

# 耐量子計算機暗号に関する いくつかの研究事例

縫田 光司 (NUIDA, Koji)

九州大学 マス・フォア・インダストリ研究所

「暗号理論の数理と社会実装」セミナー @京都大学  
2024年1月19日

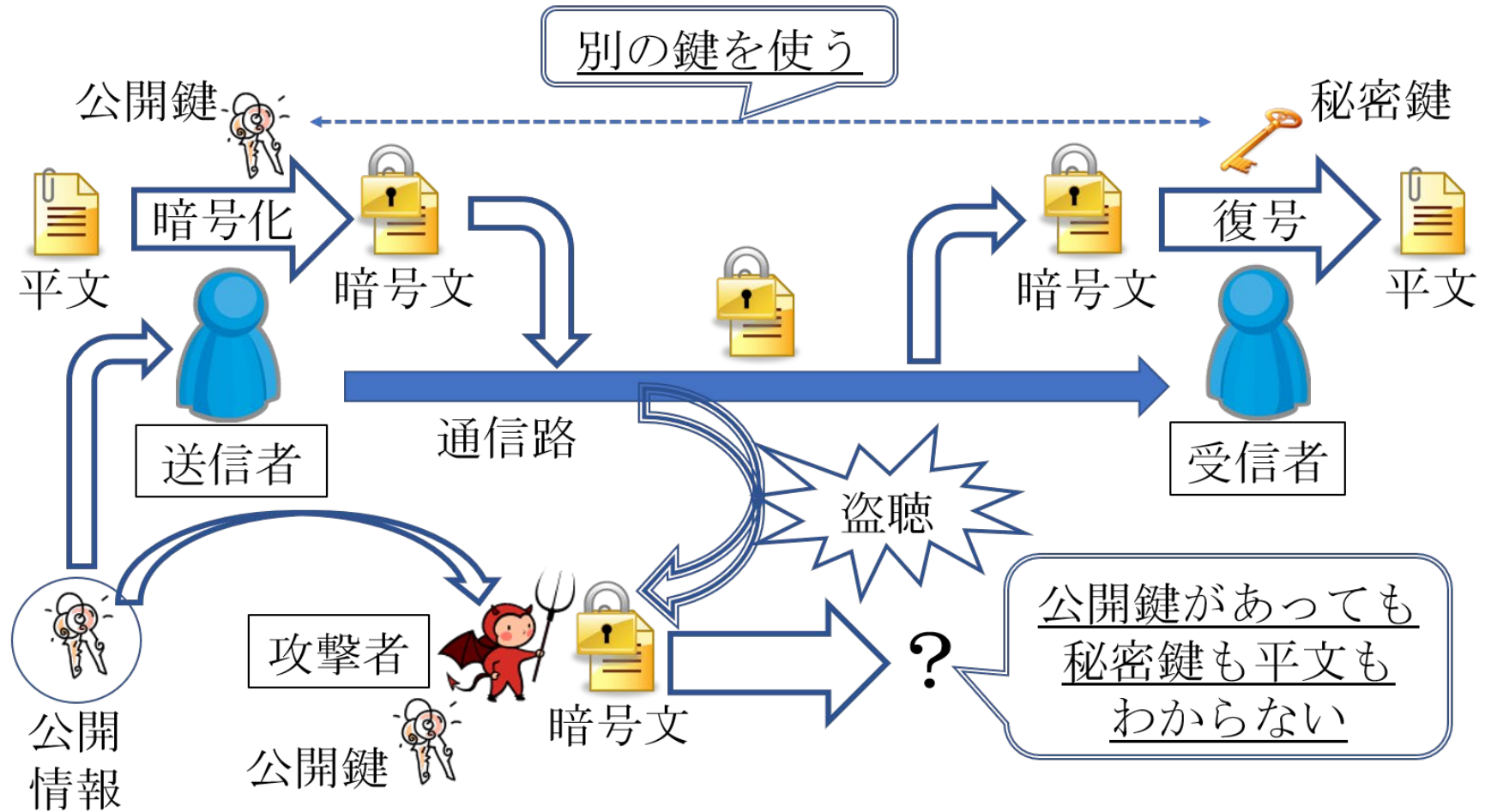
# 目次

- 概要：耐量子計算機暗号について
- 研究事例 1：署名方式UOVの改良  
[Furue et al., ASIACRYPT 2021]
- 研究事例 2：署名方式SPHINCS+の安全性解析  
[Perlner et al., PQCrypto 2022]
- 研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価  
[Odagawa-N., arXiv 2021]

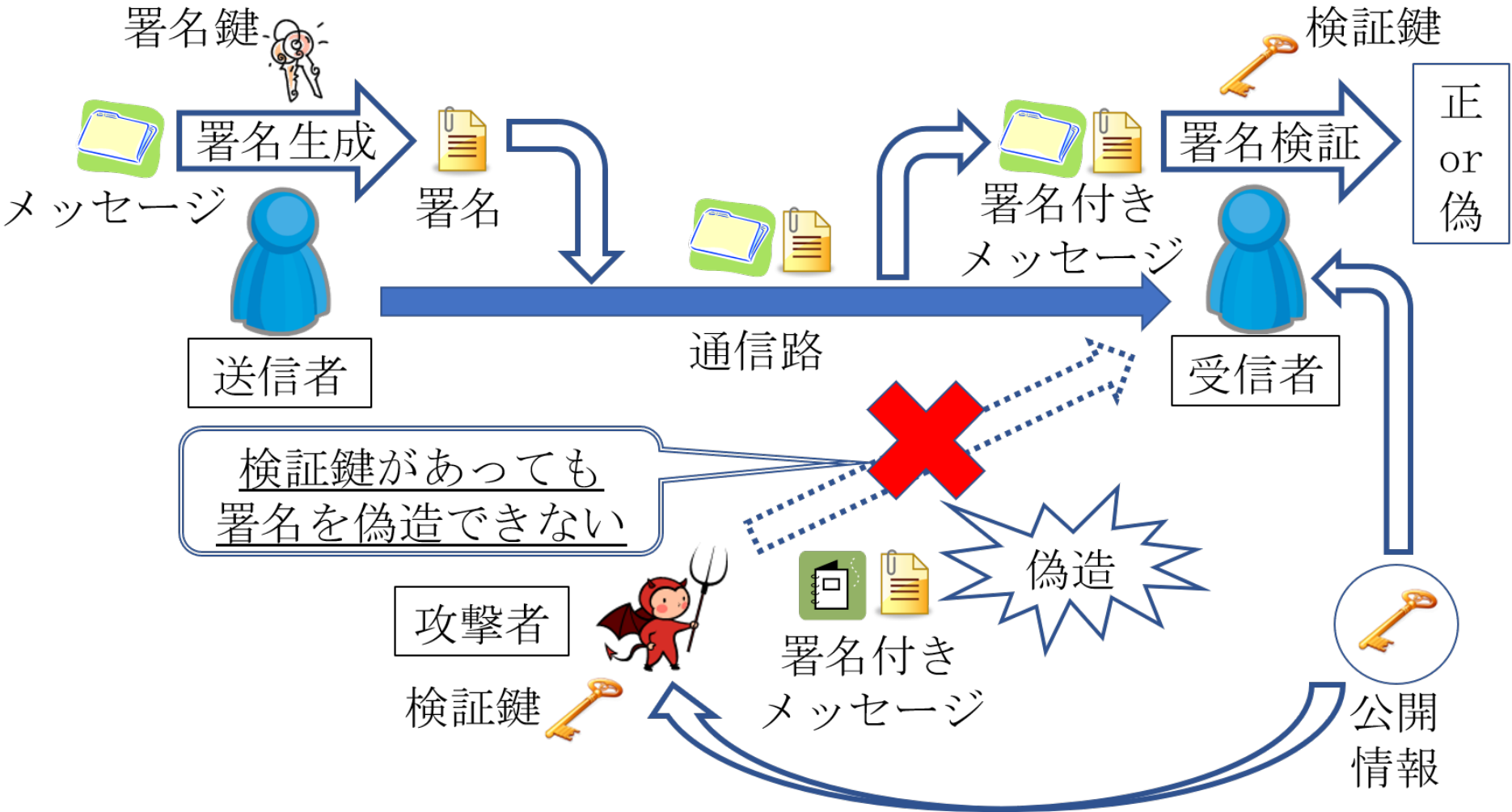
# 目次

- 概要：耐量子計算機暗号について
- 研究事例 1：署名方式UOVの改良  
[Furue et al., ASIACRYPT 2021]
- 研究事例 2：署名方式SPHINCS+の安全性解析  
[Perlner et al., PQCrypto 2022]
- 研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価  
[Odagawa-N., arXiv 2021]

# 公開鍵暗号化通信



# 電子署名



# 量子計算機（量子コンピュータ）

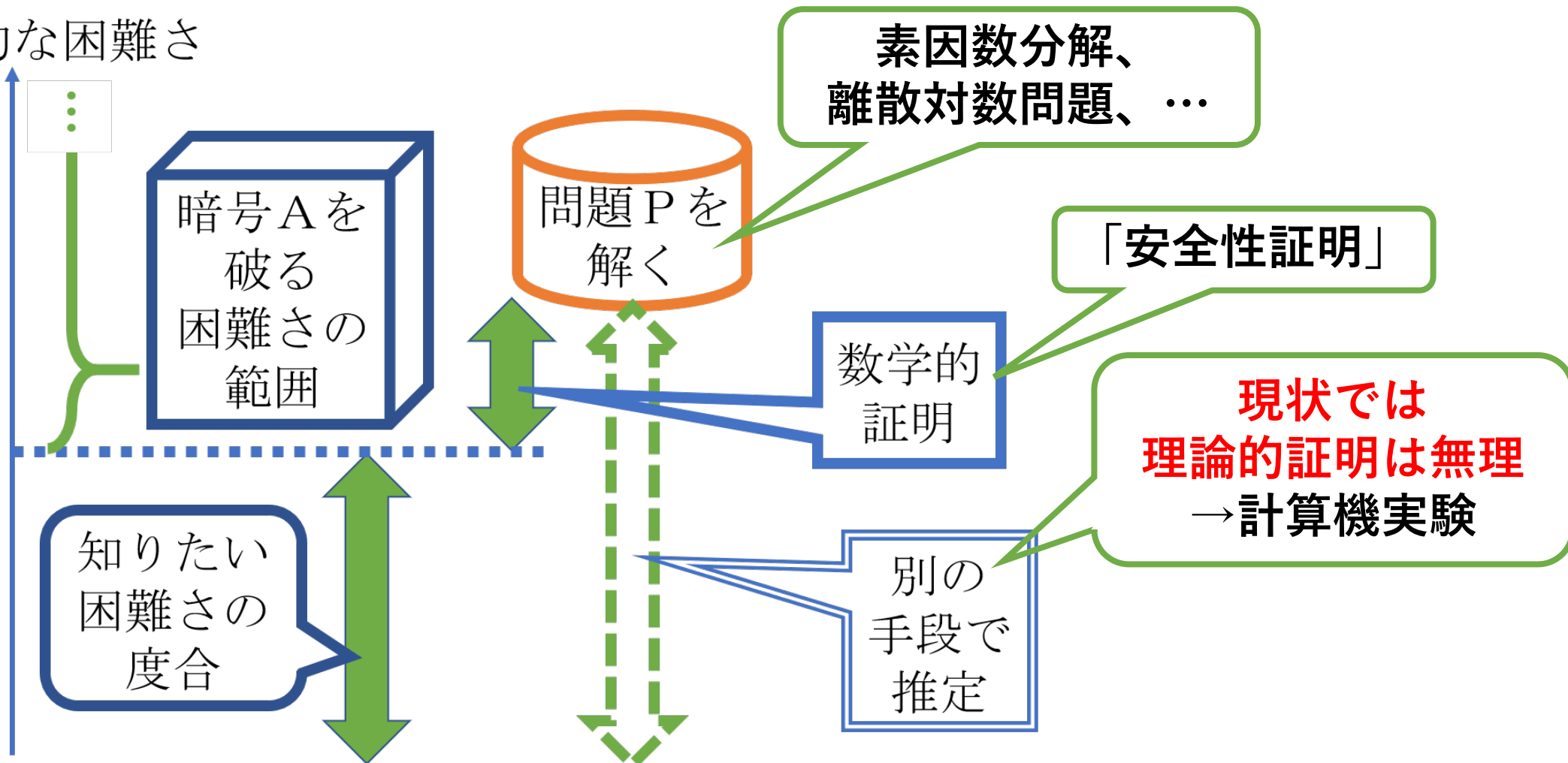
- 量子計算（量子力学的原理が組み込まれた数学モデルに基づく計算）を実行できるコンピュータ
- ある種の問題について、**古典計算機を大幅に超える性能を発揮**  
→（計算量的安全な）**暗号技術への脅威にもなり得る**
- ショアの量子アルゴリズム [Shor, 1994]を用いると、  
（よりによって）  
**素因数分解**（←RSA暗号の安全性の根拠）  
**離散対数問題**（←「楕円曲線暗号」の安全性の根拠）  
が（理論上は）効率的に計算可能となる

# 耐量子計算機暗号

- 既存の量子アルゴリズムで破れない（公開鍵）暗号技術
  - もちろん、従来型の古典計算機でも破れてはいけない
- 量子計算機の将来的な大規模化を見据えて、米国NISTにより標準化公募が行われ、応募された方式を現在厳選中
  - 2022年7月（Round 3）に最初の選定方式発表（鍵交換、署名）
  - 現在選定手順の延長戦（Round 4）中
  - さらに、署名方式を追加で公募 & 現在選定手順中

# 計算量的安全性の評価の方法論

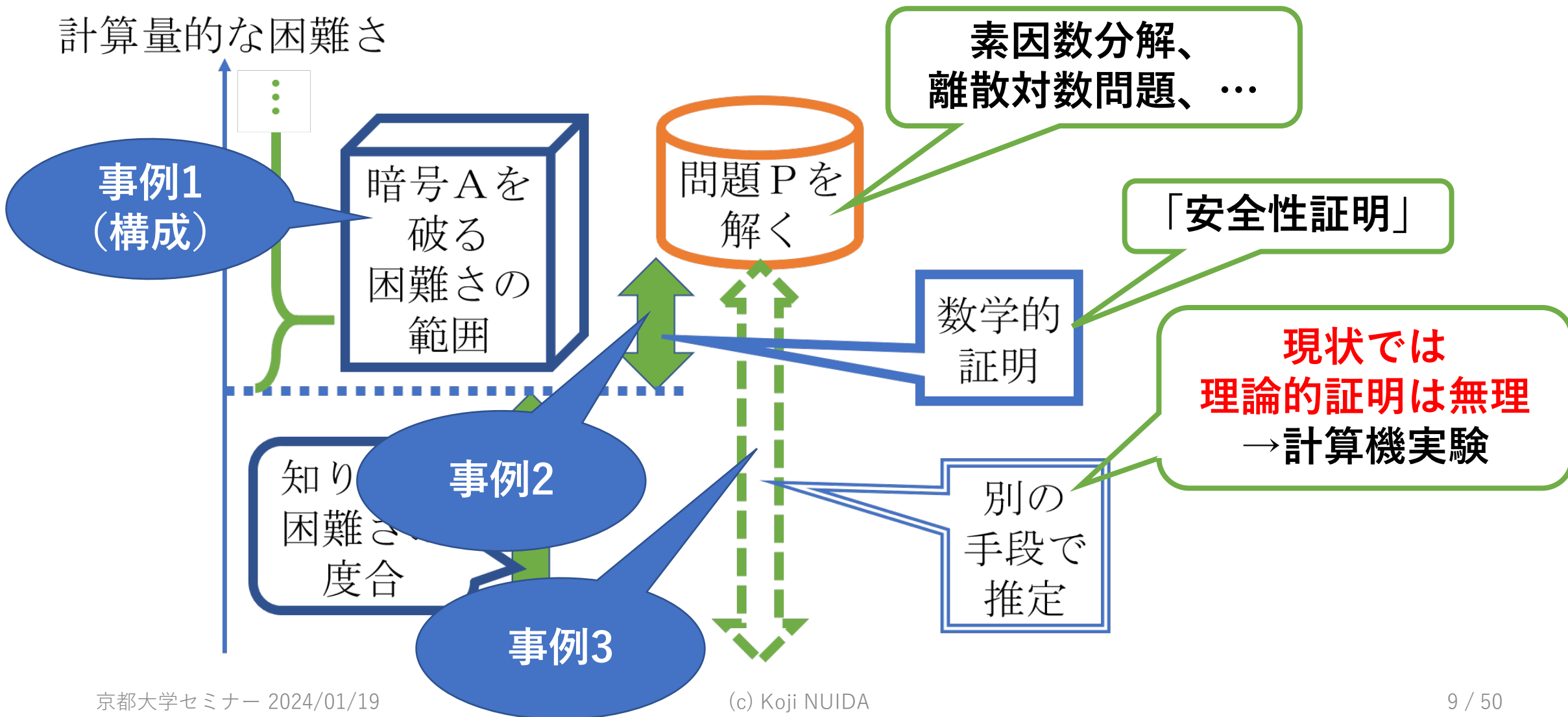
計算量的な困難さ





# 計算量的安全性の評価の方法論

計算量的な困難さ



# 目次

- 概要：耐量子計算機暗号について
- **研究事例 1：署名方式UOVの改良**  
[Furue et al., ASIACRYPT 2021]
- 研究事例 2：署名方式SPHINCS+の安全性解析  
[Perlner et al., PQCrypto 2022]
- 研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価  
[Odagawa-N., arXiv 2021]

# 署名方式UOV

- [Kipnis et al., EUROCRYPT 1999]
- 多変数多項式暗号の一種
- メッセージ $m$ の署名：連立方程式系 $\vec{P}(\sigma) = H(m)$ の解 $\sigma$ 
  - $\vec{P}$ は有限体上の2次多項式のベクトル、 $H$ は暗号的ハッシュ関数
  - 一般の状況では解 $\sigma$ を得るのは難しい (NP困難)
  - 検証鍵 $\vec{P}$ の生成時に、解を得るヒント (署名鍵) を用意しておく
- 署名の検証：与えられた $(m, \sigma)$ が $\vec{P}(\sigma) = H(m)$ を満たすか確認

# 署名方式UOV

- 検証鍵 $\vec{P}$ の生成
  - 「解きやすい」方程式系 (“central map”) に、ランダムなアフィン変数変換を施す (後者が署名鍵)
  - central map :  $v + o$ 変数の2次式で、後半 $o$ 個の変数に関する2次項なし
- 署名生成 ( $\vec{P}(\sigma) = H(m)$ の解 $\sigma$ の計算)
  - 変数変換でcentral mapの場合に帰着
  - 前半 $v$ 個の変数の値をランダムに選択
  - 残りは $o$ 変数連立1次方程式なので解が (存在すれば) 求まる (解が存在しなければ前ステップからやり直し)

# Block-Anti-Circulant (BAC-)UOV

- UOVの難点の一つ：公開鍵サイズが大きい
  - (2次形式で表した際の) 行列が2次元的なデータであるため
- BAC-UOV [Szepieniec-Preneel, SAC 2019]
  - 行列をブロック化して、各ブロックを(反)巡回行列から選ぶ
  - (反)巡回行列は1次元構造で記録可能 → 公開鍵サイズの削減
- しかし、BAC-UOVには攻撃が存在 [Furue et al., PQCrypto 2020]
  - 行列の構造の特殊性を利用
  - 完全に破られてはいないが、パラメータ選択に大きな悪影響

# 提案方式QR-UOV [Furue et al., 2021]

- QRは“quotient ring”の略
- BAC-UOVの（反）巡回行列を剰余環 $F_q[x]/(f)$ の元へ一般化
  - 各行は $g \bmod f, xg \bmod f, x^2g \bmod f, \dots$ の係数ベクトル
  - 巡回行列は $f = x^k - 1$ の形の特殊例
  - $f$ は既約多項式にする（BAC-UOVへの攻撃法の対策）
- 要となる数学的事実：ある多項式から得られる行列全体の集合を $A_f$ とするとき、下記を満たす可逆行列 $W$ が存在する：  
どの行列 $X \in A_f$ についても、 $WX$ は対称行列である
  - 2次形式として扱う際の転置の操作と相性がよい

# 提案方式QR-UOV [Furue et al., 2021]

- 多変数多項式暗号の署名方式Rainbowとの性能比較
  - Rainbow : NIST標準化Round 3候補の一つ [Ding et al.]

安全性強度	公開鍵サイズ (KB)		署名サイズ (B)	
	Rainbow	QR-UOV	Rainbow	QR-UOV
I	57.4	23.8	66.0	113.9
III	252.3	85.8	164.0	166.8
V	511.2	264.3	212.0	230.9

- 公開鍵サイズが30%~50%程度に削減（署名サイズは増大）
- NIST標準化追加公募に応募中
  - 東京大-九州大（IMI、池松）-長崎県立大-NTTの合同チーム

# 目次

- 概要：耐量子計算機暗号について
- 研究事例 1：署名方式UOVの改良  
[Furue et al., ASIACRYPT 2021]
- **研究事例 2：署名方式SPHINCS+の安全性解析**  
[Perlner et al., PQCrypto 2022]
- 研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価  
[Odagawa-N., arXiv 2021]



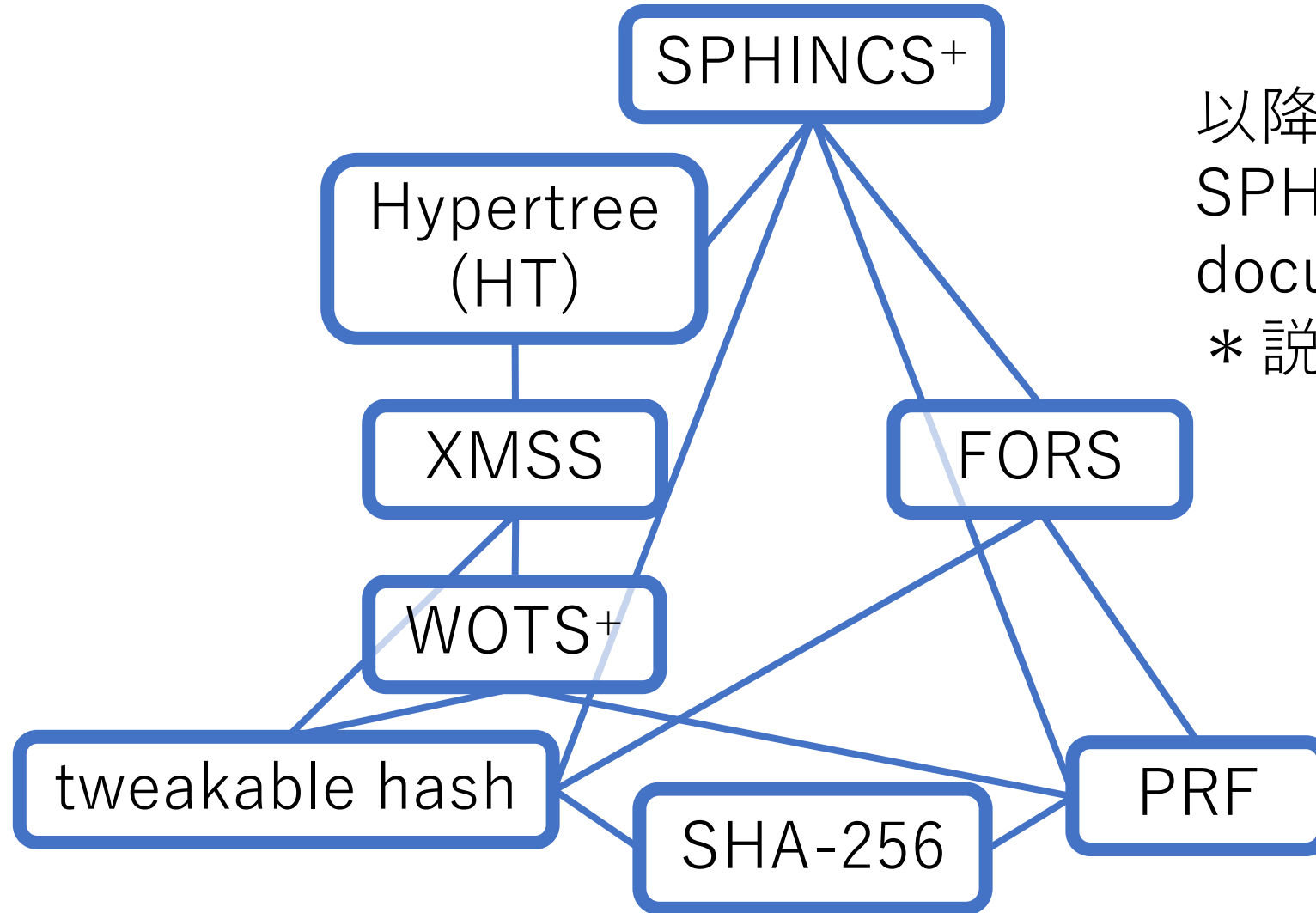
# SPHINCS<sup>+</sup>

- SPHINCS : [Bernstein et al., 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<sup>+</sup> : NIST標準に選定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<sup>+</sup>への攻撃

- [Perlner et al., PQCrypto 2022]
- SPHINCS<sup>+</sup> (SHA-256を使用) 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
  - つまり、安全性が完全に破れてはいないものの、当初の想定通りの安全性強度ではなかった
  - Round 3 official comment (2022.6.10)で対応済みとのこと

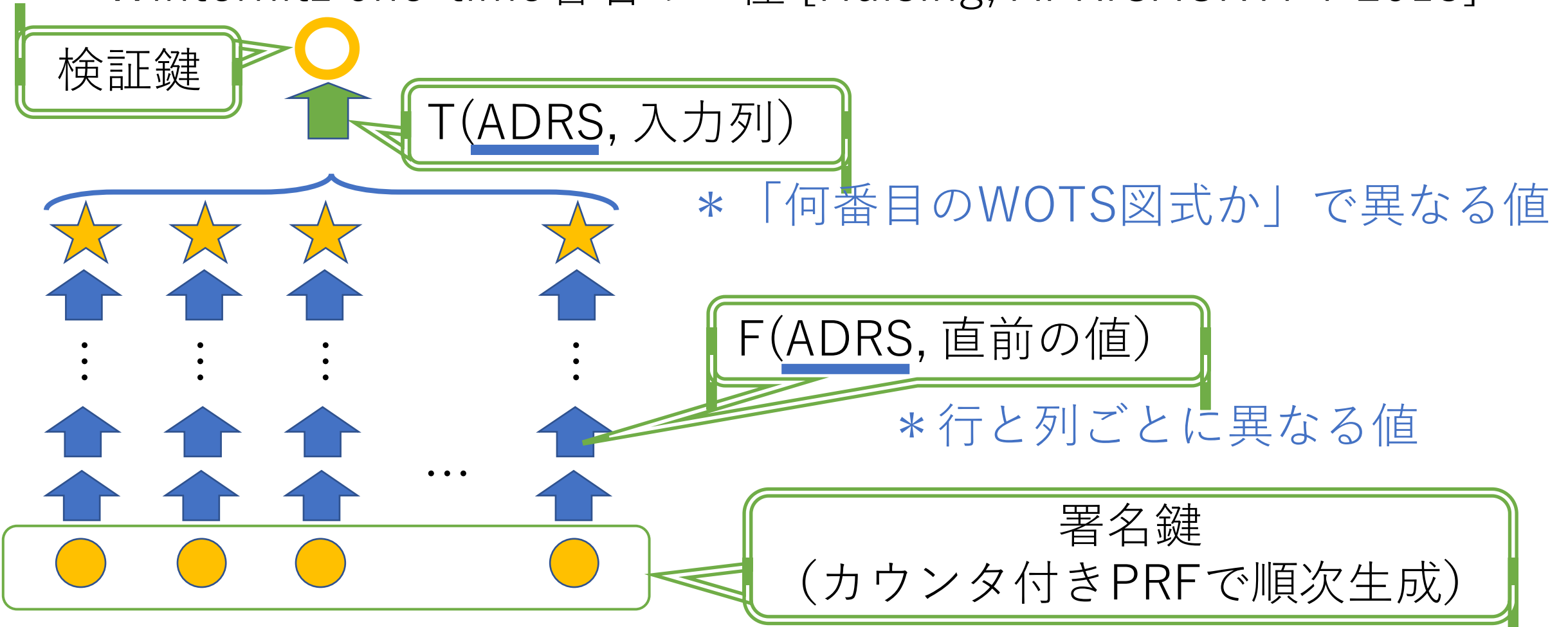
# SPHINCS+の構成要素と依存関係



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

# WOTS+

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



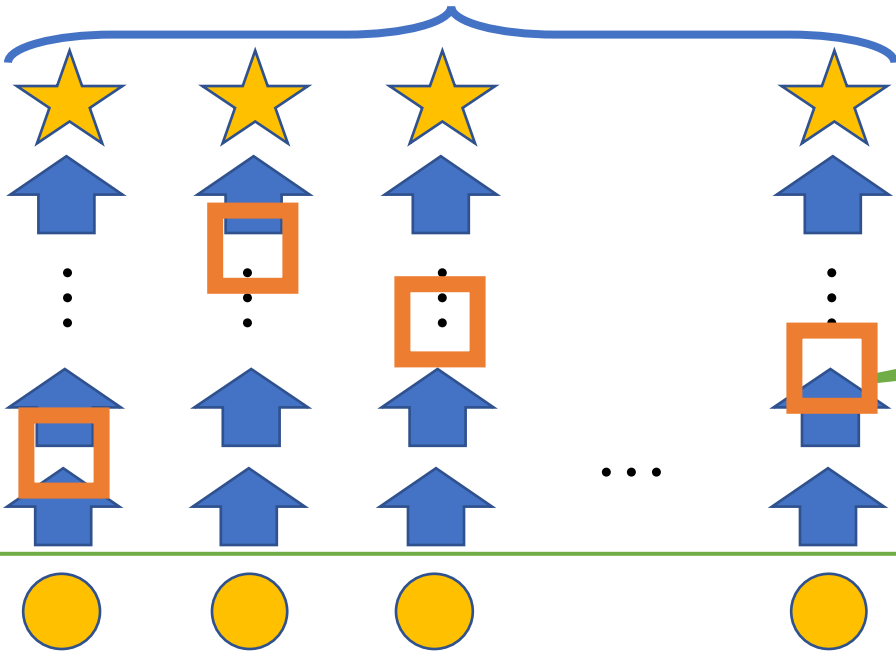
# WOTS+

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

検証鍵



\* 署名検証：一番上まで計算して、 =  か確認



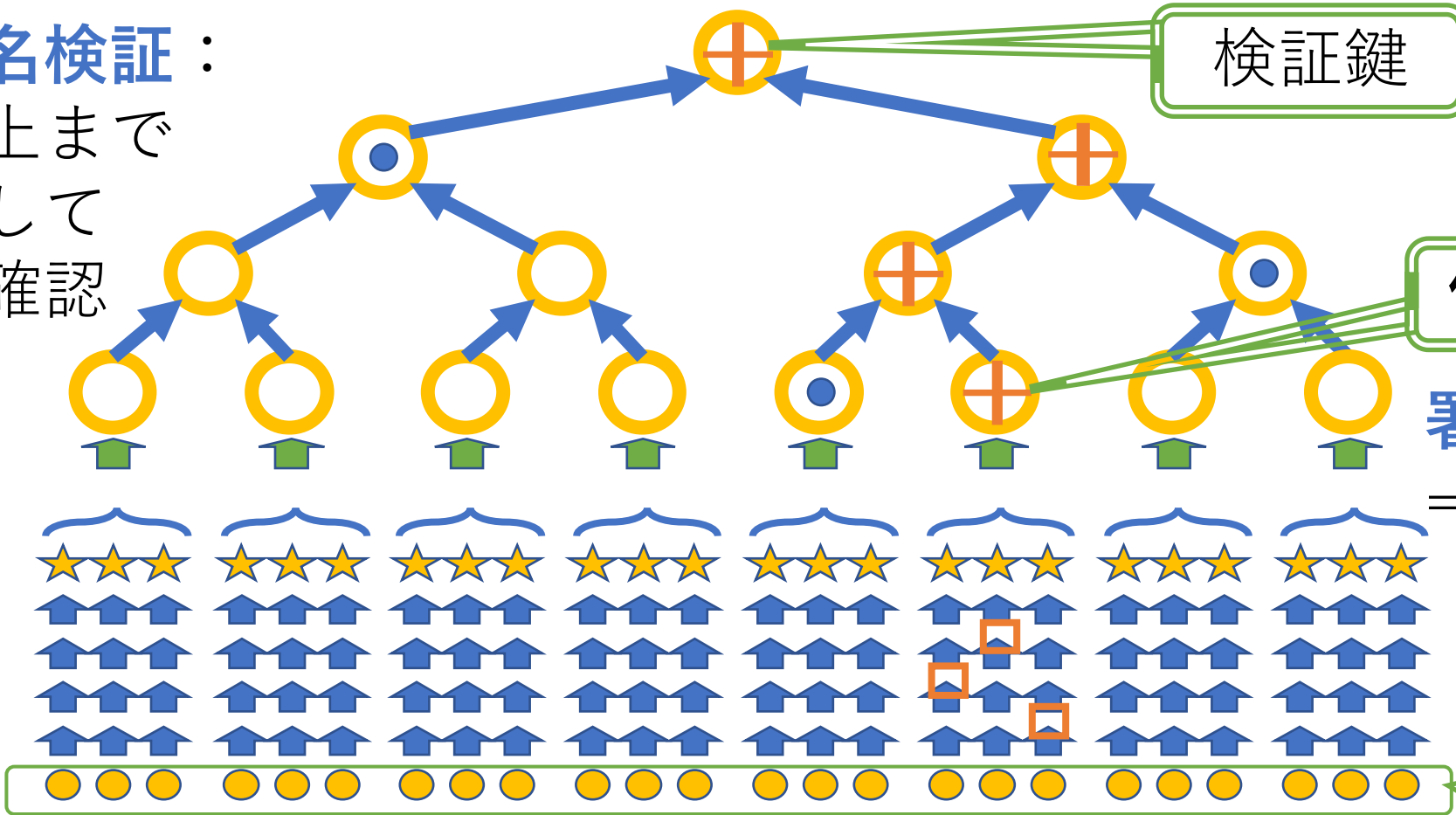
位置（高さ）を入力から決める  
（PRF等はいない）  
→ これらの位置の値の組が署名

署名鍵  
（カウンタ付きPRFで順次生成）

# (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

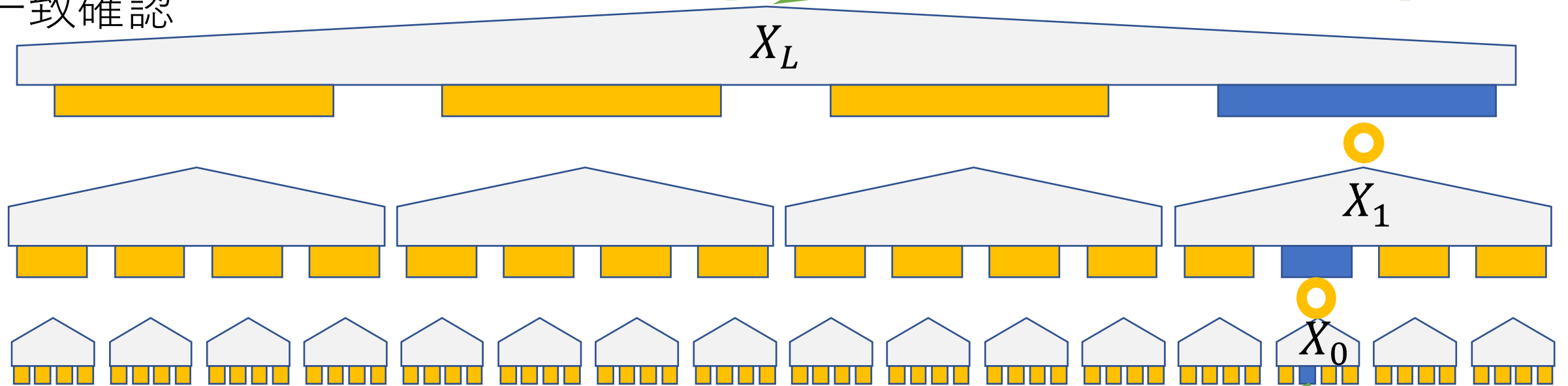
署名鍵

# 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

検証鍵 : 頂上のXMSS公開鍵





$\sigma_0$ : メッセージの  $X_0$  での署名

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

場所を別途指定

# FORS [fɔ:rs]

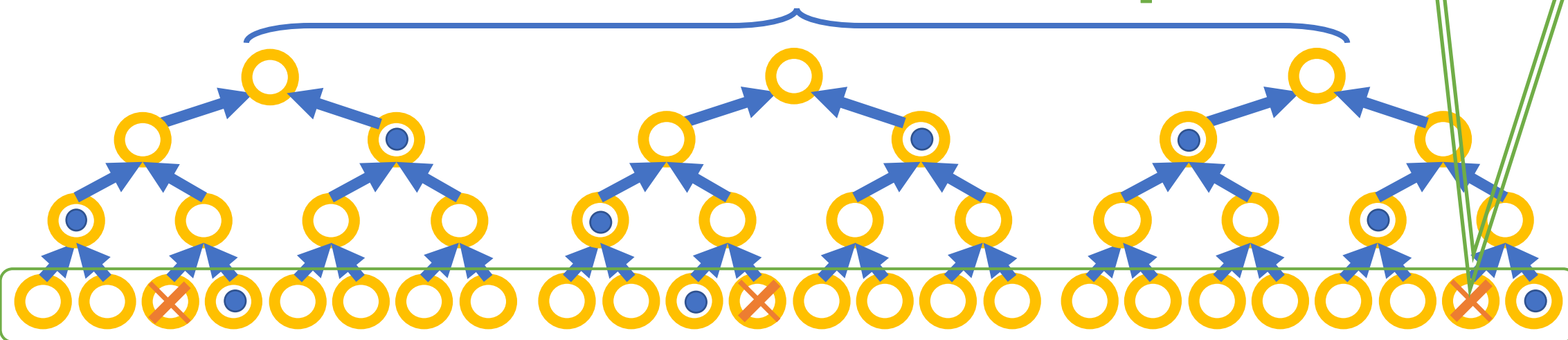
“forest of random subsets”の略

署名：{たち, たち}

署名検証：一番上まで計算して  
一致確認

検証鍵

各木の  の位置を  
入力から決める

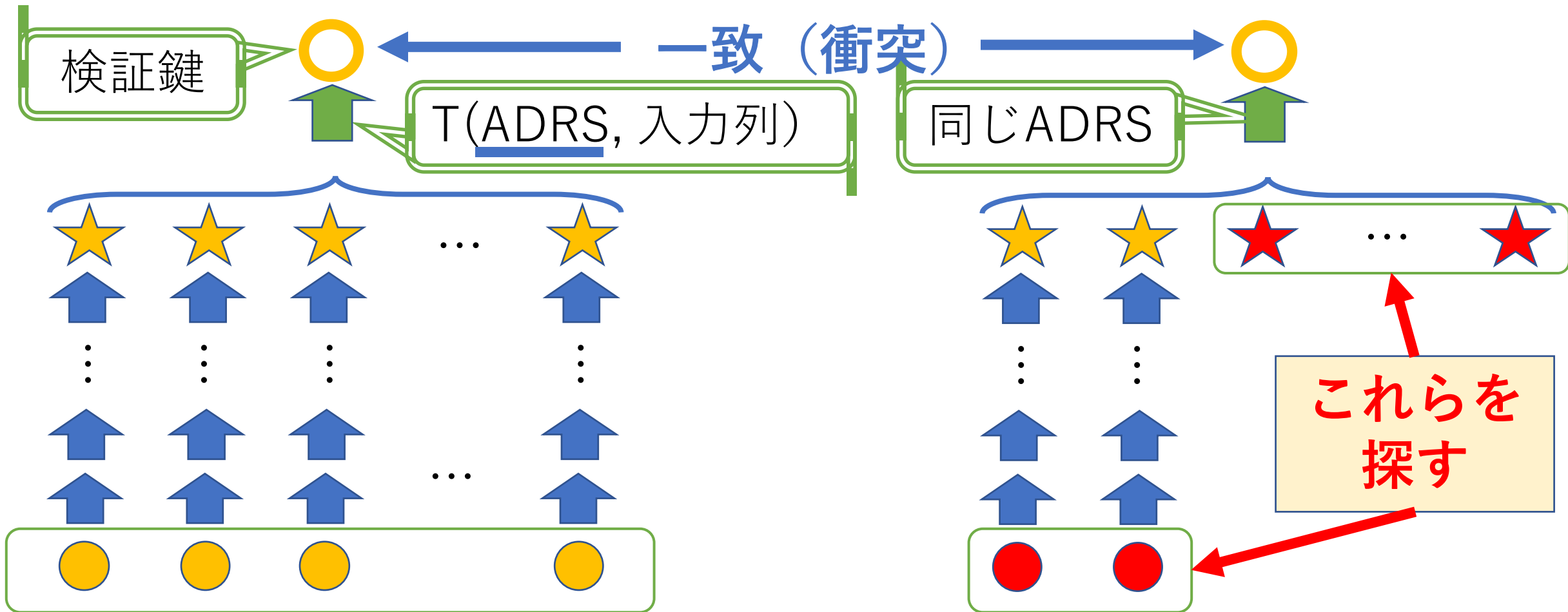


署名鍵 (PRF値) \* インデックスを別途指定



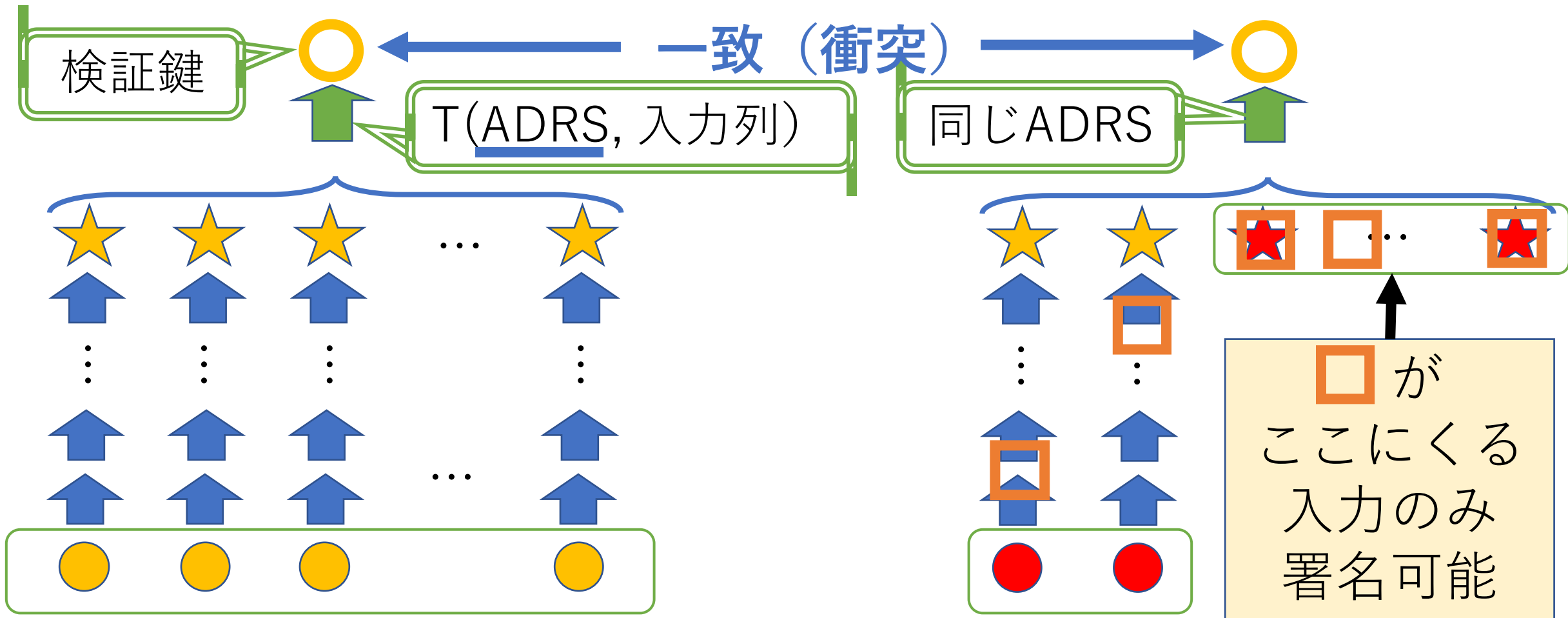
# 攻撃Step 1

ある場所のWOTS+について秘密鍵（もどき）を偽造

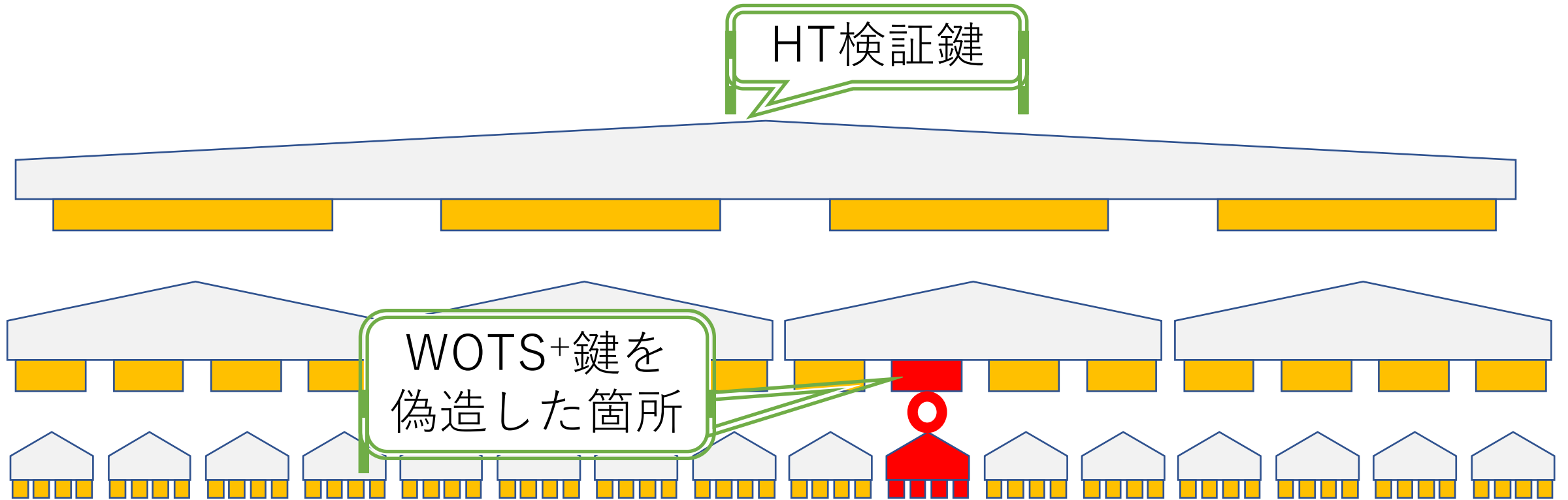



# 攻撃Step 1

ある場所のWOTS+について秘密鍵（もどき）を偽造

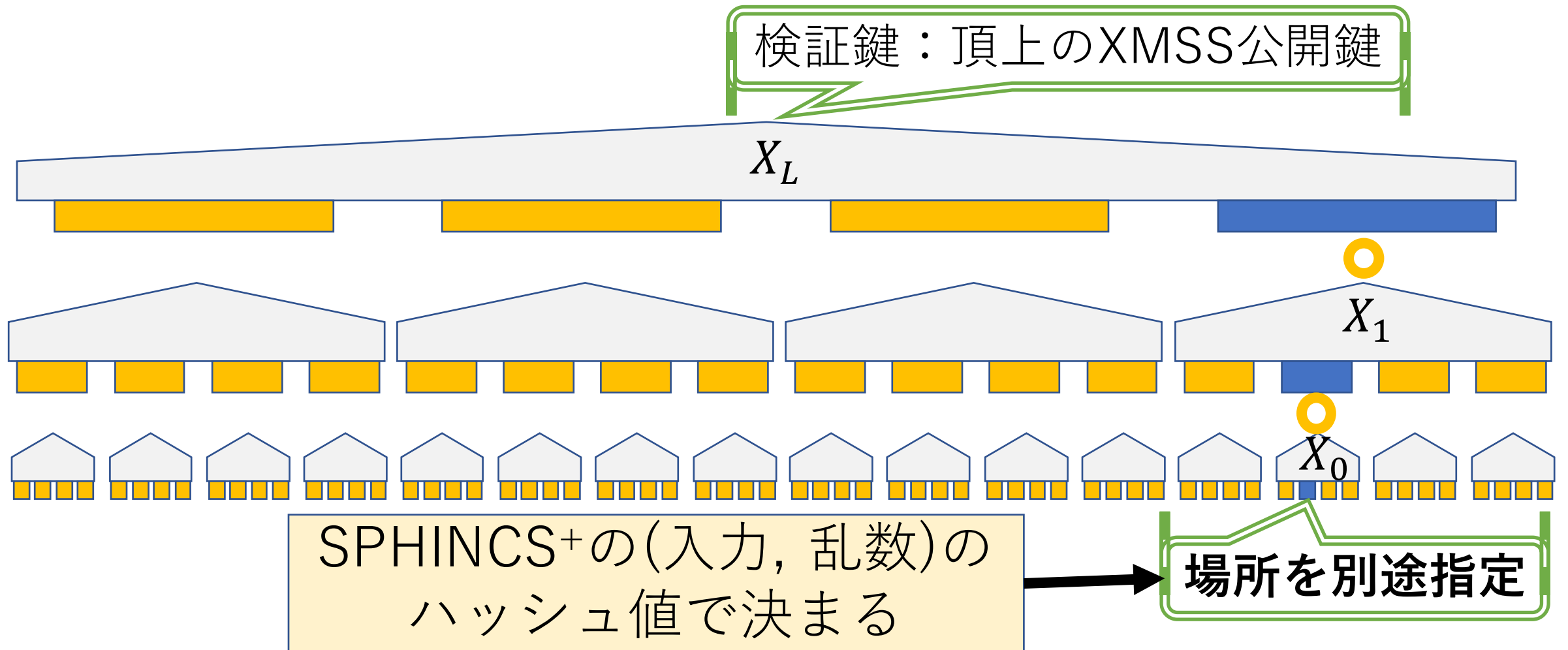


# 攻撃Step 2



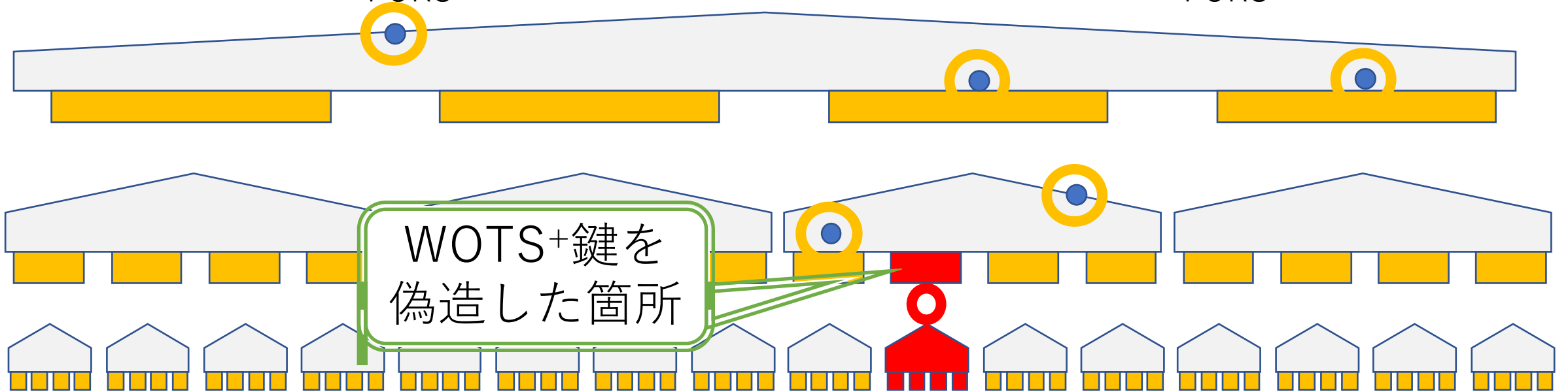
公開鍵  が偽造WOTS+鍵で署名可能となるように  
直下のXMSS署名鍵（最下位レイヤの場合はFORS署名鍵）を偽造

# 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  $sk_{\text{FORS}}$  で) 署名生成 → 偽造した鍵で  $pk_{\text{FORS}}$  を署名



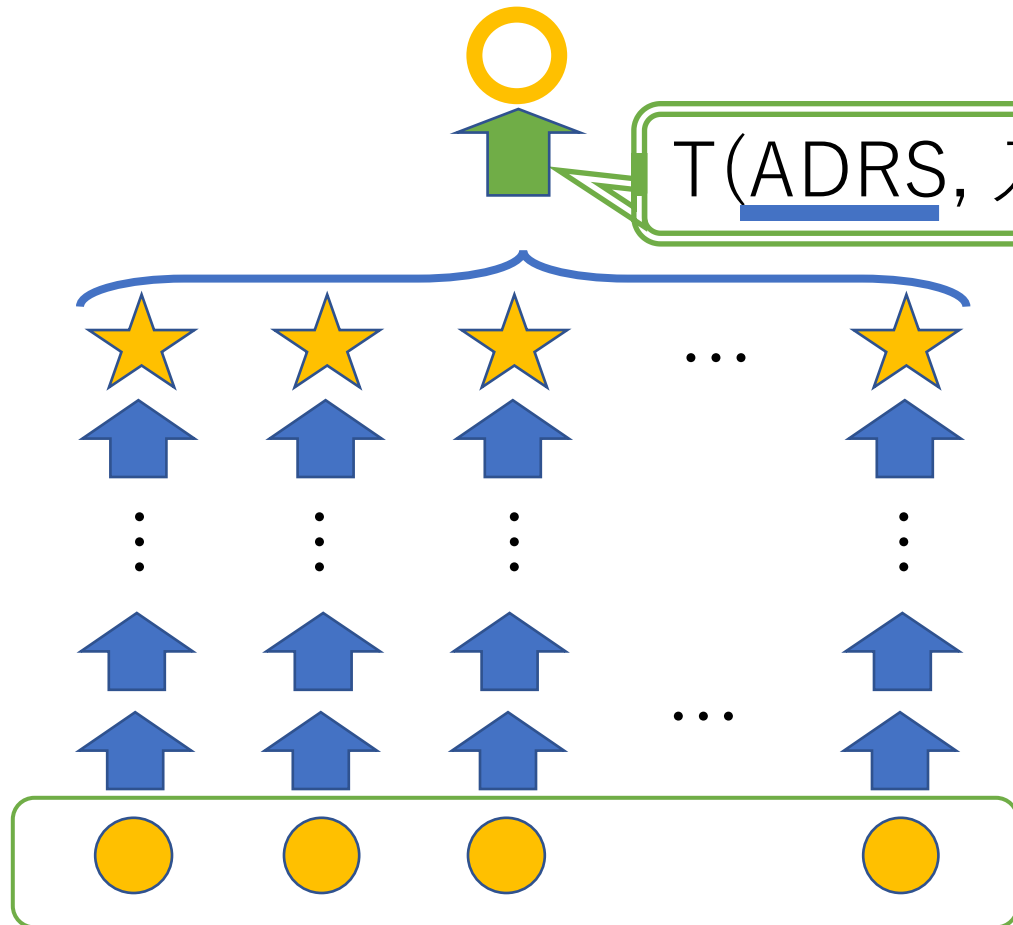
\* HTの署名生成に必要な残りの  は既知の署名から取ってくる

# 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
  - 検証鍵が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)  $\approx 2^{-68}$

# tweakable hashの構造 (概略)

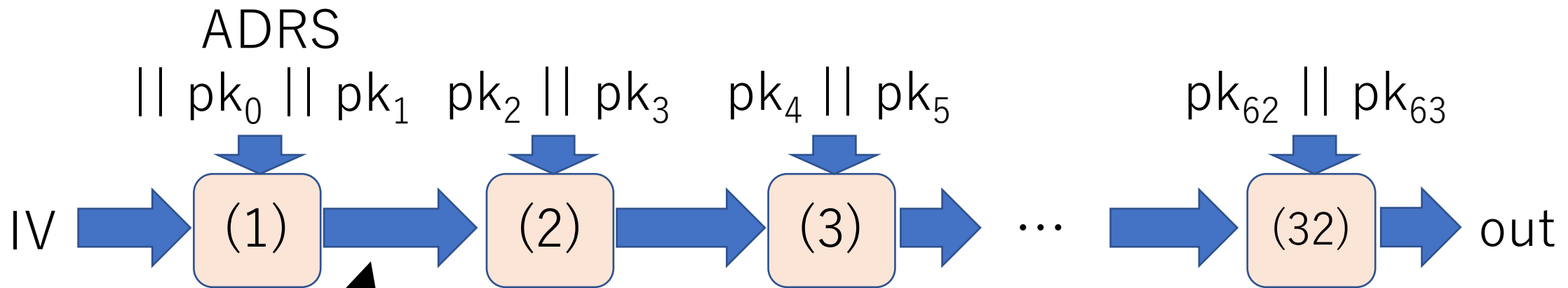


$\text{SHA-256}(\text{ADRS} \parallel \text{pk}_0 \parallel \text{pk}_1 \parallel \dots \parallel \text{pk}_{63})$

- \* 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 || pk<sub>0</sub> || pk<sub>1</sub> || ... || pk<sub>63</sub>)  
(Merkle-Damgard構成)

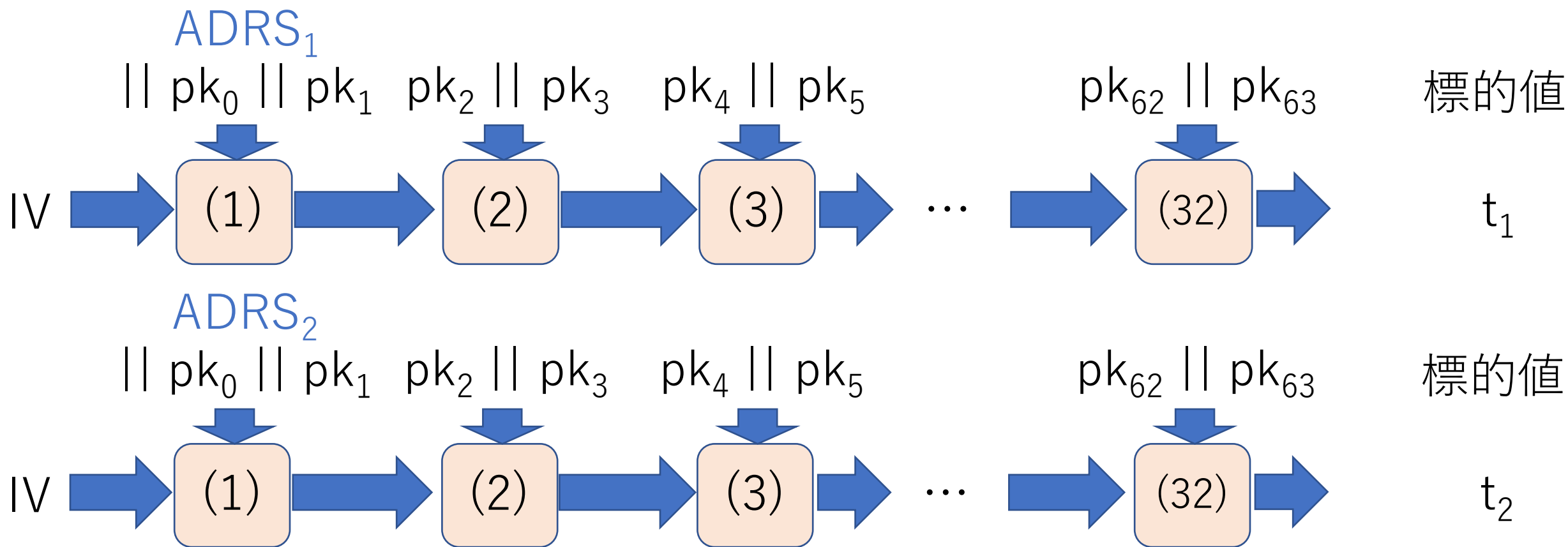


**256bit**の  
中間出力

\* 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



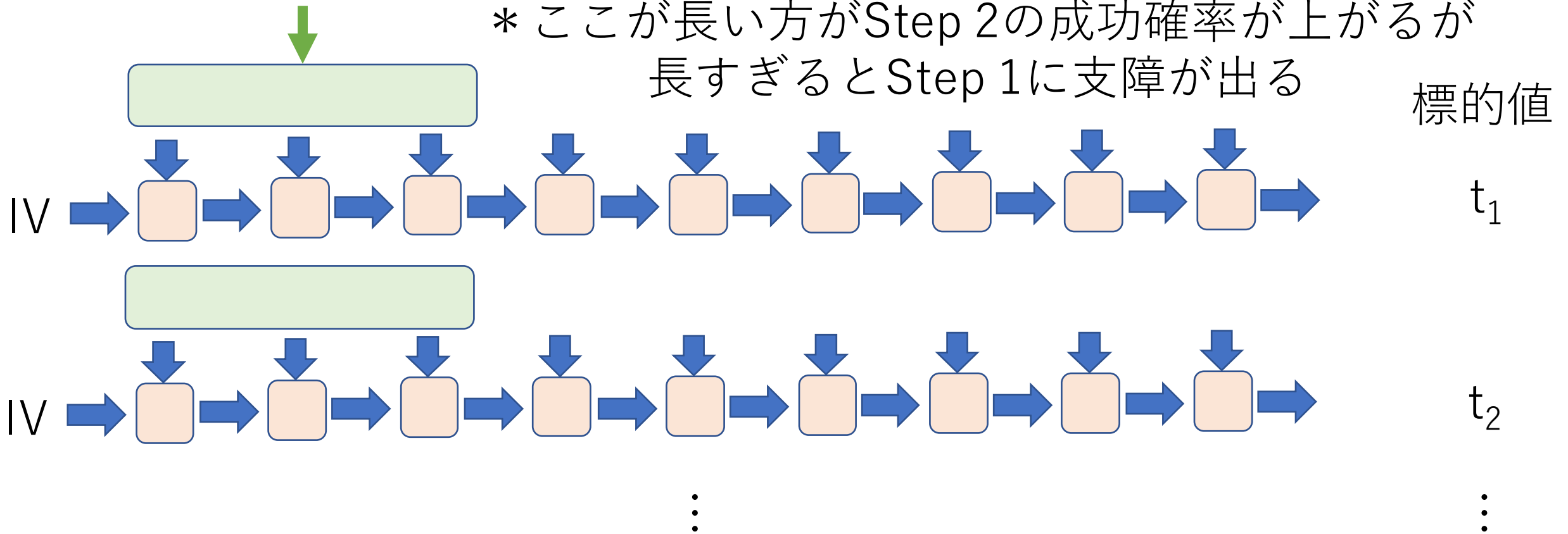
# distinct-function multi-target second-preimage (DM-SP) attack



# DM-SP attack

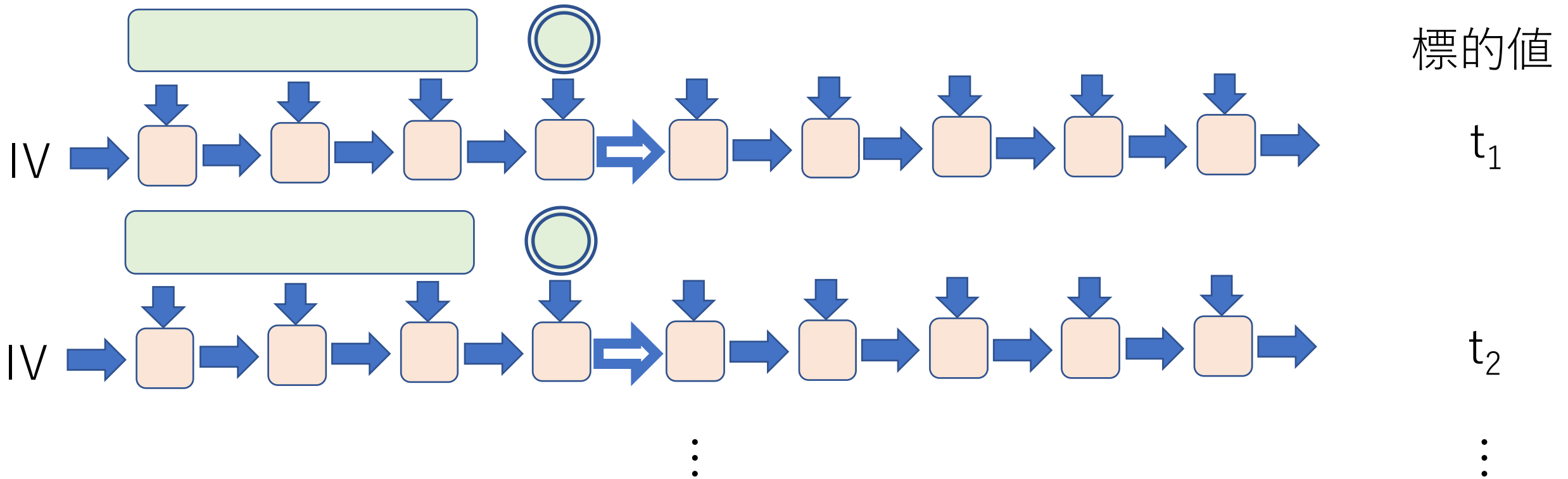
先頭の何か所かは（偽造した）WOTS+秘密鍵（の一部）から導出する

\* 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



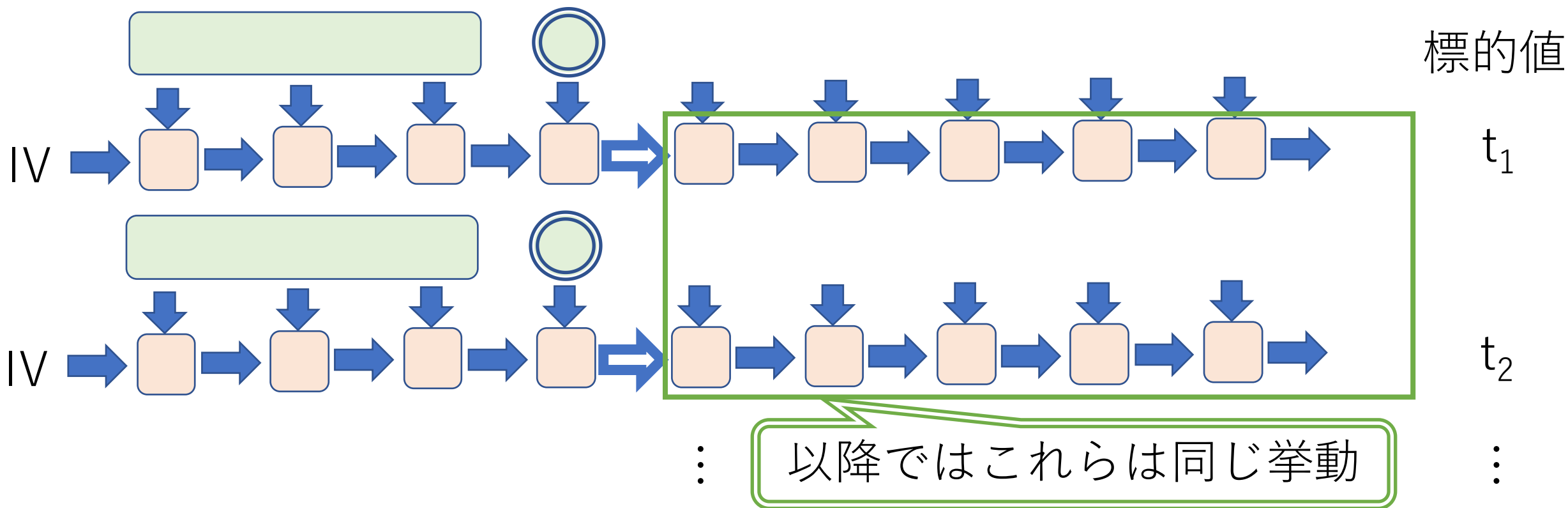
# DM-SP attack

⇒ 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



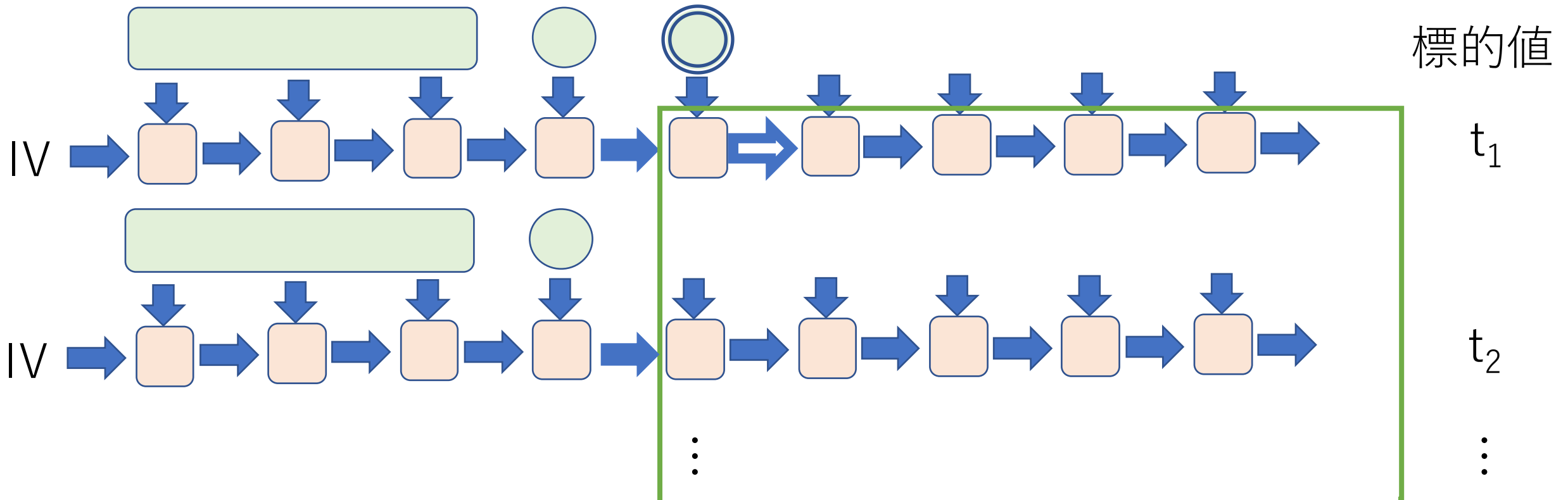
# DM-SP attack

⇒ 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



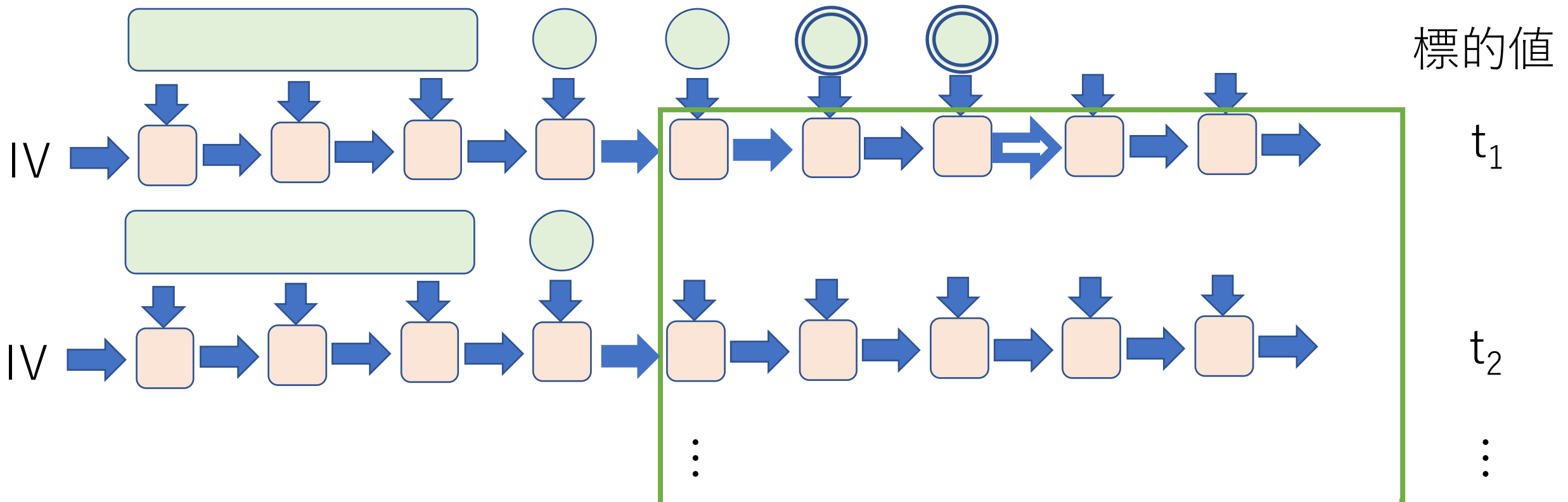
# DM-SP attack

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



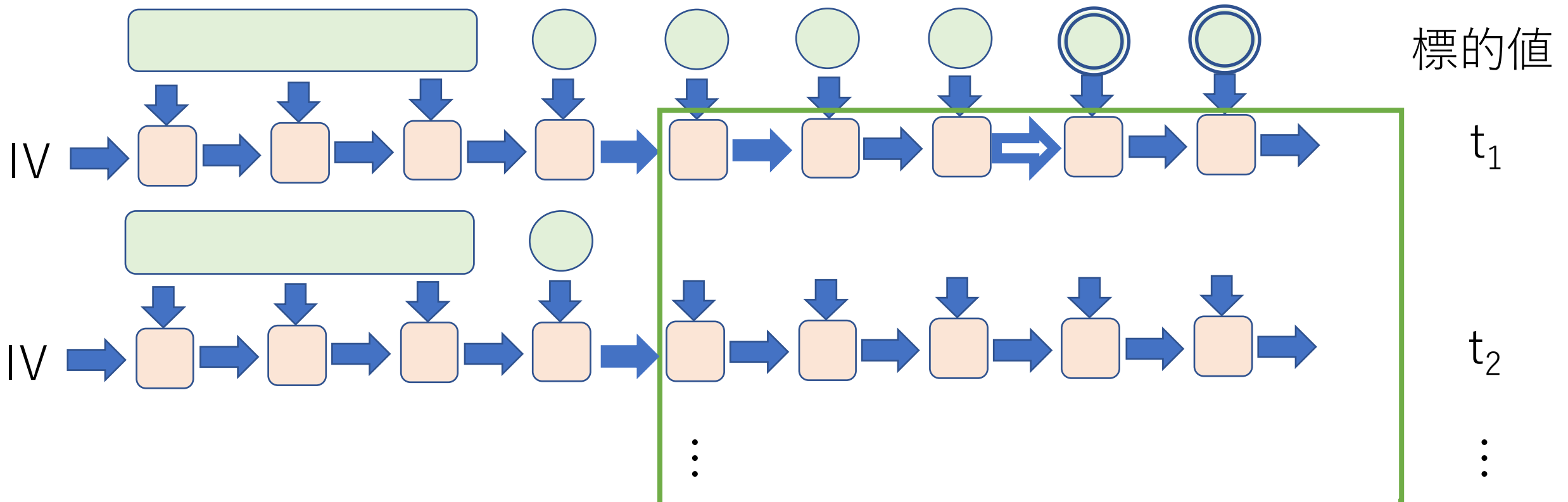
# DM-SP attack

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)



# 目次

- 概要：耐量子計算機暗号について
- 研究事例 1：署名方式UOVの改良  
[Furue et al., ASIACRYPT 2021]
- 研究事例 2：署名方式SPHINCS+の安全性解析  
[Perlner et al., PQCrypto 2022]
- **研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価**  
[Odagawa-N., arXiv 2021]

# 格子とその基底

- $\mathbb{R}$ 上一次独立なベクトル  $v_1, v_2, \dots, v_m \in \mathbb{R}^n$  を用いて

$$L = \left\{ \sum_{i=1}^m c_i v_i : c_i \in \mathbb{Z} \right\}$$

と表せる集合  $L \subseteq \mathbb{R}^n$  を ( $n$ 次元) 格子と呼ぶ

- $(v_1, v_2, \dots, v_m)$  をその基底と呼ぶ
  - 格子の基底の選び方は (順番や  $-1$  倍を除いても) 一般に複数ある
  - 以降では full-rank ( $m = n$ ) の場合のみを扱う

# 格子に関する計算問題の例

- 最短ベクトル問題 (SVP) : ノルム最小の非零格子点の計算
  - 入力は格子の基底によって与えられる
  - 近似版 ( $\gamma$ -SVP) や入力制限版 ( $\gamma$ -uSVP) などの亜種もある
- 最近ベクトル問題 (CVP) : 入力点に最も近い格子点の計算
- LWE (Learning with Errors) 問題:  
(概要) 誤差込みの有限体上の連立1次方程式を解く
  - 見た目は格子と関係なさそうだが、SVPに帰着可能
- これらの計算困難性が「格子暗号」の安全性の基盤

# 格子の基底簡約

- 入力された格子の基底に対し、より短い（ノルムが小さい）格子点からなる基底を計算する
- 格子基底簡約でSVPやCVPが直接解けることは稀だが、SVPやCVPの（近似）アルゴリズムの事前計算に用いられる
- 主な基底簡約アルゴリズム
  - LLLアルゴリズム [Lenstra et al., Math. Ann. 1982]
    - およびその亜種（DeepLLLなど）
  - BKZアルゴリズム
    - およびその亜種（BKZ 2.0、progressive BKZなど）
  - RSR（random sampling reduction）アルゴリズム

# 準備：Gram-Schmidt直交化

- ベクトルの列  $(b_1, b_2, \dots, b_n)$  から互いに直交するベクトルの列  $(b_1^*, b_2^*, \dots, b_n^*)$  を生成
- 各  $k$  について  $b_k = b_k^* + \sum_{i=1}^{k-1} \mu_{k,i} b_i^*$  を満たす ( $\mu_{k,i} \in \mathbb{R}$ )
  - ノルムは正規化しない

# LLLアルゴリズム

- 格子基底  $(b_1, b_2, \dots, b_n)$  を入力、パラメータ  $\delta \in (1/4, 1]$ 
  1.  $k \leftarrow 2$
  2.  $k \leq n$  である間以下を繰り返す：
    3. 各  $b_j$  から  $\lfloor \mu_{j,i} \rfloor b_i$  ( $1 \leq i < j$ ) を引き、 $\mu_{j,i}$  たちを更新
    4.  $\|b_k^*\|^2 \geq (\delta - \mu_{k,k-1}^2) \|b_{k-1}^*\|^2$  のとき、 $k \leftarrow k + 1$
    5. そうでなければ、 $b_k$  と  $b_{k-1}$  を交換 (&  $b_j^*, \mu_{j,i}$  たちを更新)  
おおよび  $k \leftarrow \max\{k - 1, 2\}$
  6.  $(b_1, b_2, \dots, b_n)$  を出力

# LLLアルゴリズムの計算量評価

- パラメータ  $\delta \leq 1$  が大きいほど良い出力が得られる
- 入力ベクトルのノルムの最大値を  $M$  とすると、  
 $\delta < 1$  (定数) のとき、計算量は多項式オーダー  $O(n^6(\log M)^3)$
- 一方、 $\delta = 1$  のときは多項式オーダーの評価は得られていない
  - $O(A^{n^3} \log M)$ 、 $A > (4/3)^{1/12}$  は定数 [Akhavi, Th. Comp. Sci. 2003]
  - 理由の直感的説明： $\delta < 1$  のときは、基底ベクトルのノルムの減少率が1未満の定数で抑えられる ( $\rightarrow$  ノルムが指数的に減少する) が、 $\delta = 1$  のときはそうできない
  - そのため実用上は  $\delta = 0.99$  など ad hoc なパラメータ選択をしている
- 課題： $\delta = 1$  における計算量評価を改良できないか？

# [Odagawa-N., arXiv:2105.14695]

- $\alpha = M^n / \text{vol}(L)$  とすると、LLL アルゴリズムのループ回数は

$$(n-1) \left( \frac{2}{\sqrt{5}} + 1 \right)^{n(n-2)} \alpha^{n-1} \left( \frac{3 \cdot (n+2)!}{8} \right)^{n/2}$$
$$\in \alpha^{n-1} \left( \left( \frac{2}{\sqrt{5}} + 1 \right) e^{-1/2} n^{1/2+o(1)} \right)^{n^2}$$

以下である（特に有限である）

- 証明の方針：ノルムがある値以下の格子点の個数の評価 + 次数で再帰



# 結果の比較

- $\text{vol}(L)$ が定数オーダーのとき、我々の上界は $(Mn)^{O(n^2)}$ 
  - $M = o(2^n/n)$ のとき、Ahkaviの上界よりも良い（小さい）
  - $M = \Omega(2^n)$ のとき、Ahkaviの上界よりも悪い（大きい）
    - 暗号の文脈では（残念ながら）こちらの場合の方が多い
- 我々の方針は他のLLL系のアルゴリズム（例：DeepLLL）にもほぼそのまま適用可能
  - Ahkaviの証明はLLLの場合に特化
- 課題：我々の方針をより精密化して上界を改良できないか？

# 目次

- 概要：耐量子計算機暗号について
- 研究事例 1：署名方式UOVの改良  
[Furue et al., ASIACRYPT 2021]
- 研究事例 2：署名方式SPHINCS+の安全性解析  
[Perlner et al., PQCrypto 2022]
- 研究事例 3：LLL系格子基底簡約アルゴリズムの計算量評価  
[Odagawa-N., arXiv 2021]