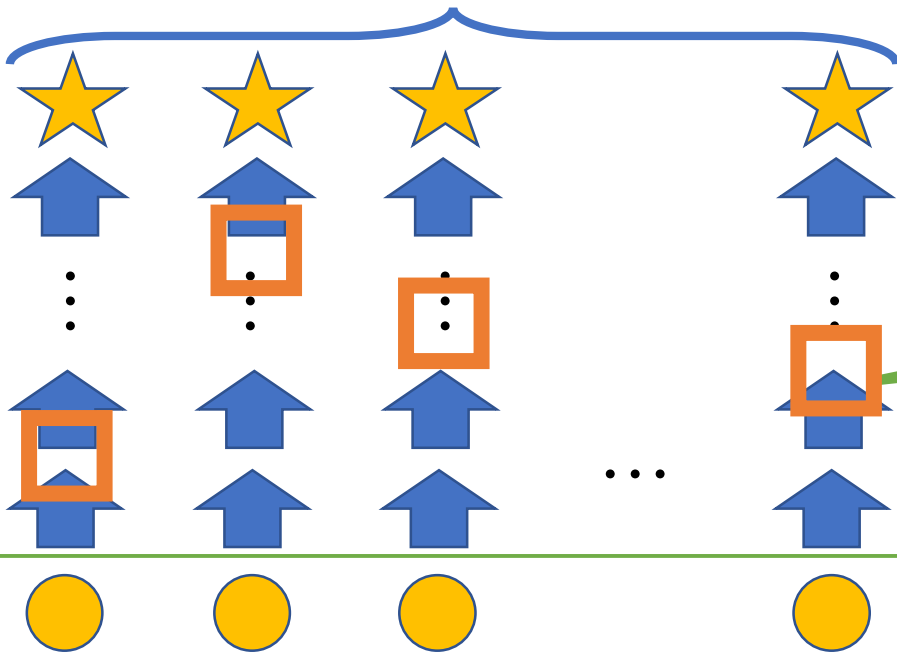
WOTS+

Winternitz one-time署名の一種 [Hulsing, AFRICACRYPT 2013]

公開鍵



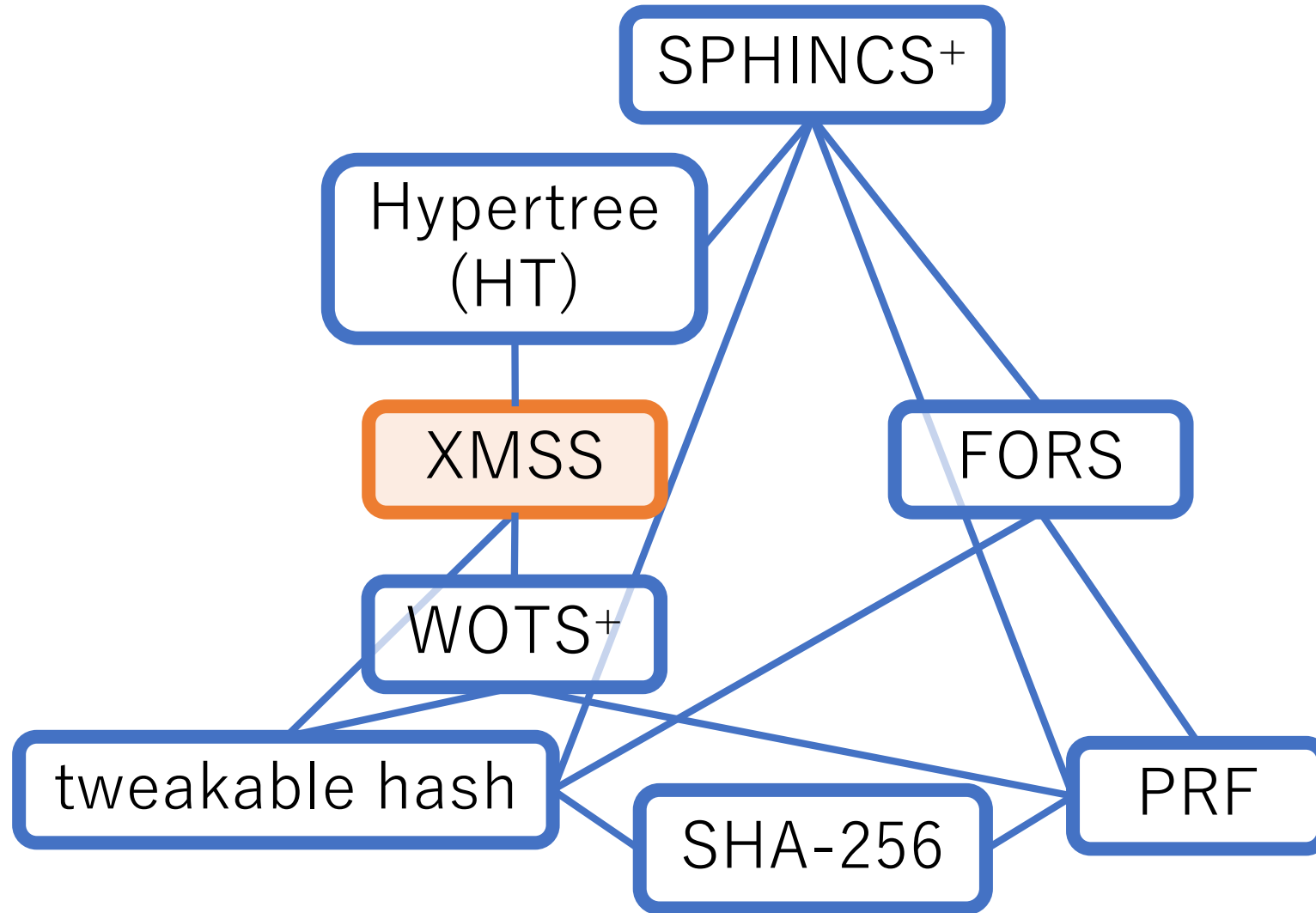
* 署名検証：一番上まで計算して、 =  か確認



位置（高さ）を入力から決める
（PRF等はいない）
→ これらの位置の値の組が署名

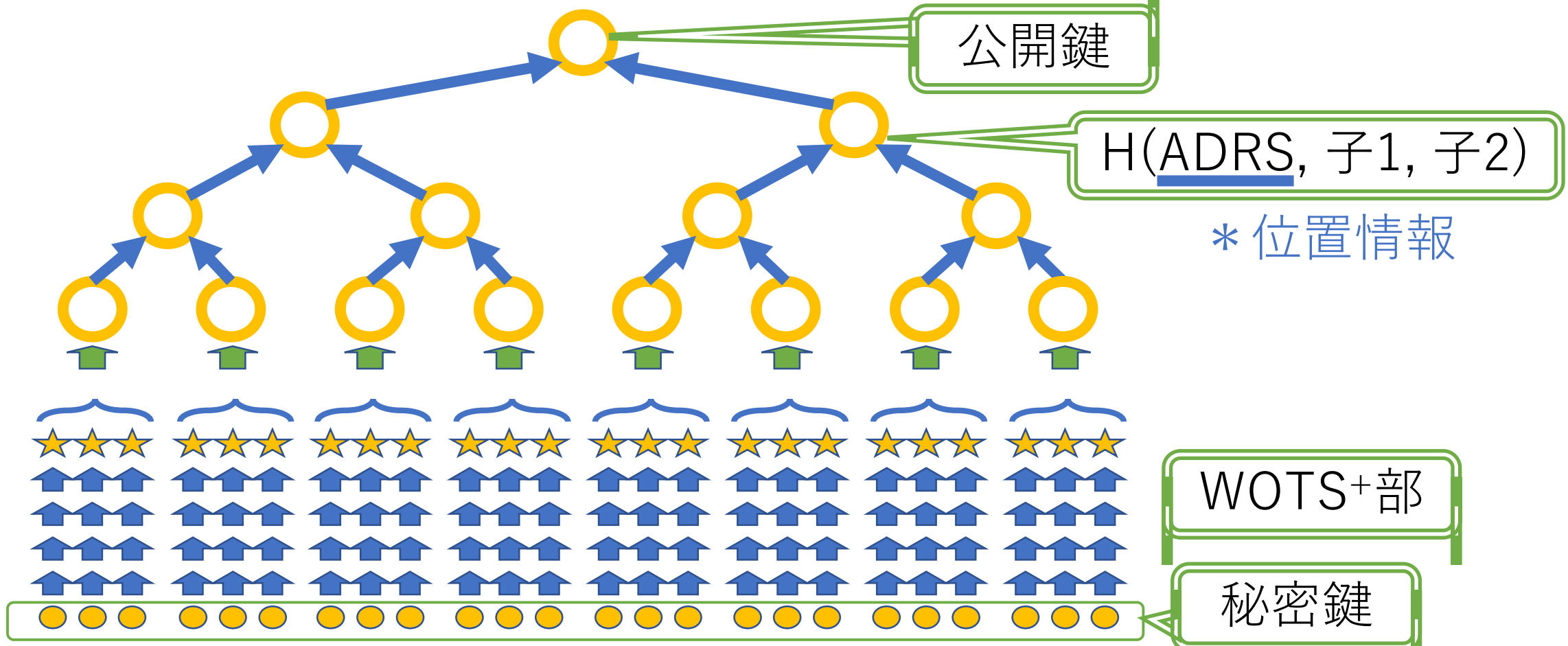
秘密鍵
（カウンタ付きPRFで順次生成）

SPHINCS+の構成要素と依存関係



(固定入力長) XMSS

eXtended Merkle Signature Scheme [Buchmann et al., PQCrypto 2011]

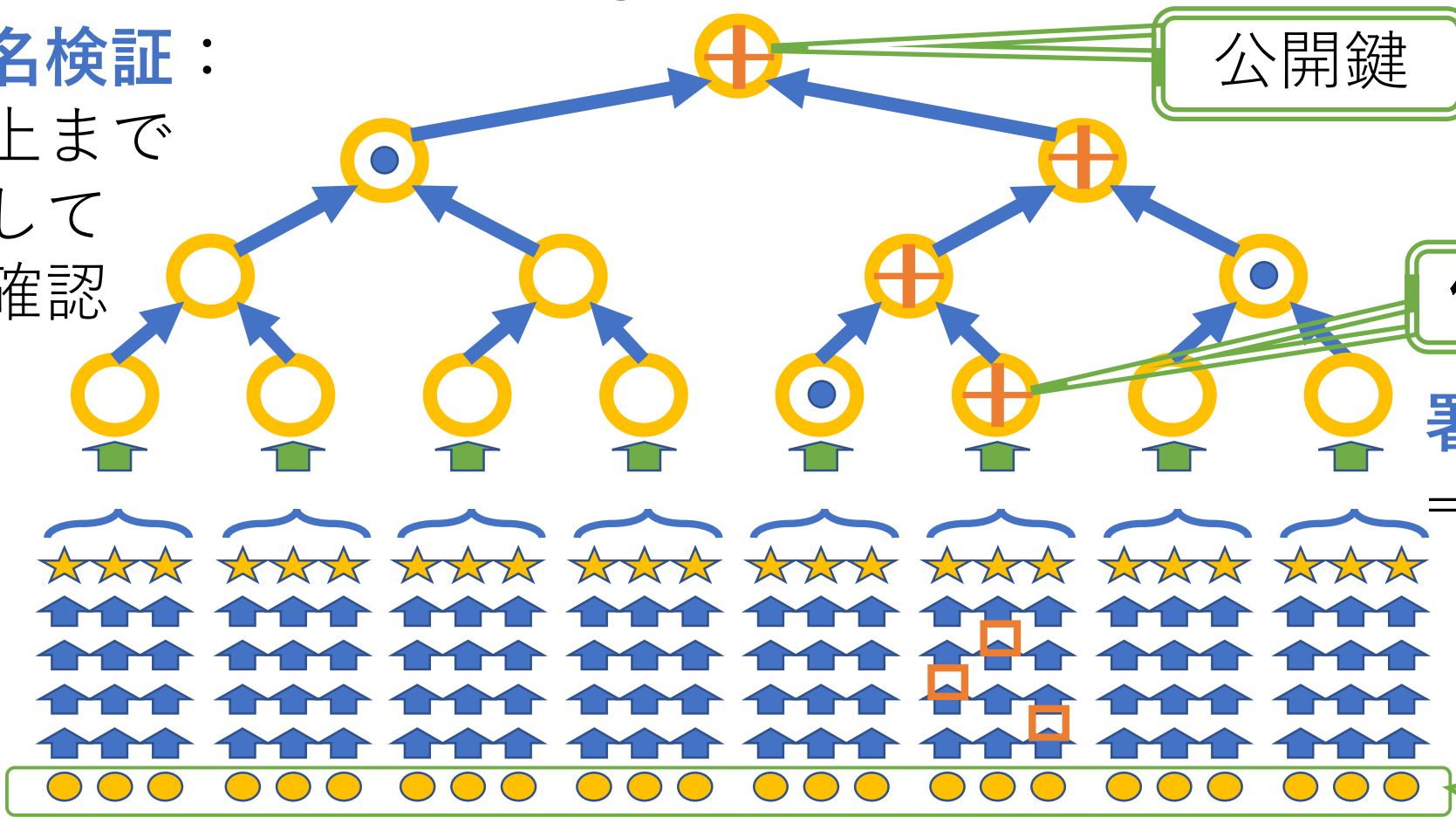


(固定入力長) XMSS

eXtended Merkle Signature Scheme [Buchmann et al, PQCrypto 2011]

* **署名検証** :

一番上まで
計算して
一致確認



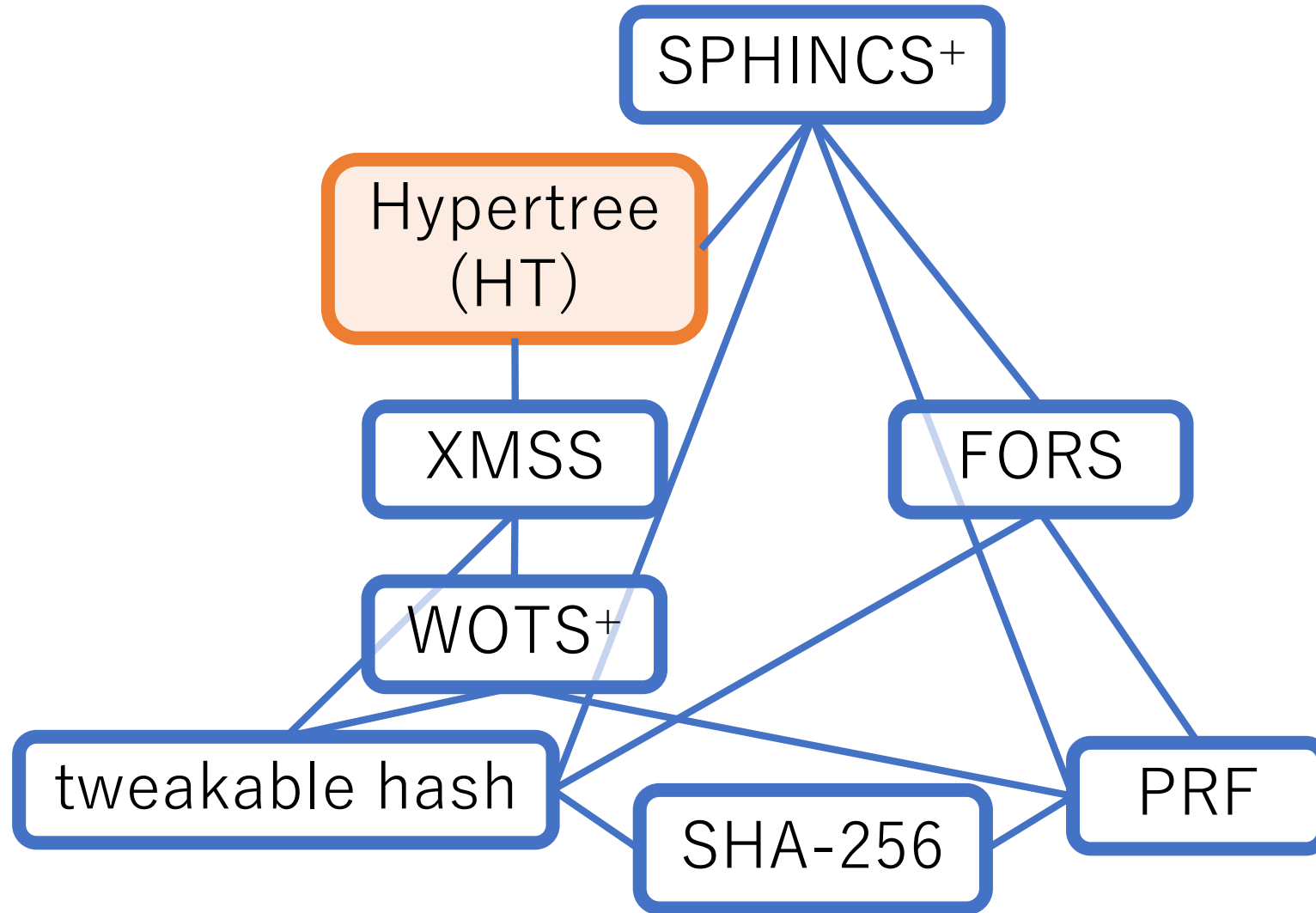
使う葉を別途指定

署名
= { □たち, ●たち }

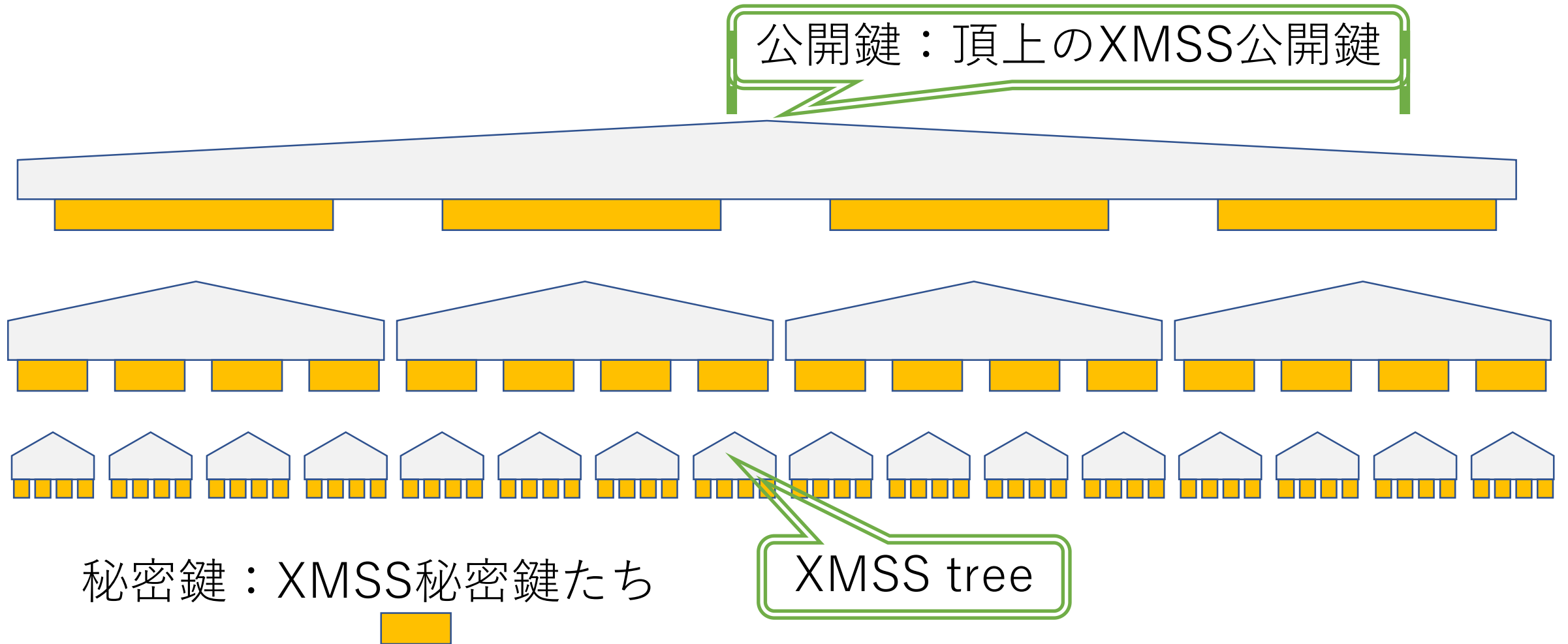
* WOTS+
による署名

秘密鍵

SPHINCS+の構成要素と依存関係



Hypertree (HT)

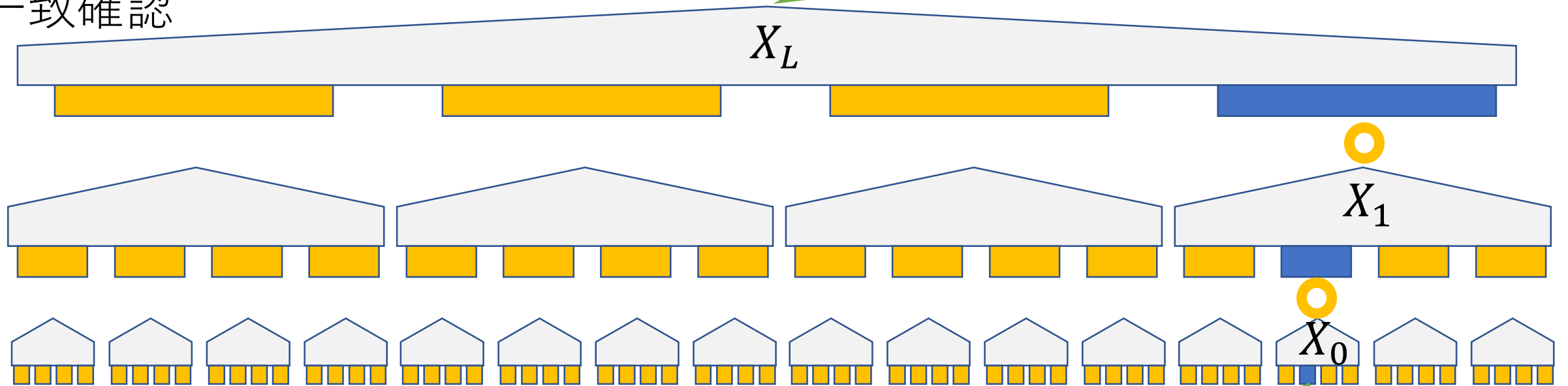


Hypertree (HT)

署名 : $(\sigma_i)_{i=0, \dots, L}$

署名検証 : 一番上まで計算して
一致確認

公開鍵 : 頂上のXMSS公開鍵

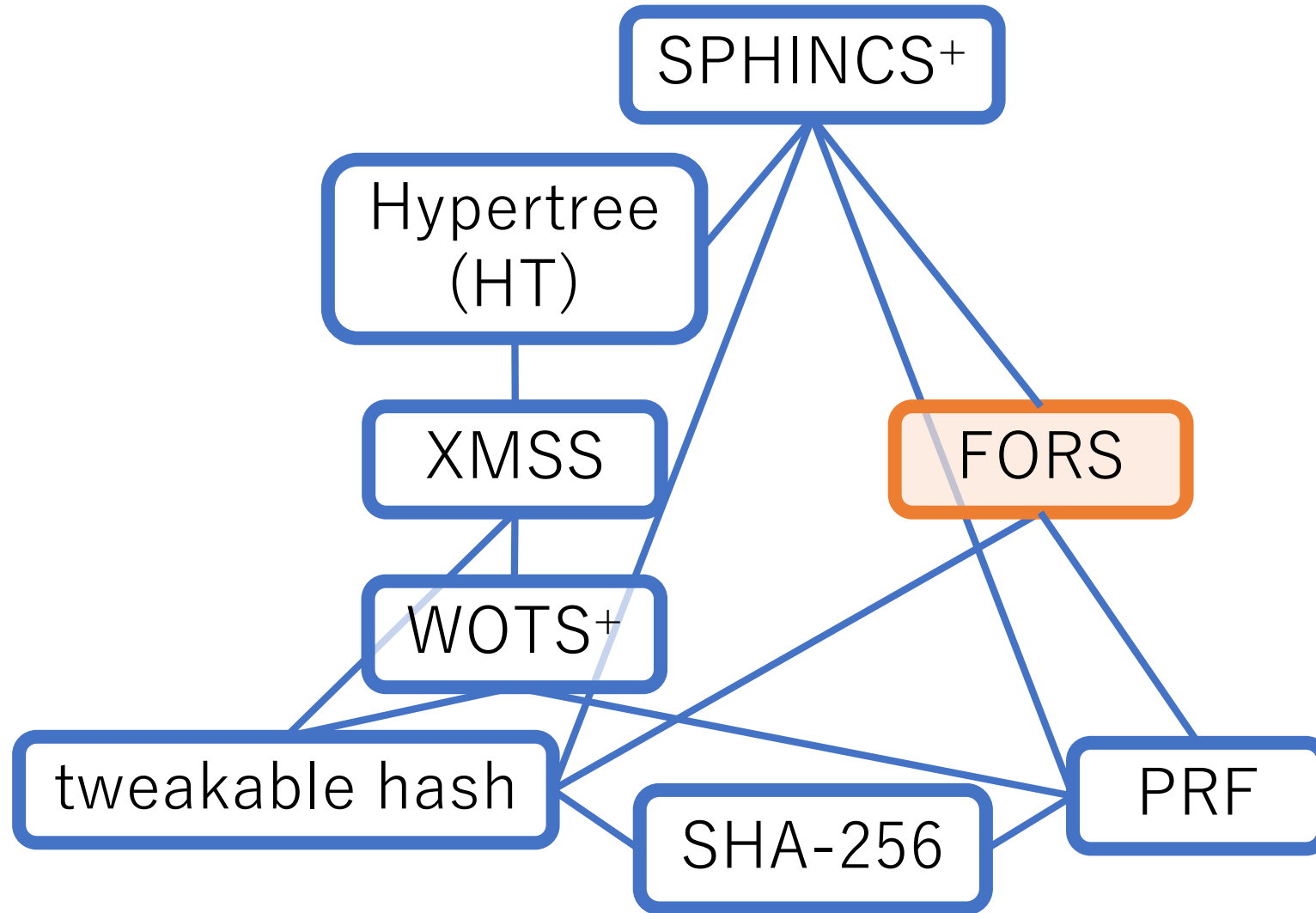


σ_0 : メッセージの X_0 での署名

σ_i : X_{i-1} の公開鍵の X_i での署名 ($i > 0$)

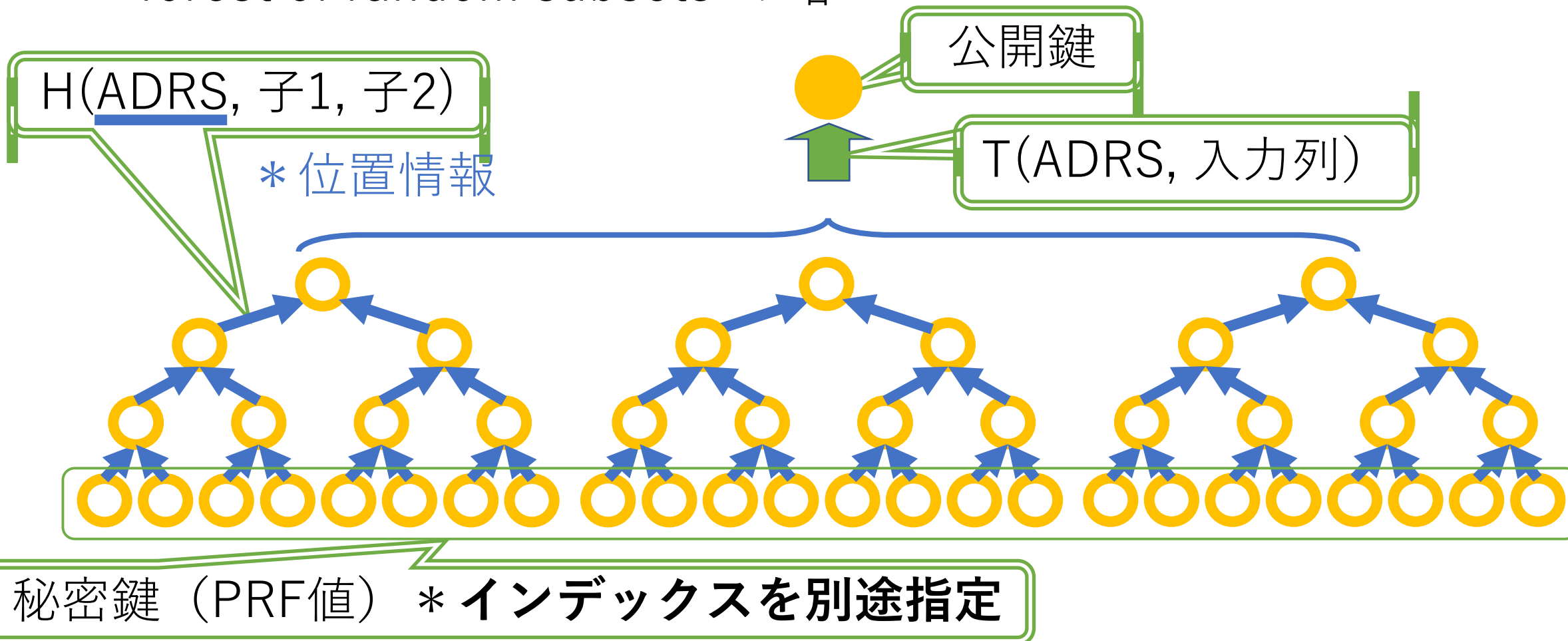
場所を別途指定

SPHINCS+の構成要素と依存関係





FORS [fɔ:rs]

“forest of random subsets”の略



FORS [fɔ:rs]

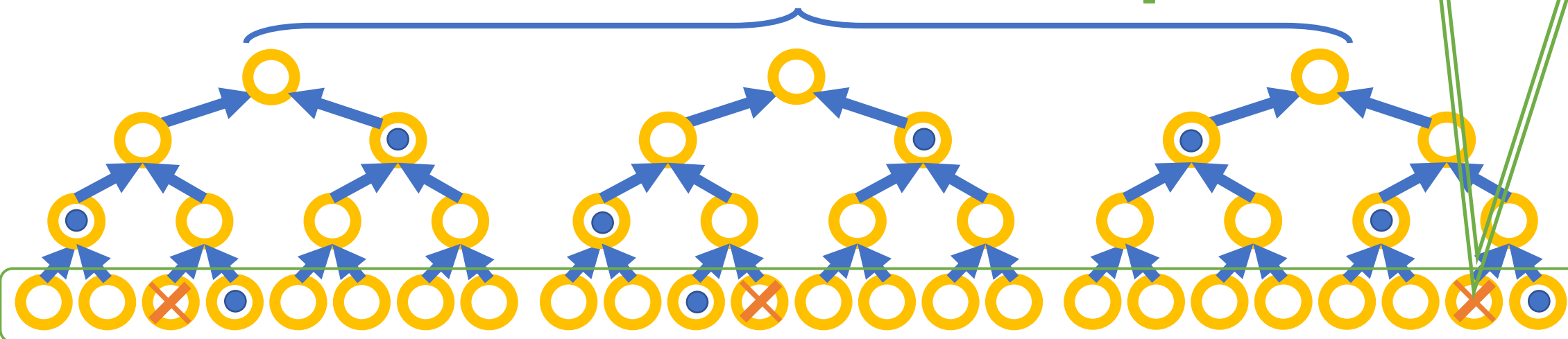
“forest of random subsets”の略

署名：{たち, たち}

署名検証：一番上まで計算して
一致確認

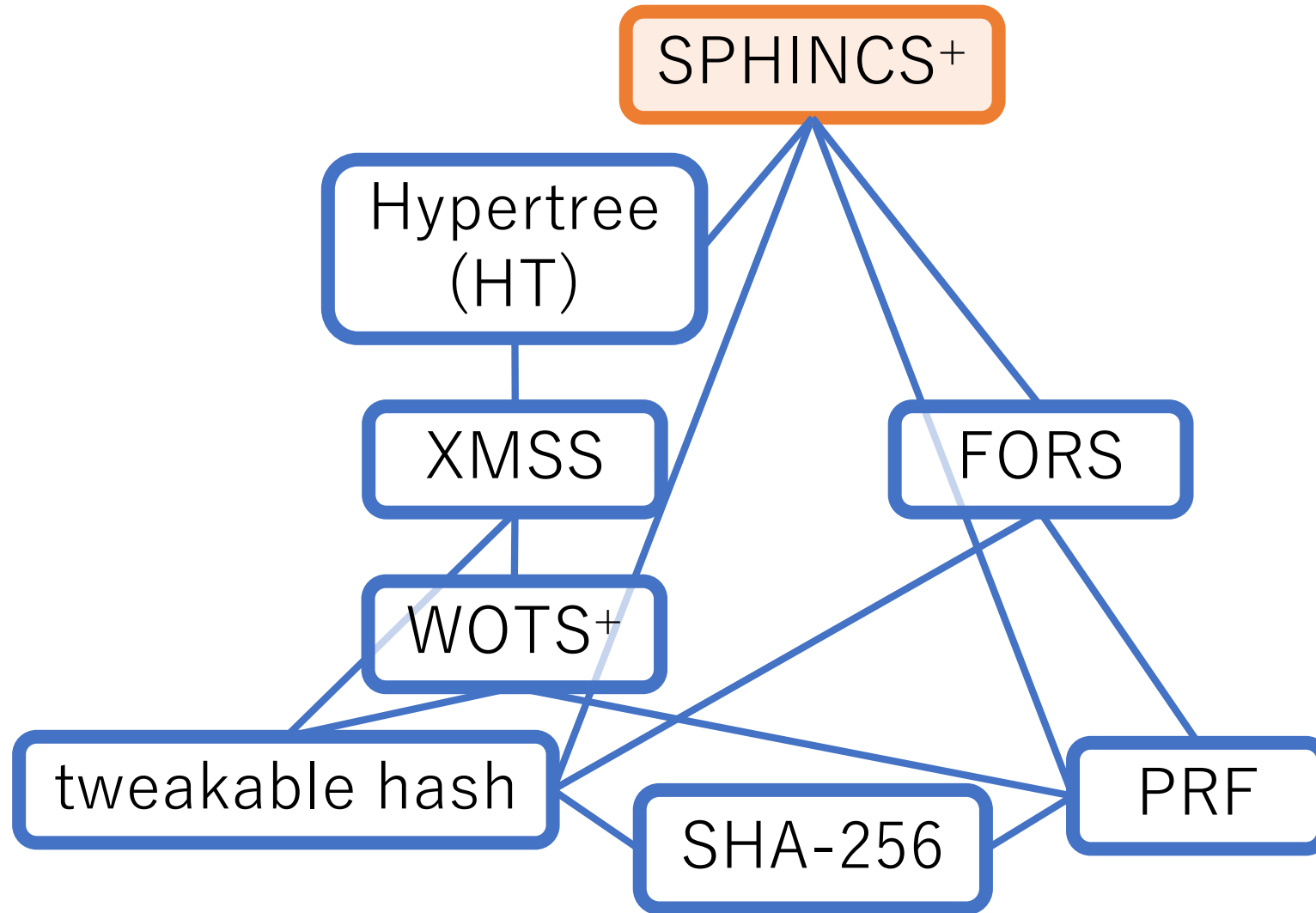
公開鍵

各木の  の位置を
入力から決める



秘密鍵 (PRF値) * インデックスを別途指定

SPHINCS+の構成要素と依存関係



SPHINCS+ (概要)

- HTとFORSの特徴：
 - seedとindexを指定すると秘密鍵（と公開鍵）が決まる
 - メッセージと署名があると公開鍵を計算できる
 - 署名検証：公開鍵（の候補）を計算して一致判定
- SPHINCS+秘密鍵：seed、公開鍵：HTの公開鍵
- $\text{Sign}(M, sk)$: MからHTとFORSのindexを決めて、Mのハッシュ（w/ 乱数）値をFORSで署名、対応するFORS公開鍵をHTで署名 → その署名の組（+乱数）を出力
 - Verify: FORS公開鍵（候補）を計算してHTで署名検証

目次

- SPHINCS+の構成の概要
- 攻撃論文 (@PQCrypto 2022) の紹介

Tweakable HashとPRF

- F, H, T : “tweakable”ハッシュ関数
 - (公開seedと) 固定長メッセージ M と **補助入力 ADRS**
 - ハッシュ関数族とみたとき、量子 (多項式) 攻撃者に対して **distinct-function multi-target second-preimage 耐性**:
一様ランダムなメッセージ M_1, \dots, M_p に対して、
 $M' \neq M_j$ かつ $\text{Hash}_j(M') = \text{Hash}_j(M_j)$ を満たす (j, M') が
見つかる確率が negligible
 - F については追加の仮定が必要 (割愛)
- 疑似ランダム関数PRFにも (耐量子的) 仮定 (割愛)
- どれもハッシュ関数 (SHA-256) から作られる

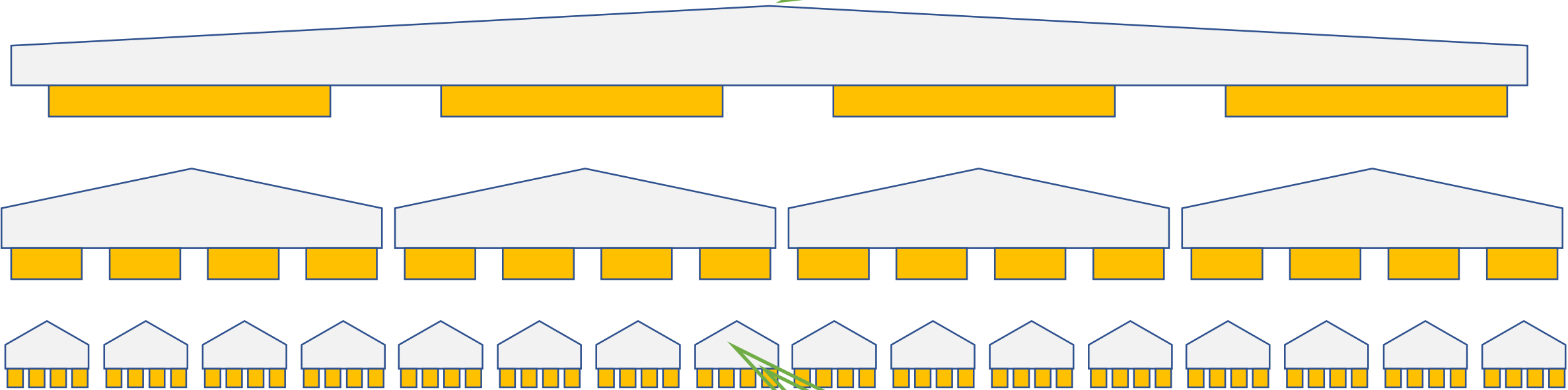
攻撃の背景と概要

- Sydney Antonov: Round 3 official comment (2022)
 - **SPHINCS+ w/ SHA-256のハッシュはDM-SPRでない**
- Ray Perlner, John Kelsey, and David Cooper:
“Breaking Category Five SPHINCS+ with SHA-256”,
PQCrypto 2022
 - tweakable hashに対するAntonovの攻撃を、
SPHINCS+に対する実際の署名偽造攻撃に昇華
- 以下、攻撃の大まかな流れを説明

再掲

Hypertree (HT)

公開鍵：頂上のXMSS公開鍵



秘密鍵：XMSS秘密鍵たち

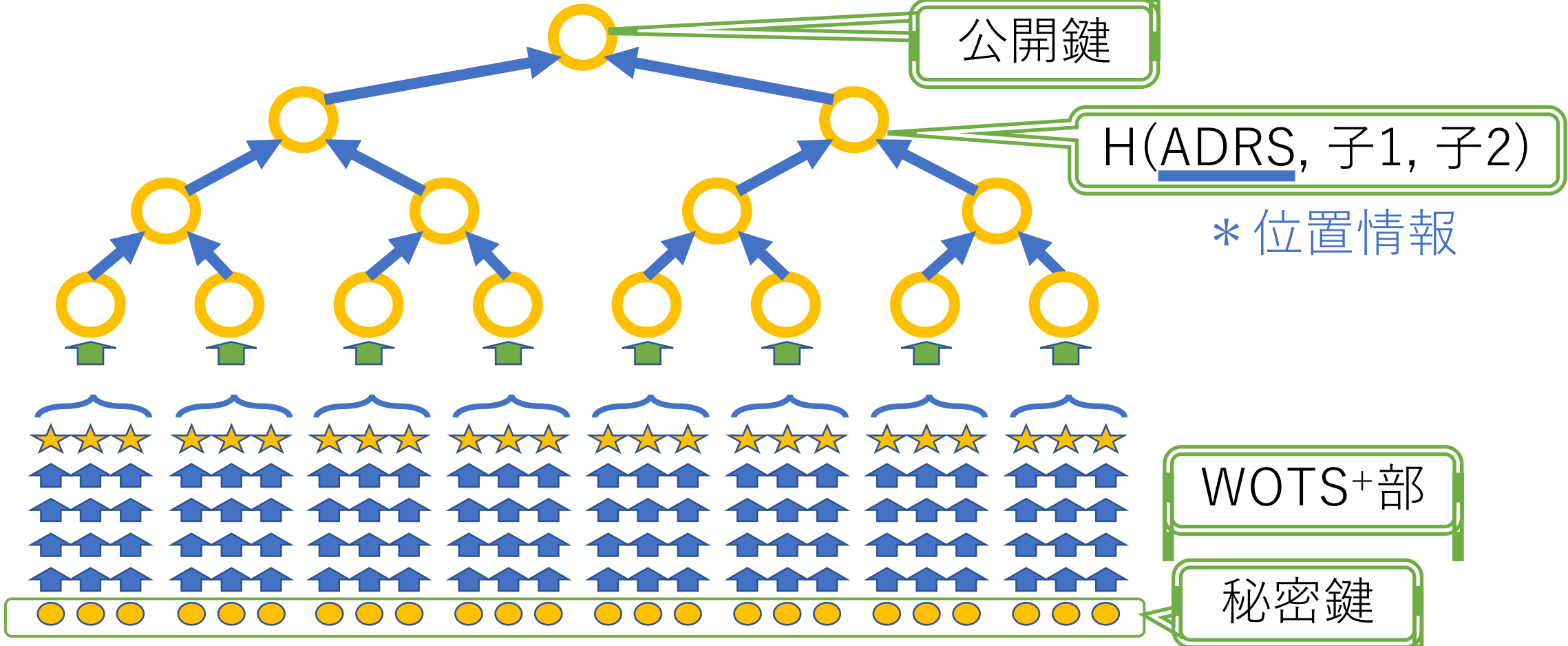


XMSS tree

再掲

(固定入力長) XMSS

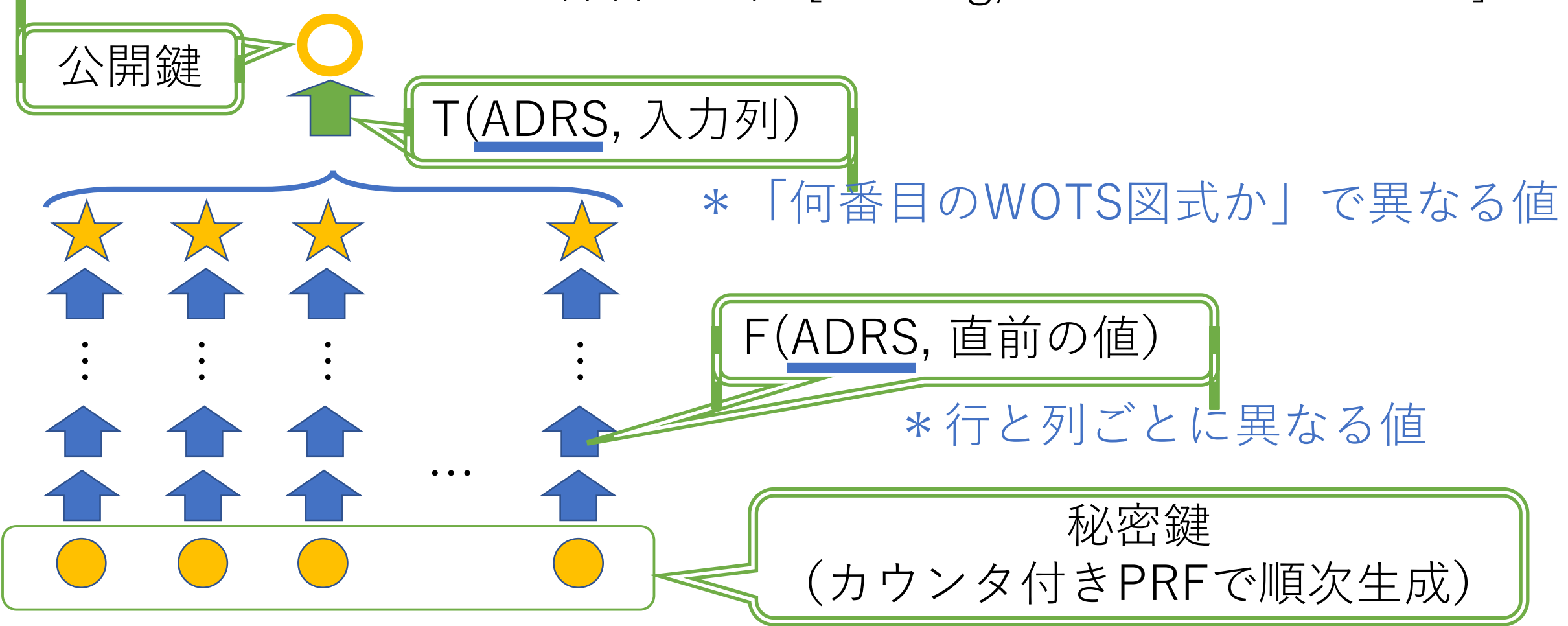
eXtended Merkle Signature Scheme [Buchmann et al., PQCrypto 2011]



再掲

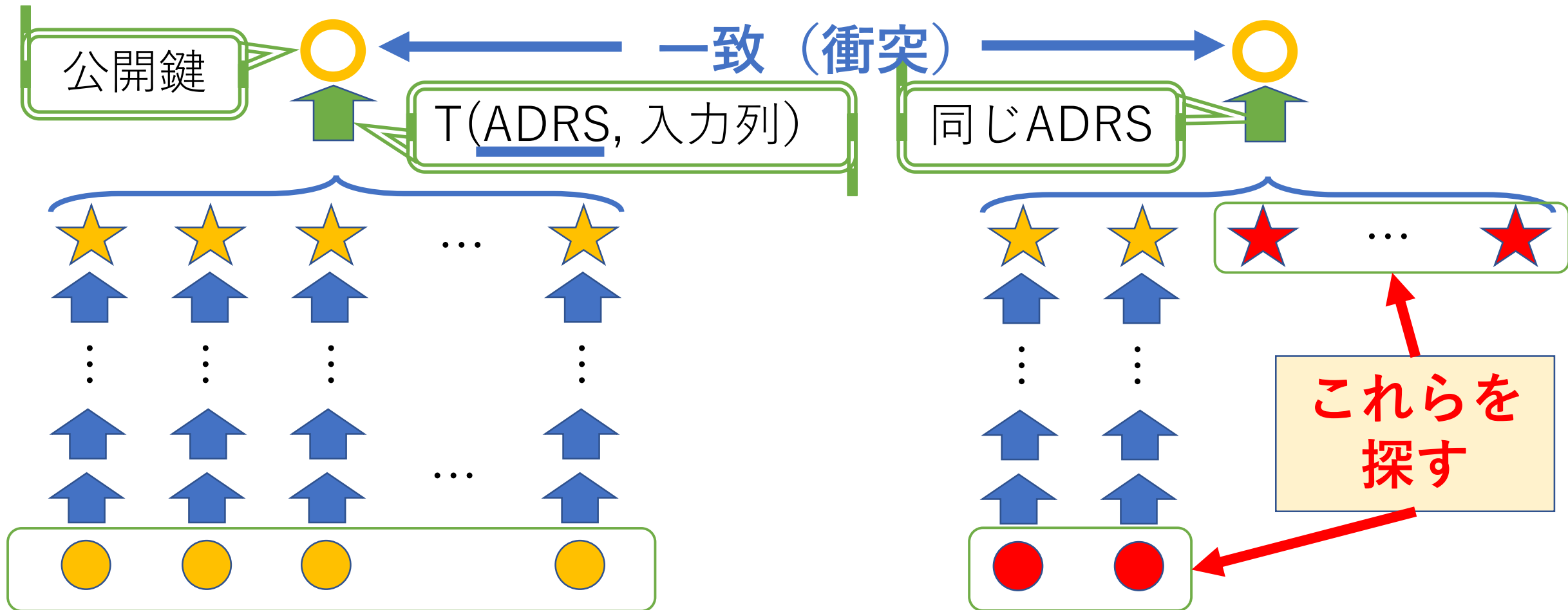
WOTS+

Winternitz one-time署名の一種 [Hulsing, AFRICACRYPT 2013]



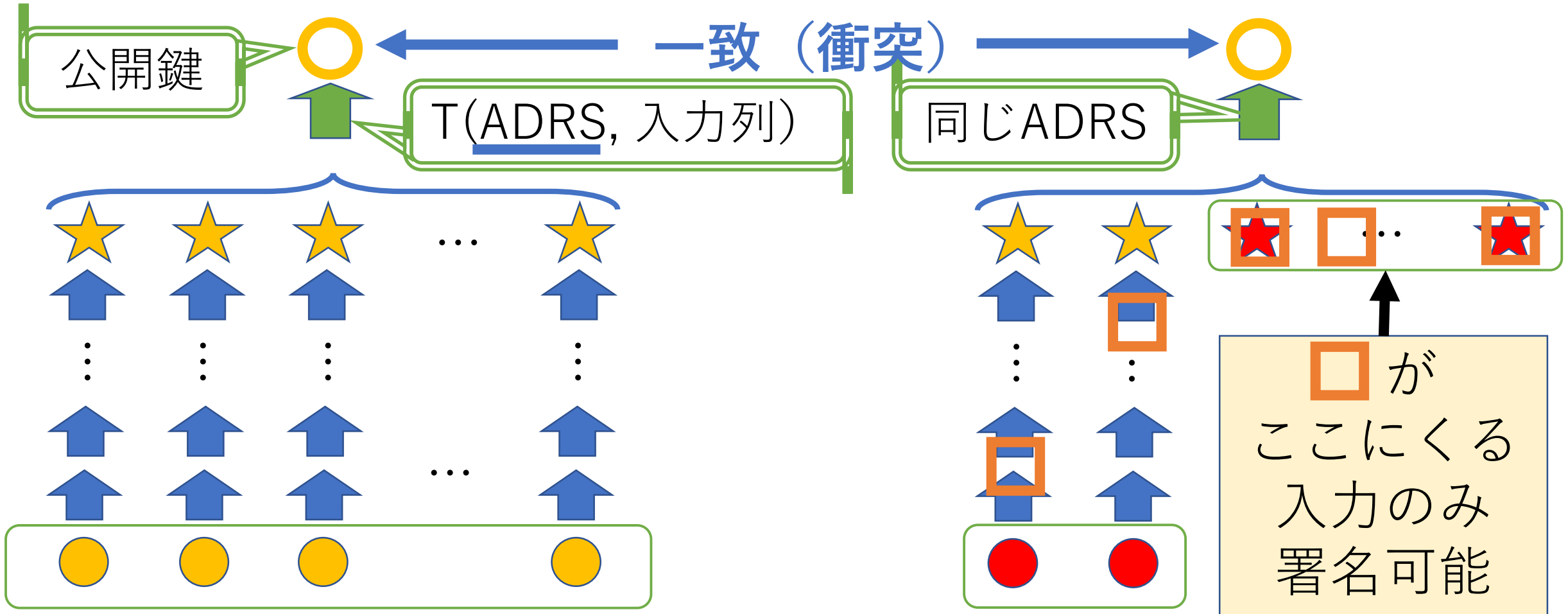
攻撃Step 1

ある場所のWOTS+について秘密鍵（もどき）を偽造

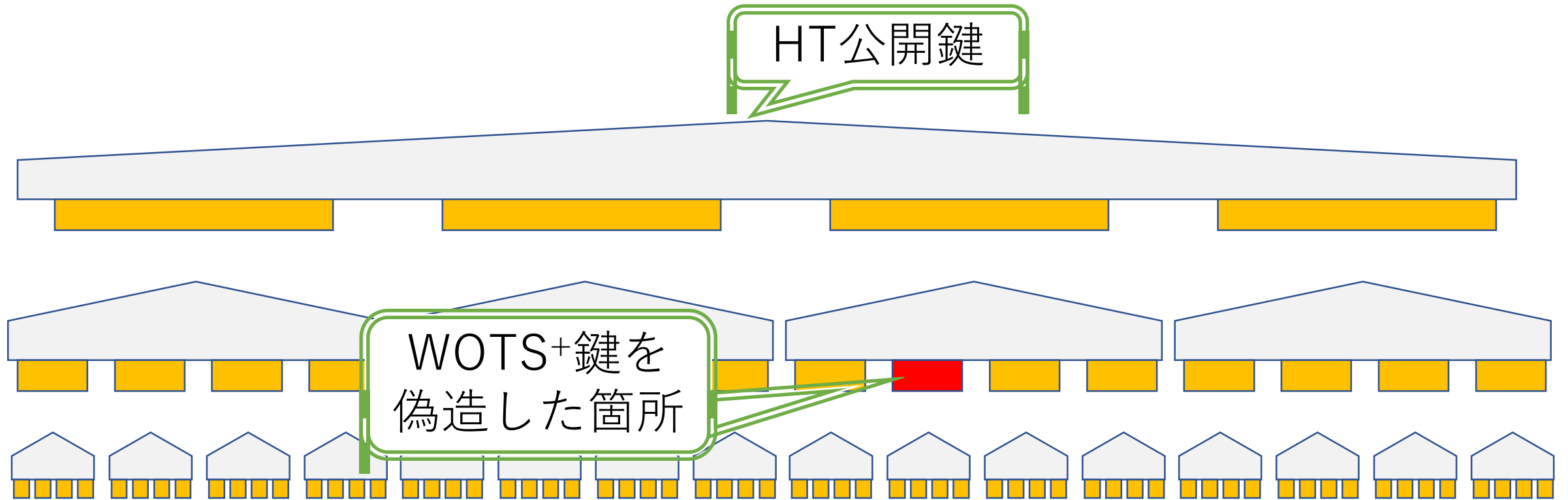


攻撃Step 1

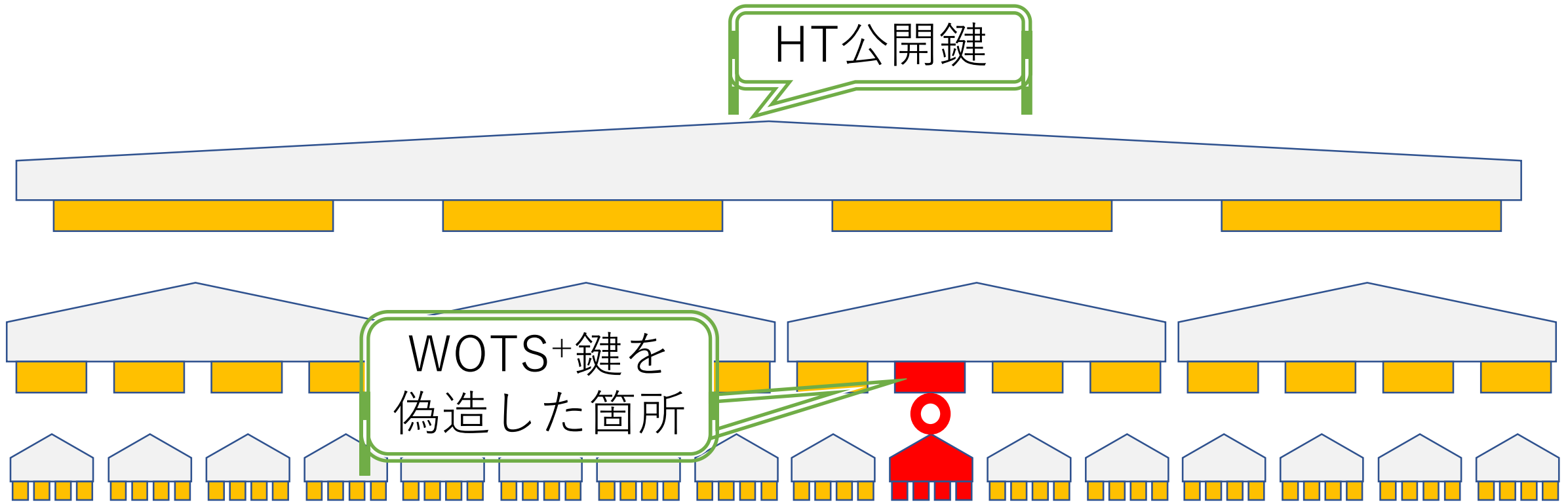
ある場所のWOTS+について秘密鍵（もどき）を偽造



攻撃Step 2



攻撃Step 2



公開鍵 **○** が偽造WOTS+鍵で署名可能となるように
直下のXMSS秘密鍵（最下位レイヤの場合はFORS秘密鍵）を偽造

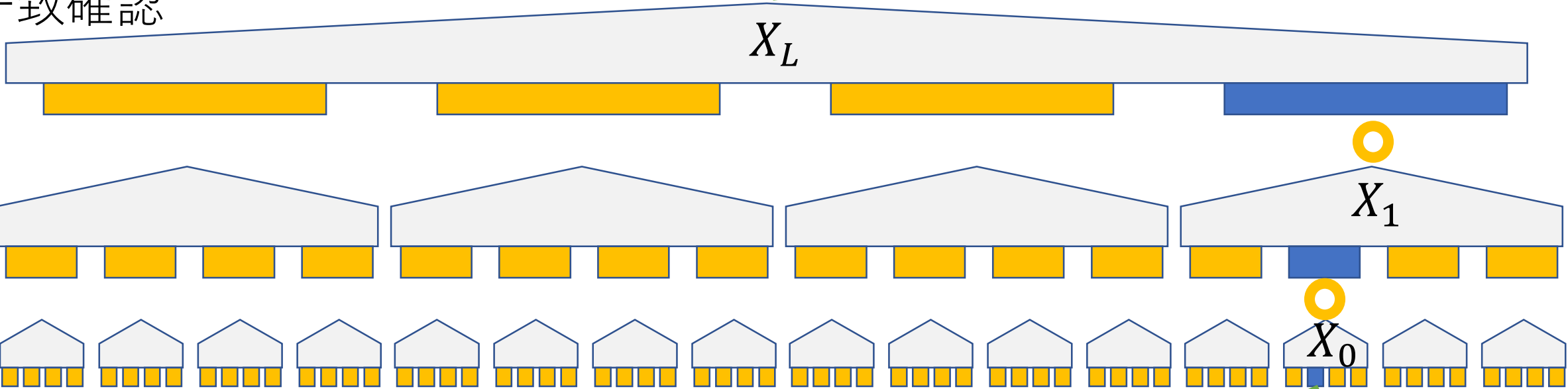
再掲

Hypertree (HT)

署名 : $(\sigma_i)_{i=0, \dots, L}$

署名検証 : 一番上まで計算して
一致確認

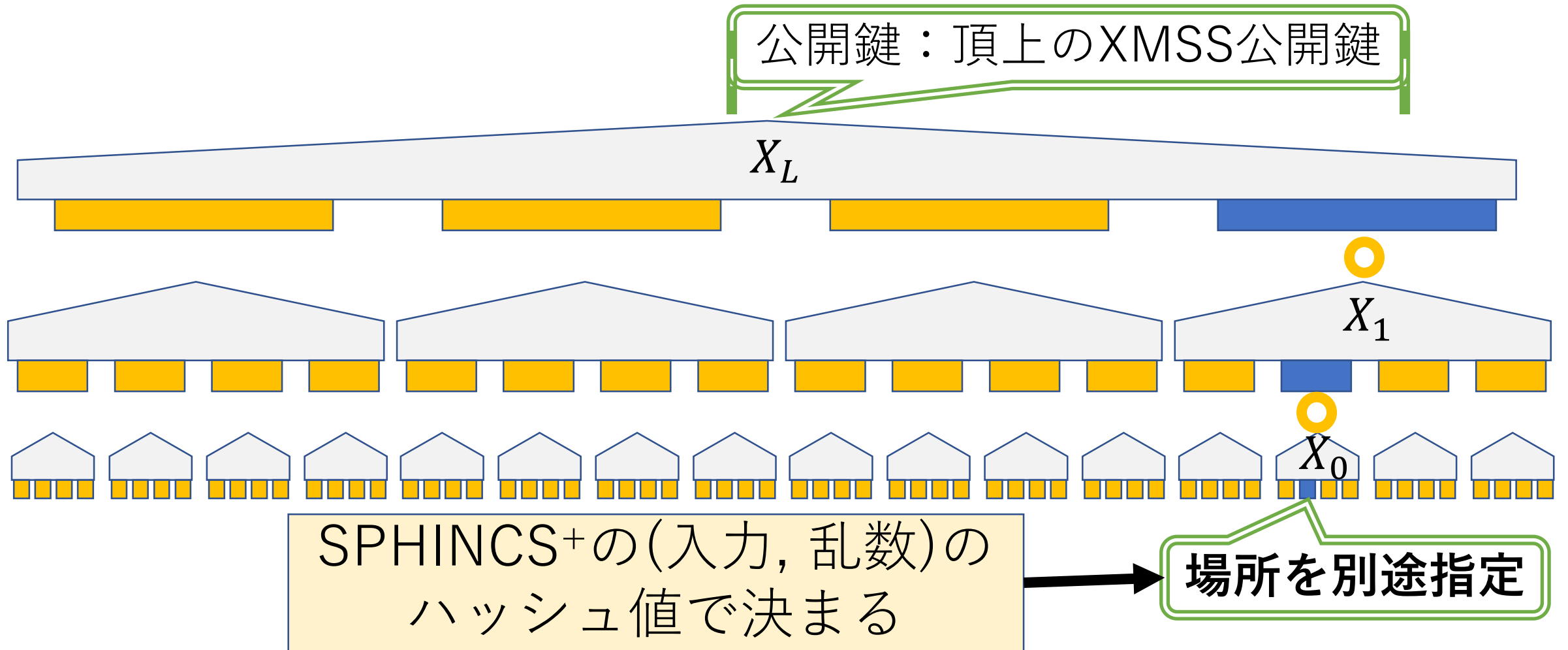
公開鍵 : 頂上のXMSS公開鍵



σ_0 : メッセージの X_0 での署名
 σ_i : X_{i-1} の公開鍵の X_i での署名 ($i > 0$)

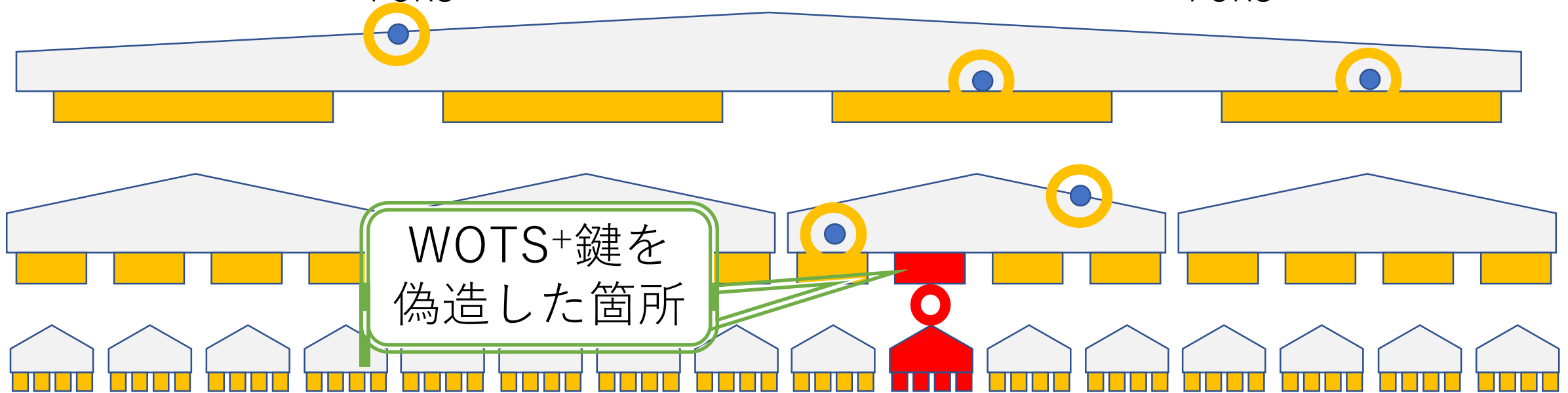
場所を別途指定

Hypertree (HT)



攻撃Step 3

偽造したWOTS+鍵が署名生成に使われるような(入力, 乱数)を探す
→ (勝手な sk_{FORS} で) 署名生成 → 偽造した鍵で pk_{FORS} を署名



* HTの署名生成に必要な残りの  は既知の署名から取ってくる

攻撃の概略と計算量

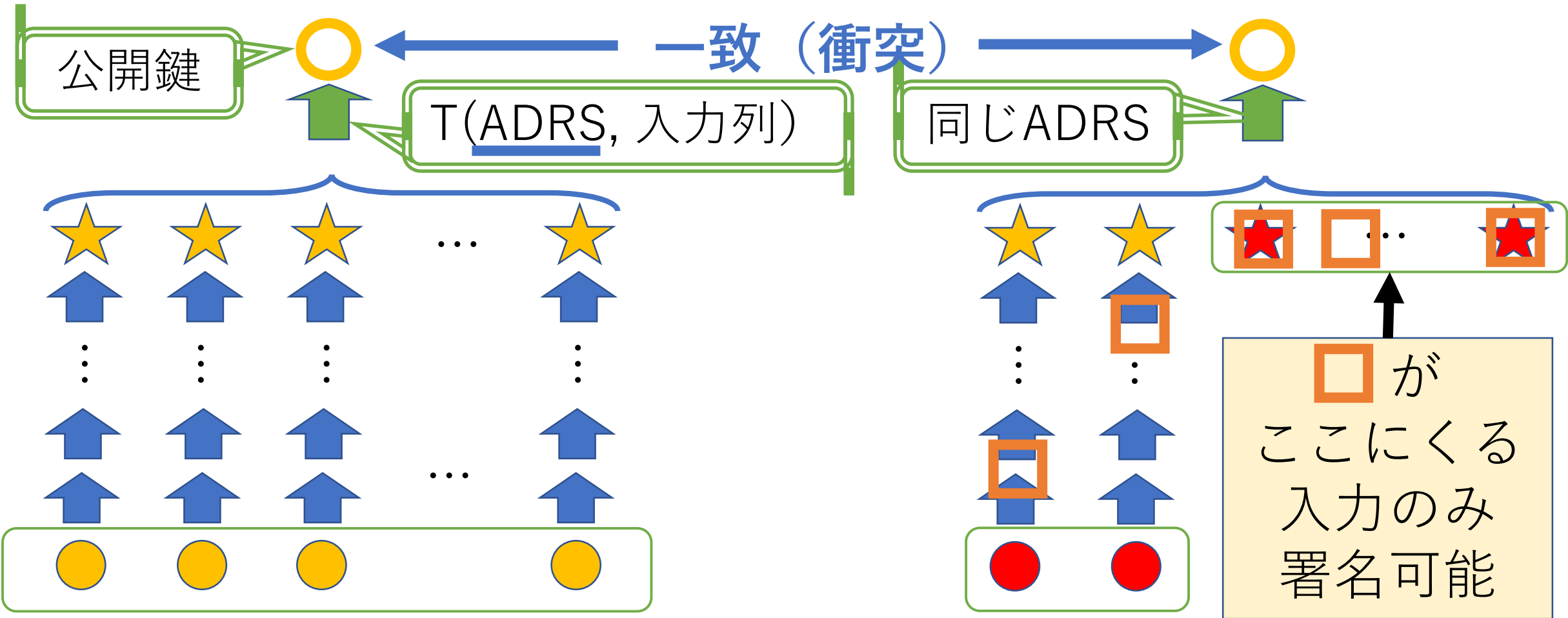
* category 5 パラメータ w/ SHA-256の場合

- Step 1 : ある場所のWOTS+鍵 (もどき) の偽造
 - tweakable hashの (多関数) multi-target第二原像探索
 - $\approx 2^{58}$ 個の既知WOTS+鍵を使用
 - 計算量 : 前半 $\approx 2^{214}$ (or $\approx 2^{196}$)、後半 $\approx 2^{216.42}$ (**後述**)
- Step 2 : 偽造箇所直下のXMSS or FORS鍵の偽造
 - 公開鍵がStep 1の鍵で署名可能になるまでランダム生成
 - 各回の成功確率 $\approx 2^{-215.68}$
- Step 3 : 偽造WOTS+鍵が用いられる (入力, 乱数) の探索
 - 乱数を変えて試す \rightarrow 各回の成功確率 (最悪時) $\approx 2^{-68}$

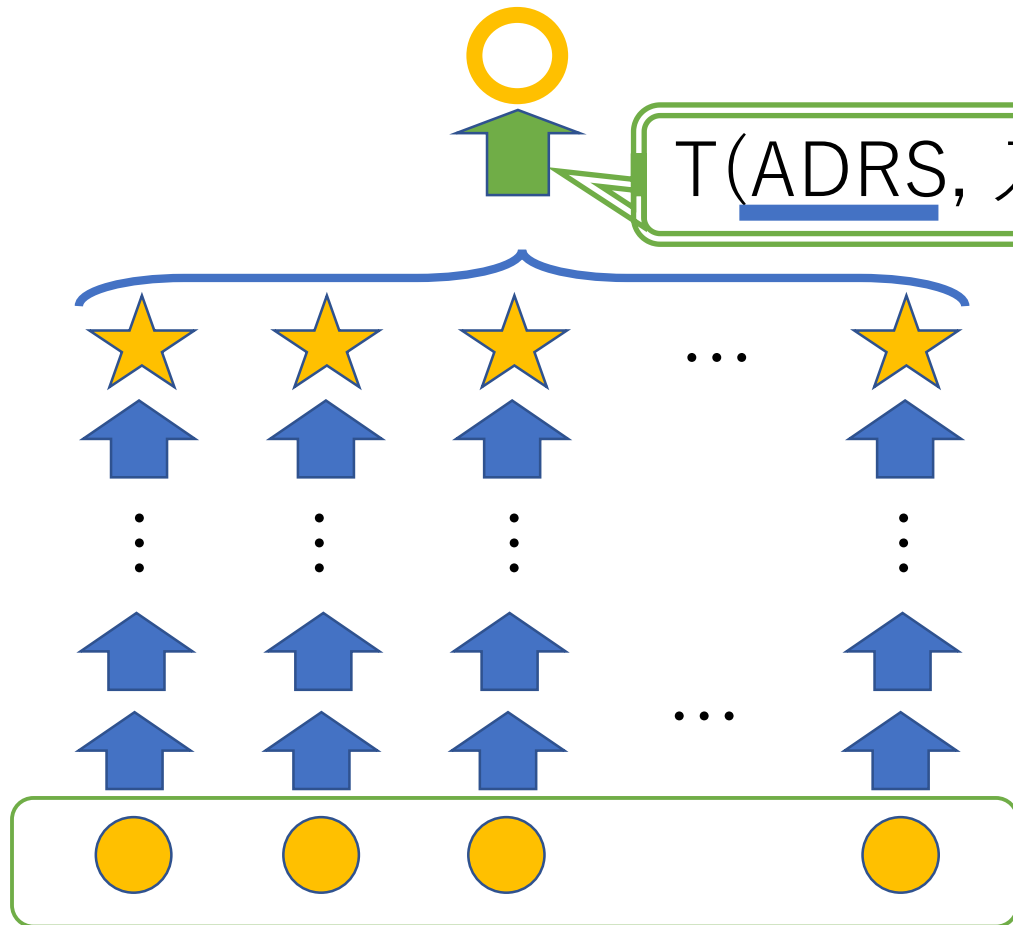
再掲

攻撃Step 1

ある場所のWOTS+について秘密鍵（もどき）を偽造



tweakable hashの構造 (概略)

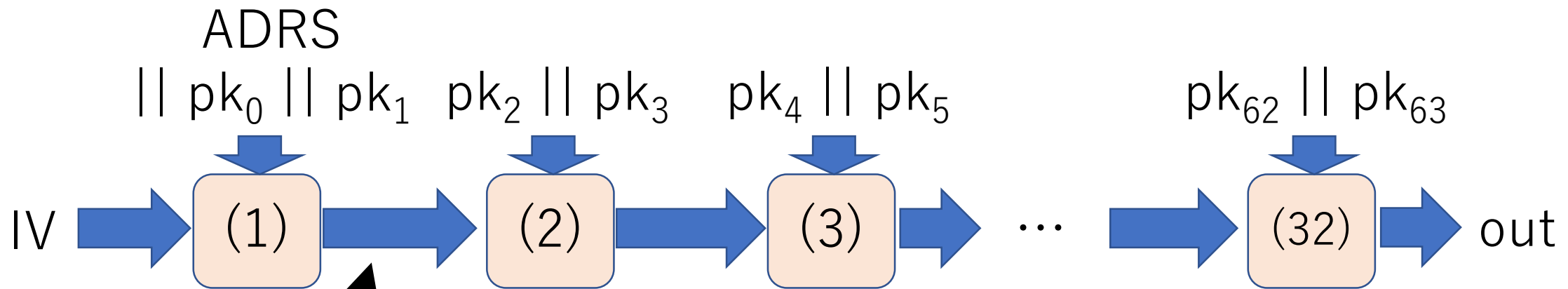


SHA-256(ADRS || pk_0 || pk_1 || ... || pk_{63})

- * 実際は「チェックサム項」を入力のも
末尾に含むが、簡略化のため割愛
- Step 1用に既知署名がより多く必要
- Step 2 (XMSS or FORS鍵偽造) の
成功確率がやや低下

tweakable hashの構造 (概略)

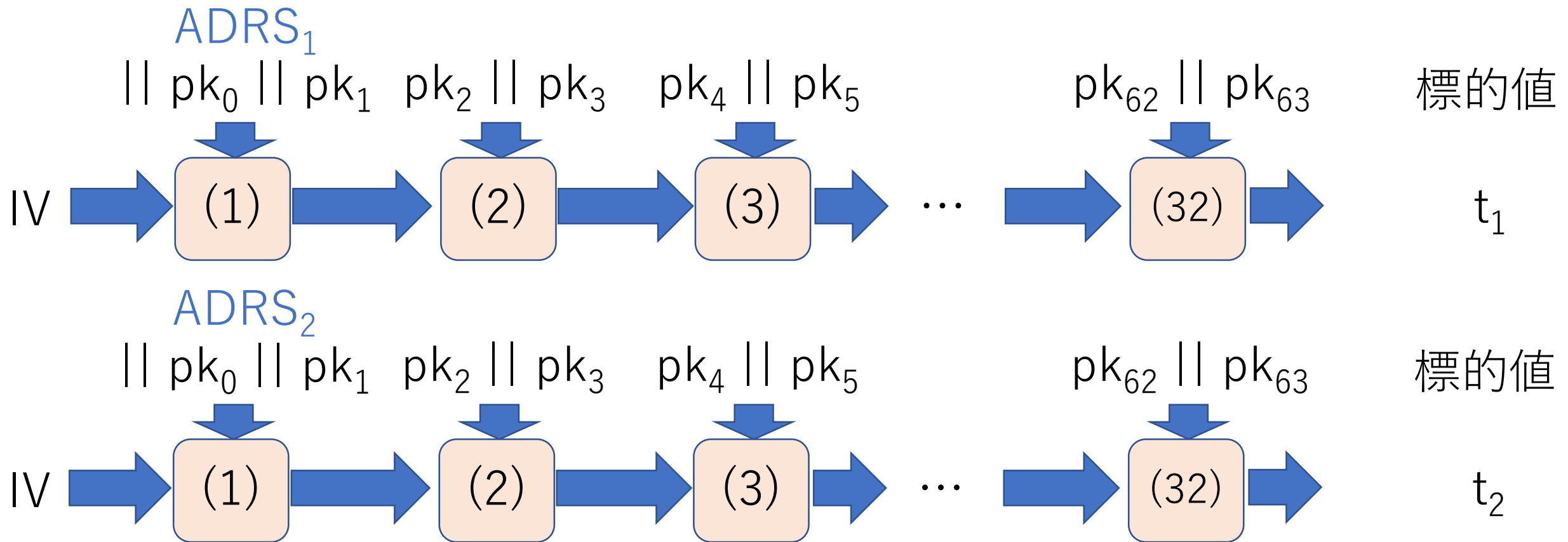
SHA-256(ADRS || pk₀ || pk₁ || ... || pk₆₃)
(Merkle-Damgard構成)



256bitの
中間出力

* 実際は入力ブロックの切れ目が異なるが、説明の簡略化を優先する

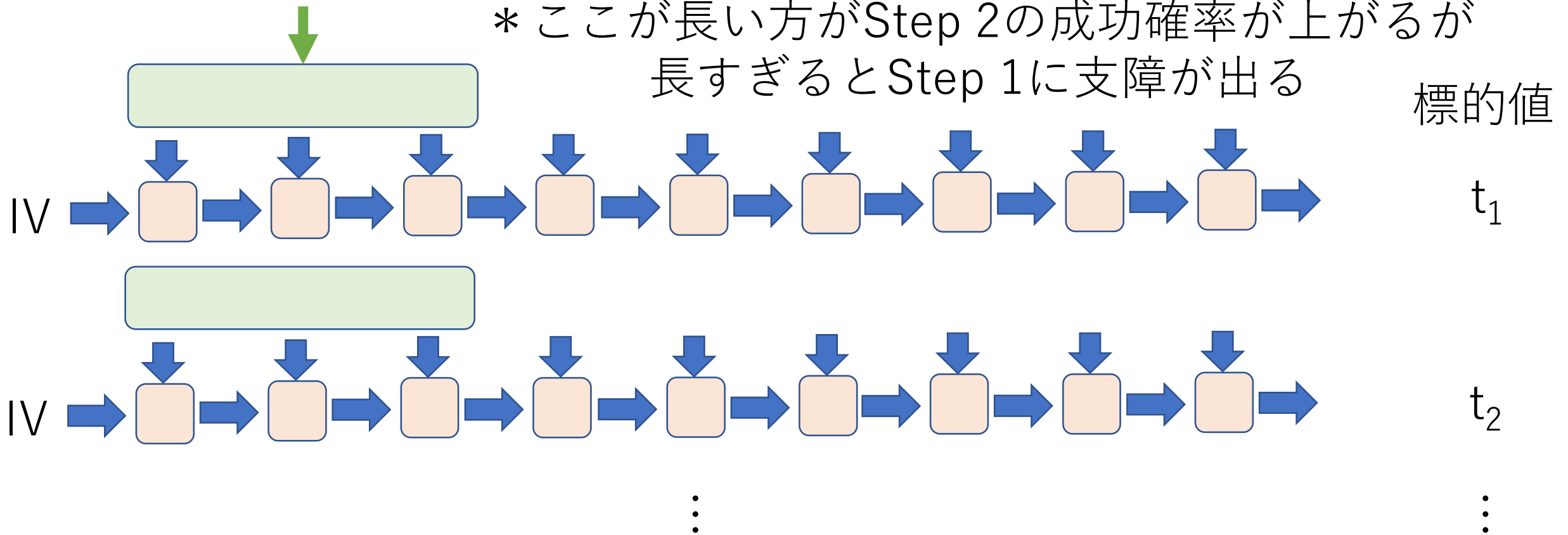
distinct-function multi-target second-preimage (DM-SP) attack



DM-SP attack

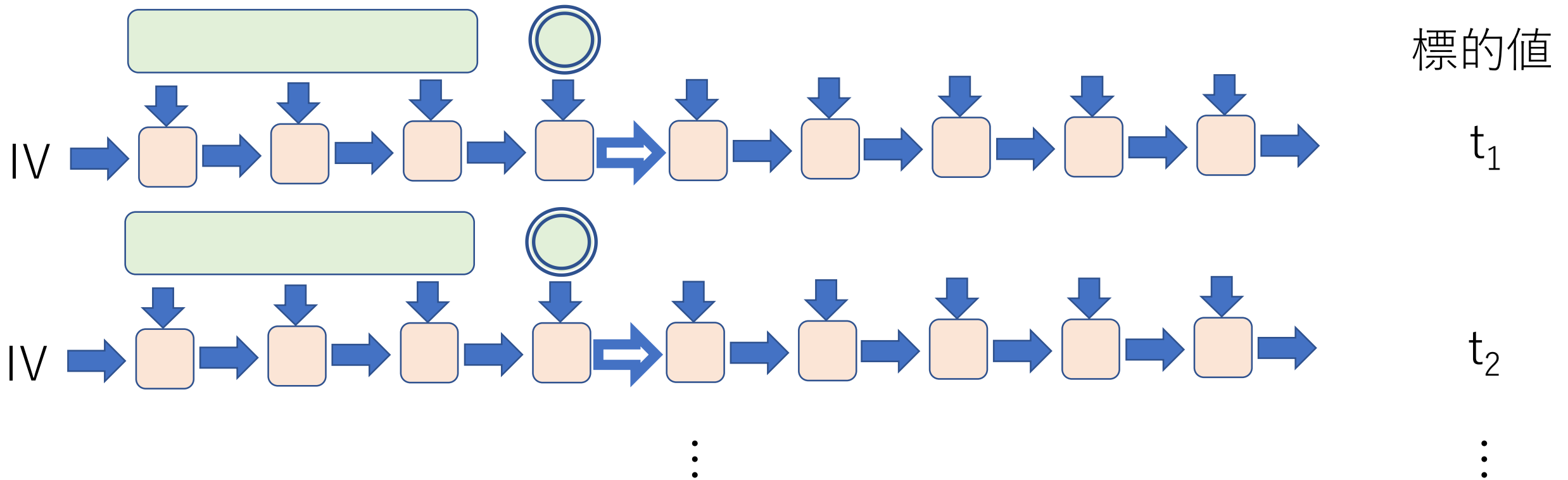
先頭の何か所かは（偽造した）WOTS+秘密鍵（の一部）から導出する

* ここが長い方がStep 2の成功確率が上がるが
長すぎるとStep 1に支障が出る



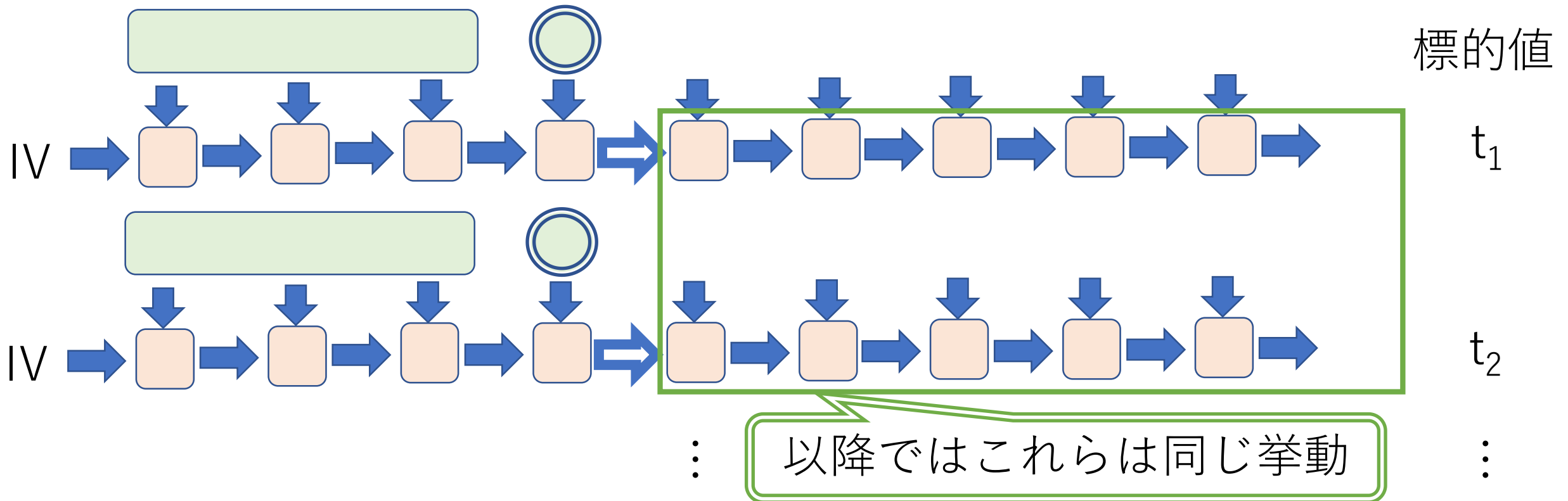
DM-SP attack

⇒ の出力が一致する ◎ たちを探す
* 実際には4本の出力を一致させる



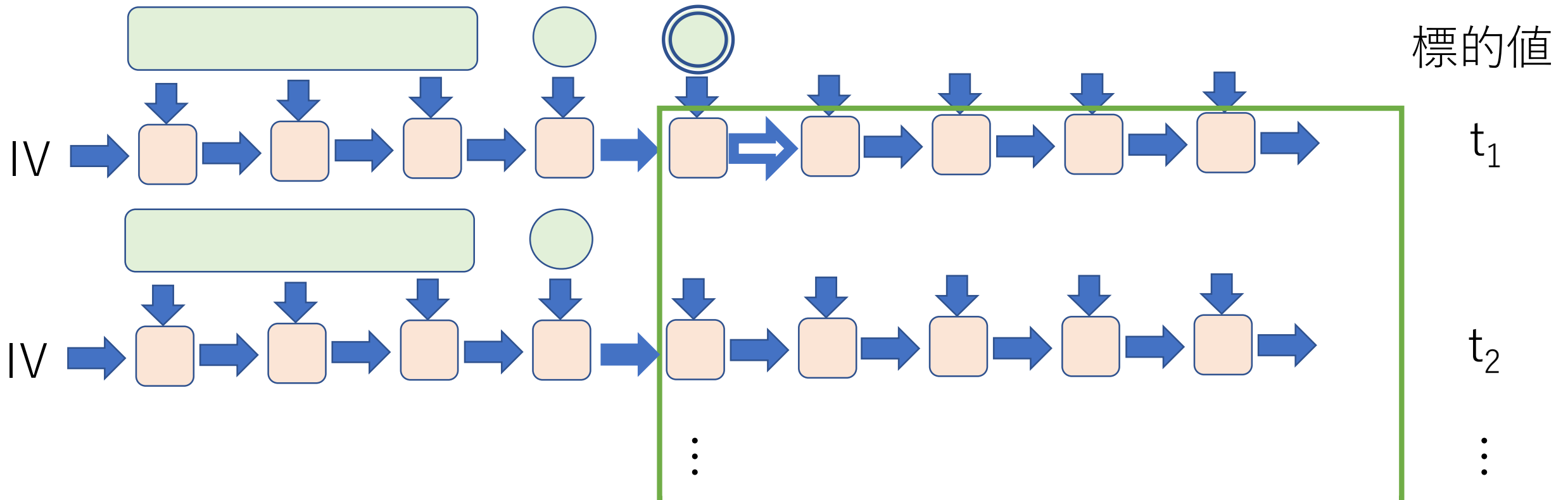
DM-SP attack

⇒ の出力が一致する ◎ たちを探す
* 実際には4本の出力を一致させる



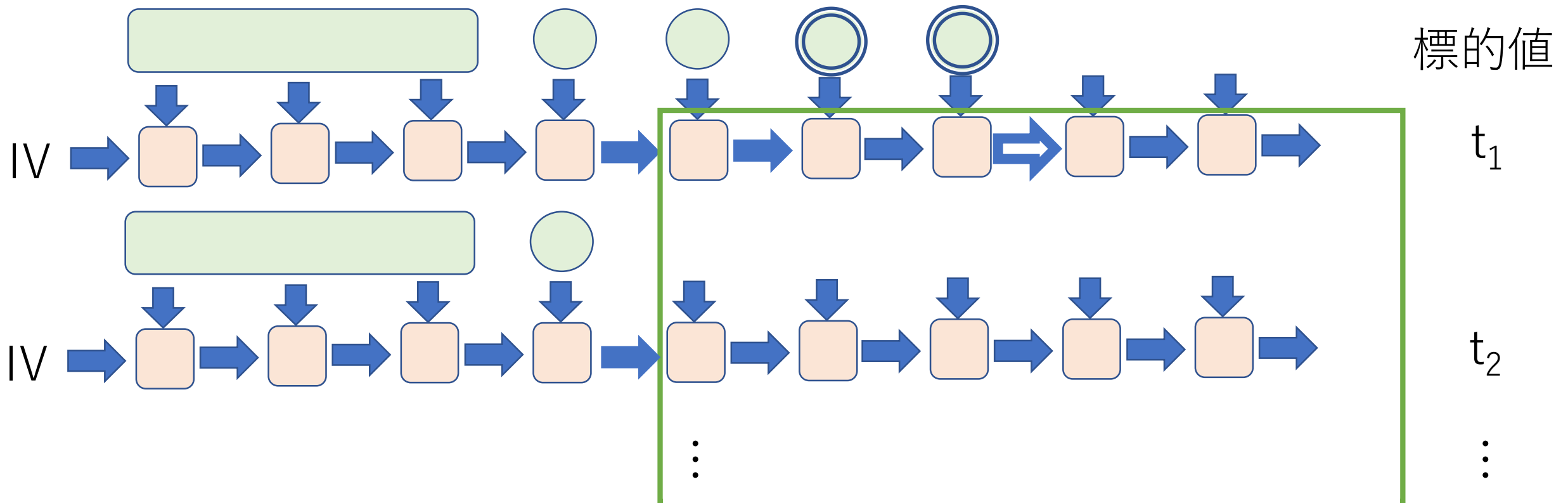
DM-SP attack

4 × 4 = 16本で \Rightarrow の出力が一致する \odot たちを探す



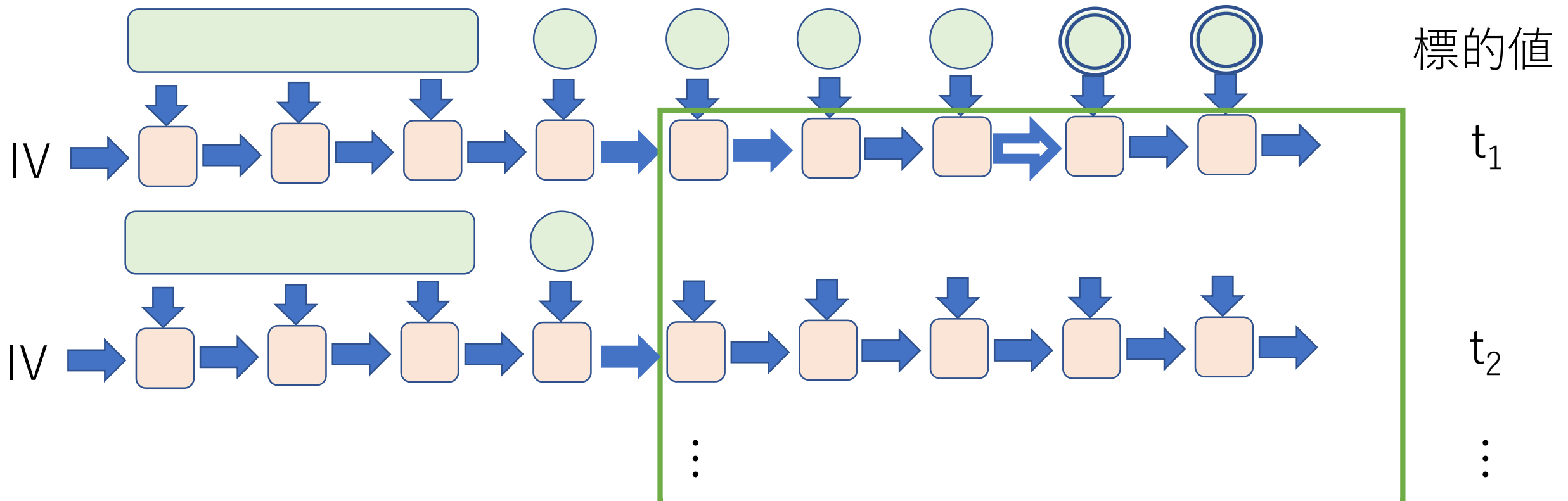
DM-SP attack

4本ずつ束ねていき、すべての \Rightarrow の出力が一致するようにする



DM-SP attack

残りの入力を調整して、標的値のどれか一つと衝突させる
↑ **distinct-functionではない** multi-target第二原像探索



DM-SP attack : 計算量と対策

- 前半 : 圧縮関数 (**256bit出力**) の4-collision
 - 計算量 $\approx 2^{256 \times 3/4} = 2^{192}$ を $2^{36.42}$ 回行う \rightarrow 計 $\approx 2^{228.42}$
 - 実際には並列化などで総計算量を削減可能
- 後半 : 同一関数のmulti-target第二原像探索
 - 標的値 $\approx 2^{39.58}$ 個 \rightarrow 計算量 $\approx 2^{216.42}$
- 対策 : **より長い出力のハッシュ関数を使用**
 - 最新版では対応済み (SHA-256 \rightarrow SHA-512)
 - open problem : より根本的な (SHA-256でもよい) 対策?
 - 逆に、ハッシュ関数の具体的構成を利用した攻撃?

まとめ

- Ray Perlner, John Kelsey, and David Cooper: “Breaking Category Five SPHINCS+ with SHA-256”, PQCrypto 2022 の紹介
 - ハッシュベース署名SPHINCS+ (w/ SHA-256) の NIST category 5 (≒256ビット古典安全性) パラメータを約217.4ビット安全性に低下させた
 - Round 3 official comment (2022.6.10)で対応済みとのこと
- SPHINCS+の構成 (の概要) の説明