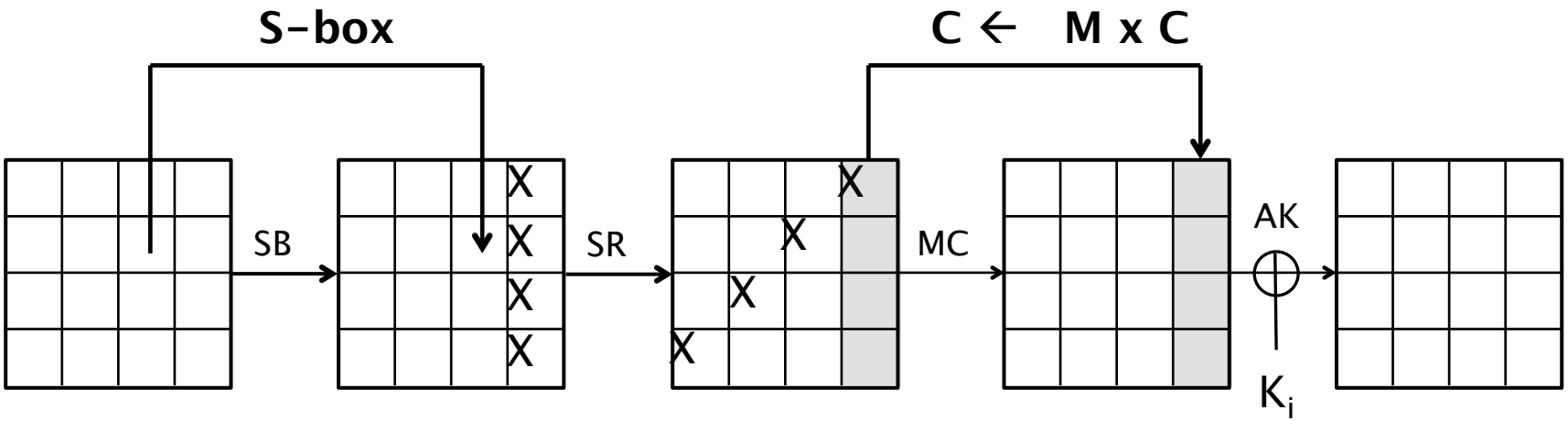
Généralisation des attaques de Demirci et Selçuk

Pierre-Alain Fouque
Ecole Normale Supérieure
(Travail en collaboration avec P. Derbez et J. Jean)

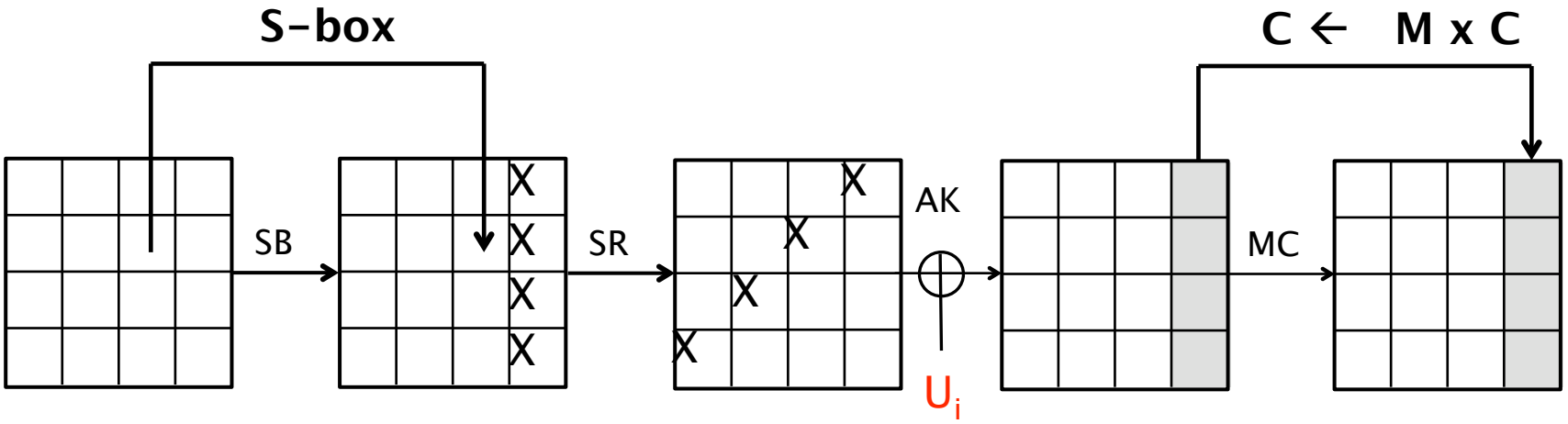
Advanced Encryption Standard

- Conçu par Daemen et Rijmen pour la compétition AES
- Choisi car très bonne propriétés de sécurité et bonne performence
- Réseau de SPN (substitution-permutation)
- Taille des blocs
  - — 128 bits, 192, or 256 bits
- Nombre de tours dépend de la taille de clé (10/12/14, respectivement)

Description de l'AES



Description de l'AES



Meet-in-the-middle attacks

- Récentes attaques par bicliques

Key size	Data	Time	Memory
128	2^{88}	$2^{126,2}$	2^{8}
192	2^{80}	$2^{189,4}$	2^{8}
256	2^{40}	$2^{254,4}$	2^{8}

- Gain marginal par rapport à la recherche exhaustive
- Peut-être une piste pour réellement casser l'AES ?

Meet-in-the-middle attacks

- Bouillaguet, Derbez et Fouque [C'11]:
  - classe d'attaques meet-in-the-middle basées sur la forme des équations décrivant l'AES
  - Conception d'un outil recherchant les meilleurs attaques
  - Très bon résultats sur 1, 2, 3 et 4 tours

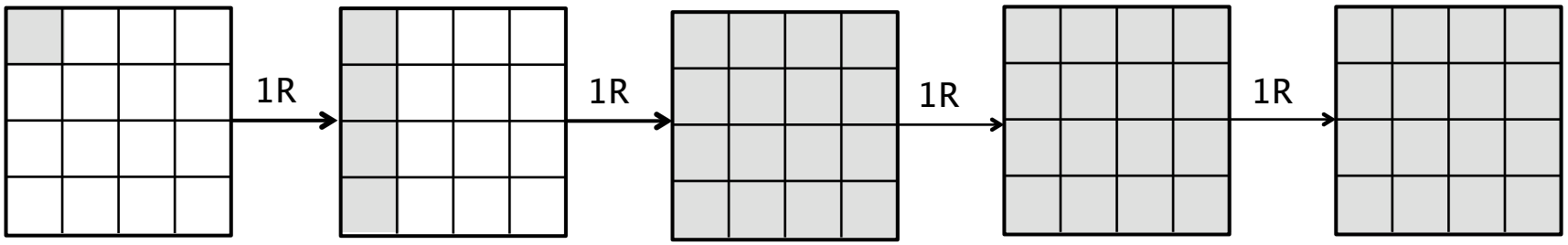
- Inconvénients :
  - Complexité de la recherche exponentielle en le nombre de variables
  - Prise en compte d'un nombre limité de messages

Meet-in-the-middle attacks

- A FSE 2008, Demirci et Selçuk ont présenté de nouvelles attaques meet-in-the-middle sur 7 et 8 tours d'AES
- Amélioration de l'attaque de Gibert et Minier
- Asiacrypt 2010, Dunkelman, Keller et Shamir améliorent ces résultats et obtiennent les meilleurs attaques connues sur 7 et 8 tours d'AES

Demirci-Selçuk Attack

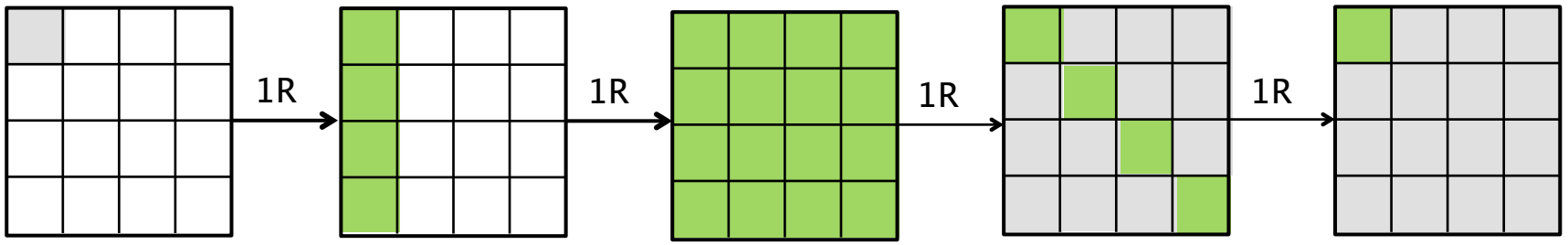
Considérons un ensemble de 256 messages de la forme :



- Pour chaque octet du dernier état, la suite ordonnée des 256 valeurs de cet octet est entièrement déterminée par 25 paramètres de 8 bits
- Donc, seulement 2^{200} suites possibles au lieu de 2^{2048}
- Les 25 paramètres sont faciles à obtenir

Demirci-Selçuk Attack

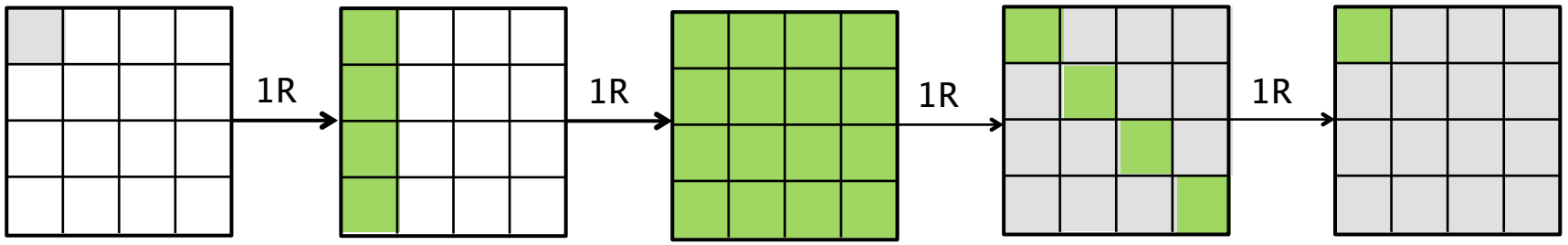
Considérons un ensemble de 256 messages de la forme :



- Pour chaque octet du dernier état, la suite ordonnée des 256 valeurs de cet octets est entièrement déterminée par 25 paramètres de 8 bits
- Donc seulement 2^{200} suites possibles au lieu de 2^{2048}
- Les 25 paramètres sont faciles à obtenir

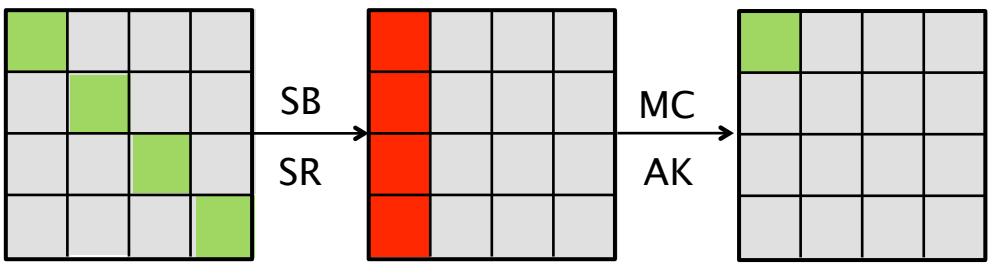
Demirci-Selçuk Attack

Considérons un ensemble de 256 messages de la forme :



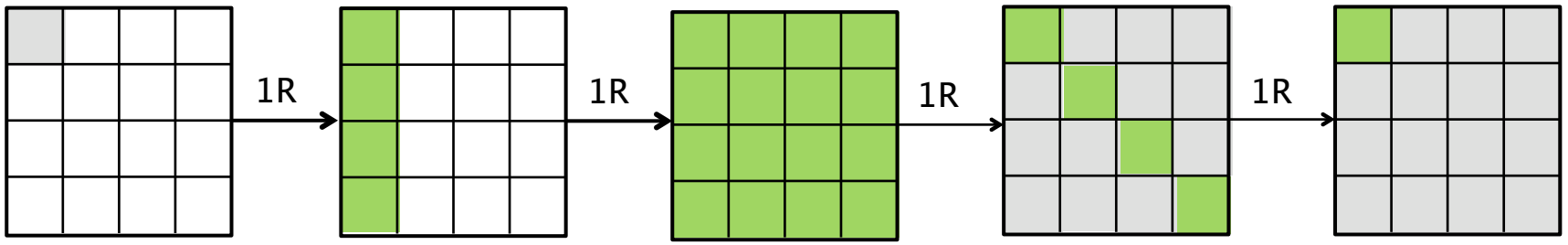
- Pour chaque octets du dernier état, la suite ordonnée [illegible] est
- [illegible]
- Les 25 para[illegible]tenir.

Quatrième tour

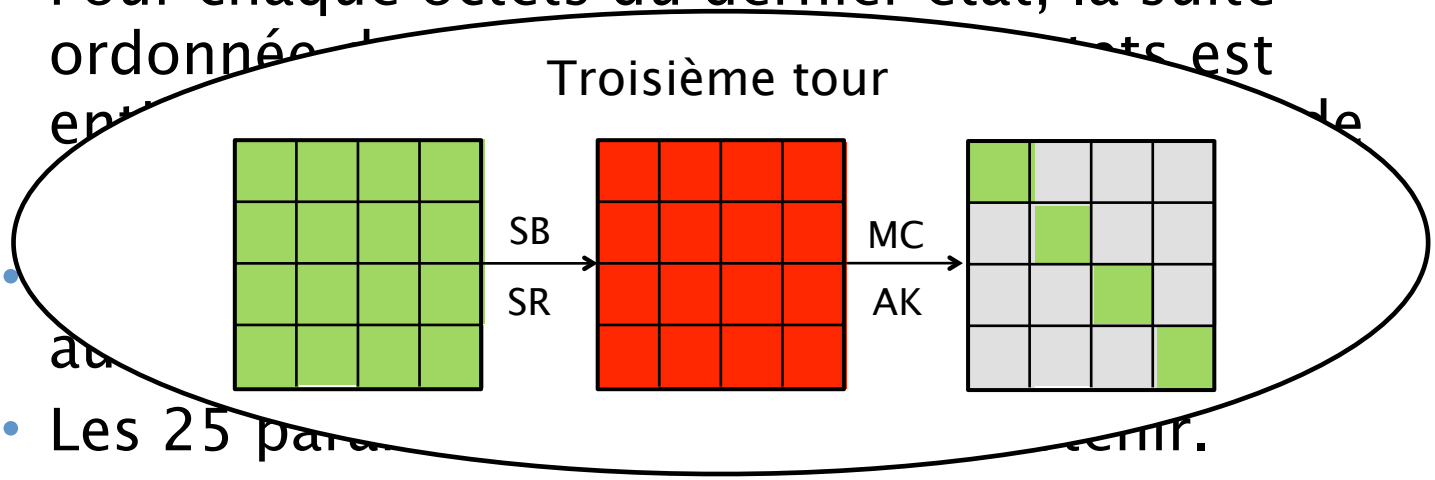


Demirci-Selçuk Attack

Considérons un ensemble de 256 messages de la forme :

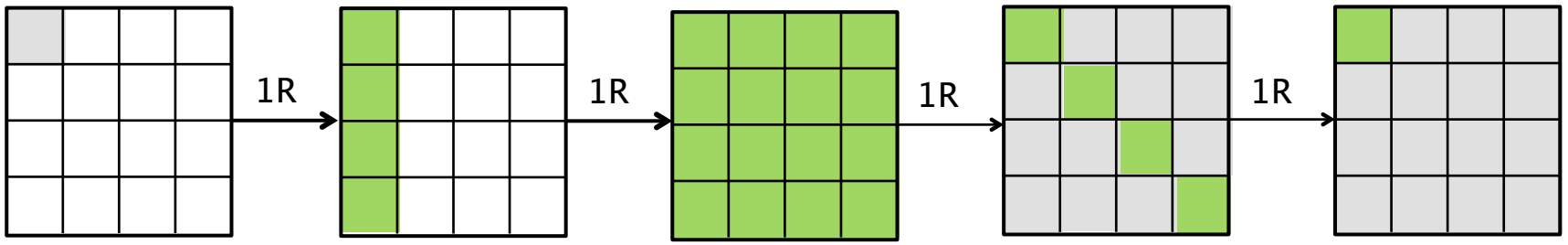


- Pour chaque octets du dernier état, la suite ordonnée [illegible] est
- [illegible]
- Les 25 [illegible]



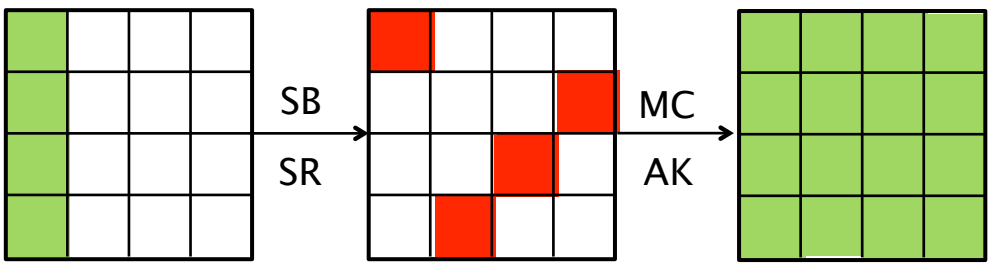
Demirci-Selçuk Attack

Considérons un ensemble de 256 messages de la forme :



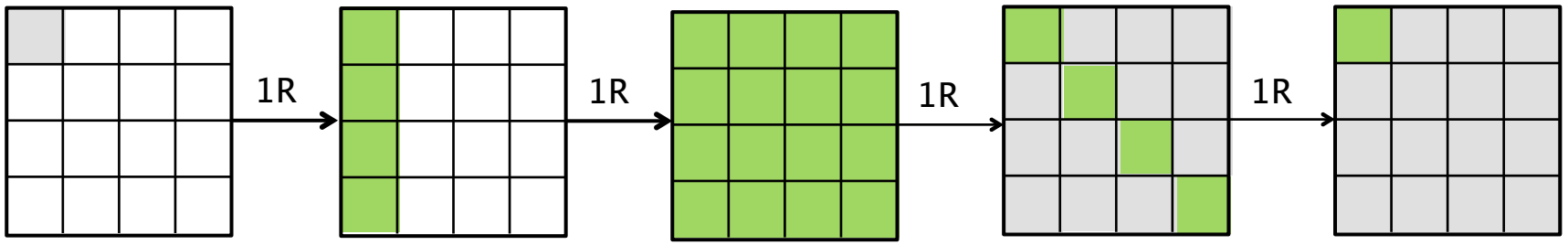
- Pour chaque octets du dernier état, la suite ordonnée [illegible] est
- [illegible]
- Les 25 para[illegible]tenir.

Deuxième tour

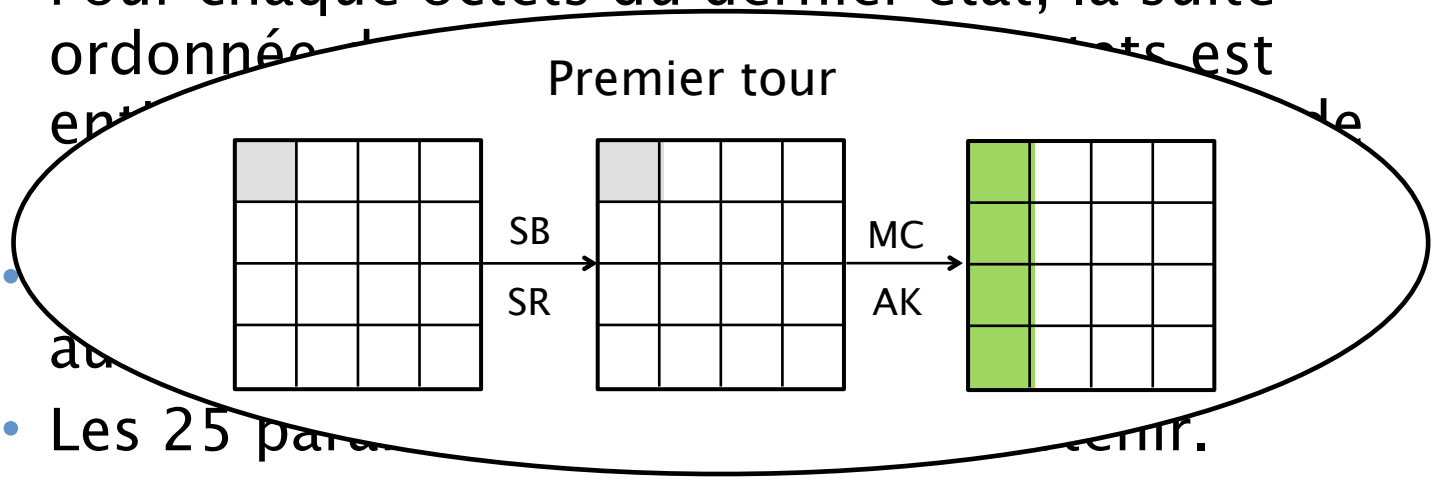


Demirci-Selçuk Attack

Considérons un ensemble de 256 messages de la forme :



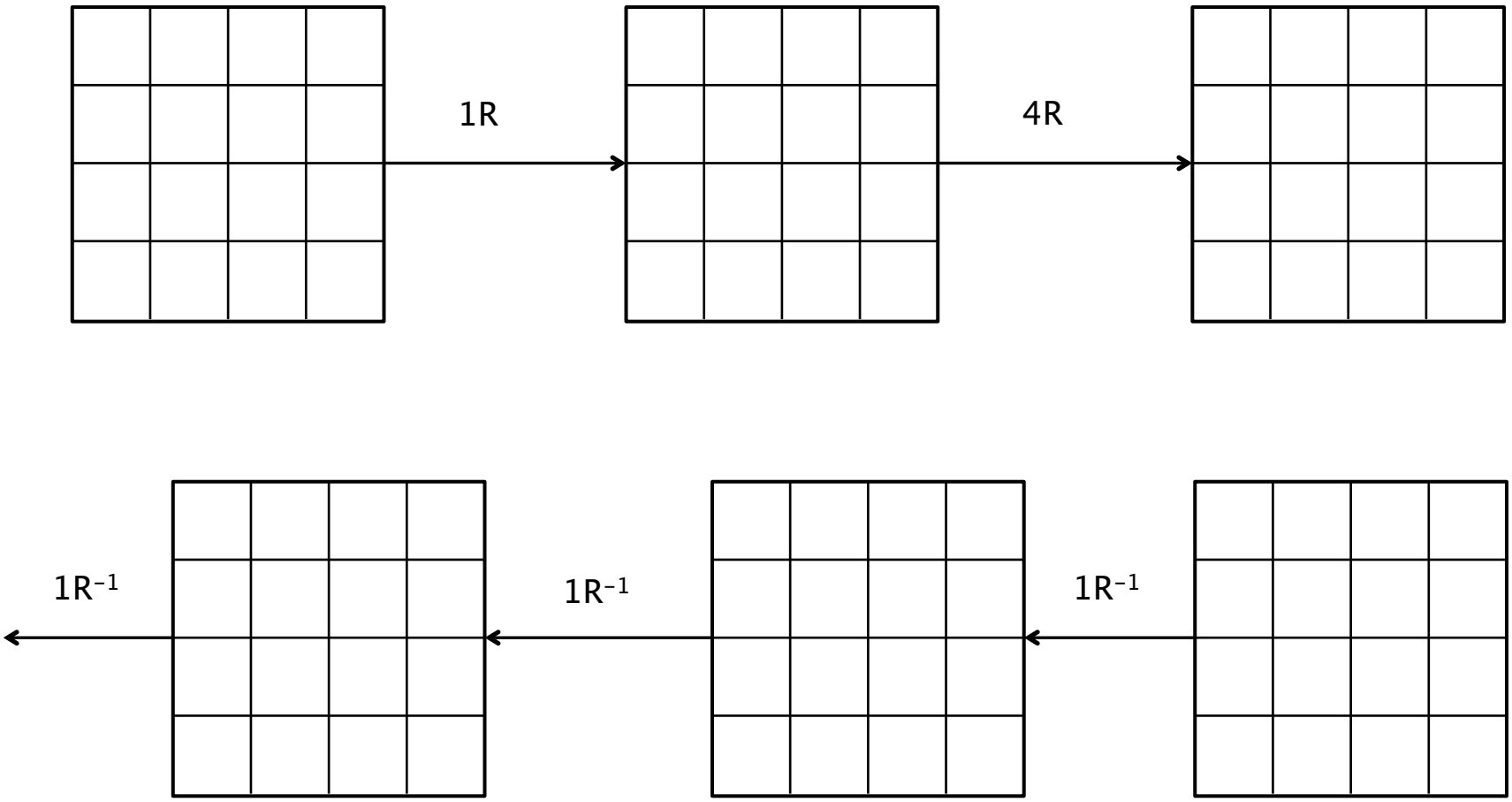
- Pour chaque octets du dernier état, la suite ordonnée [illegible] est
- [illegible]
- Les 25 para[illegible]tenir.



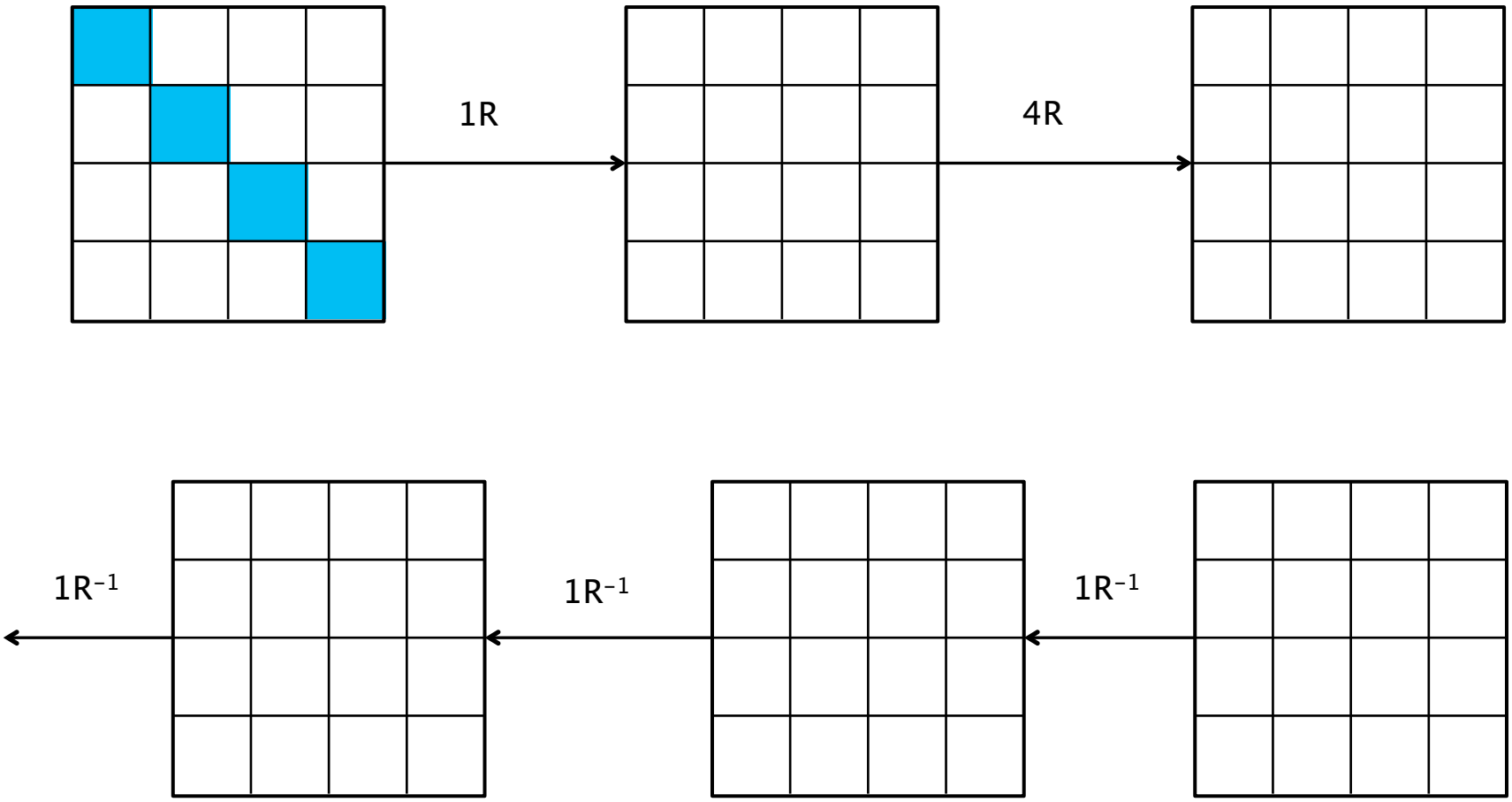
Demirci-Selçuk Attack

- L'idée est de construire ces 2^{200} suites et de les stocker dans une table de hachage
- Dans la phase online, l'adversaire déchiffre partiellement un bon ensemble de 256 messages et vérifie si la suite est dans la table
- La probabilité qu'une mauvaise clé passe ce test est négligeable

Demirci–Selçuk Attack

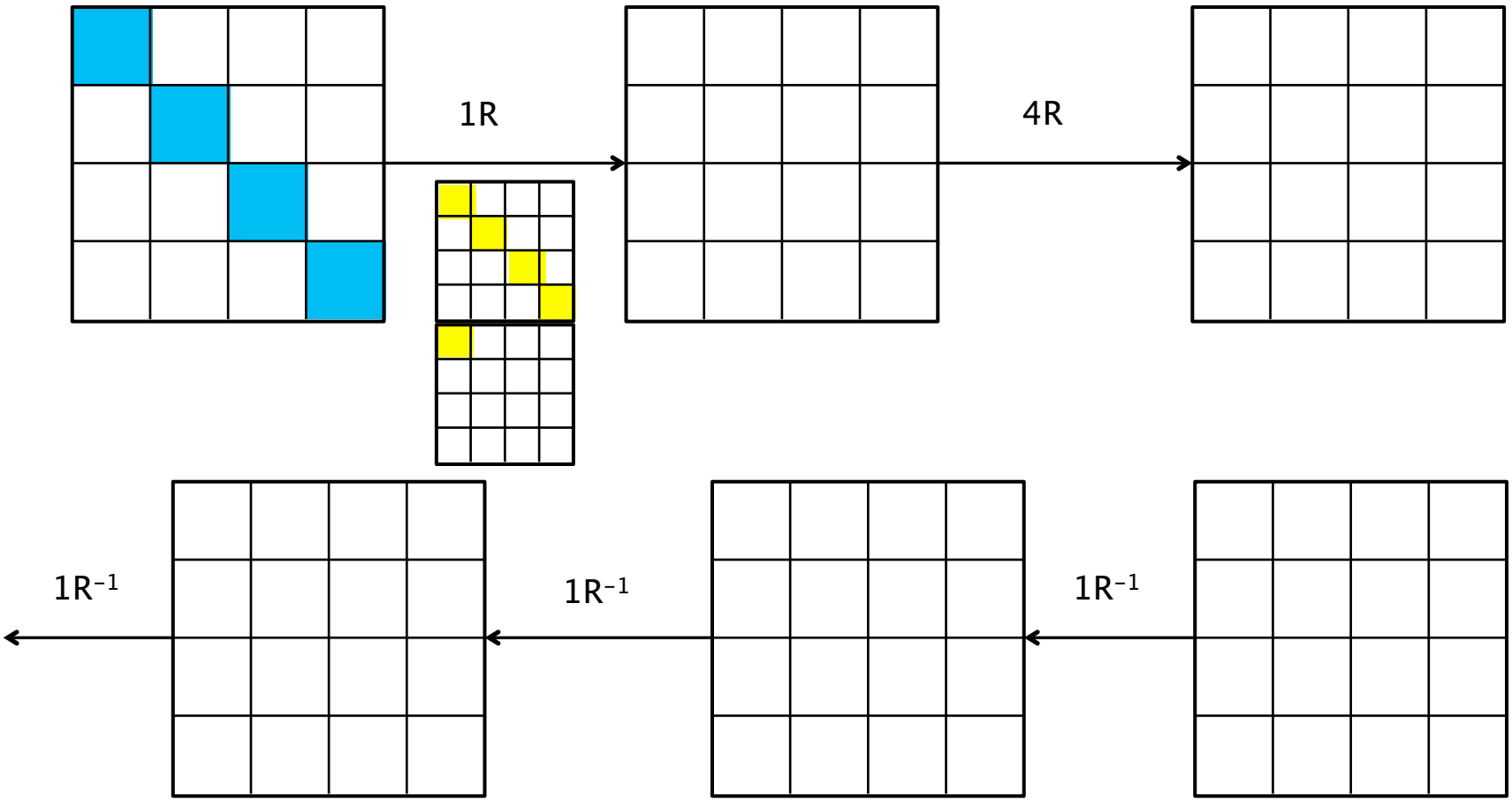


Demirci–Selçuk Attack



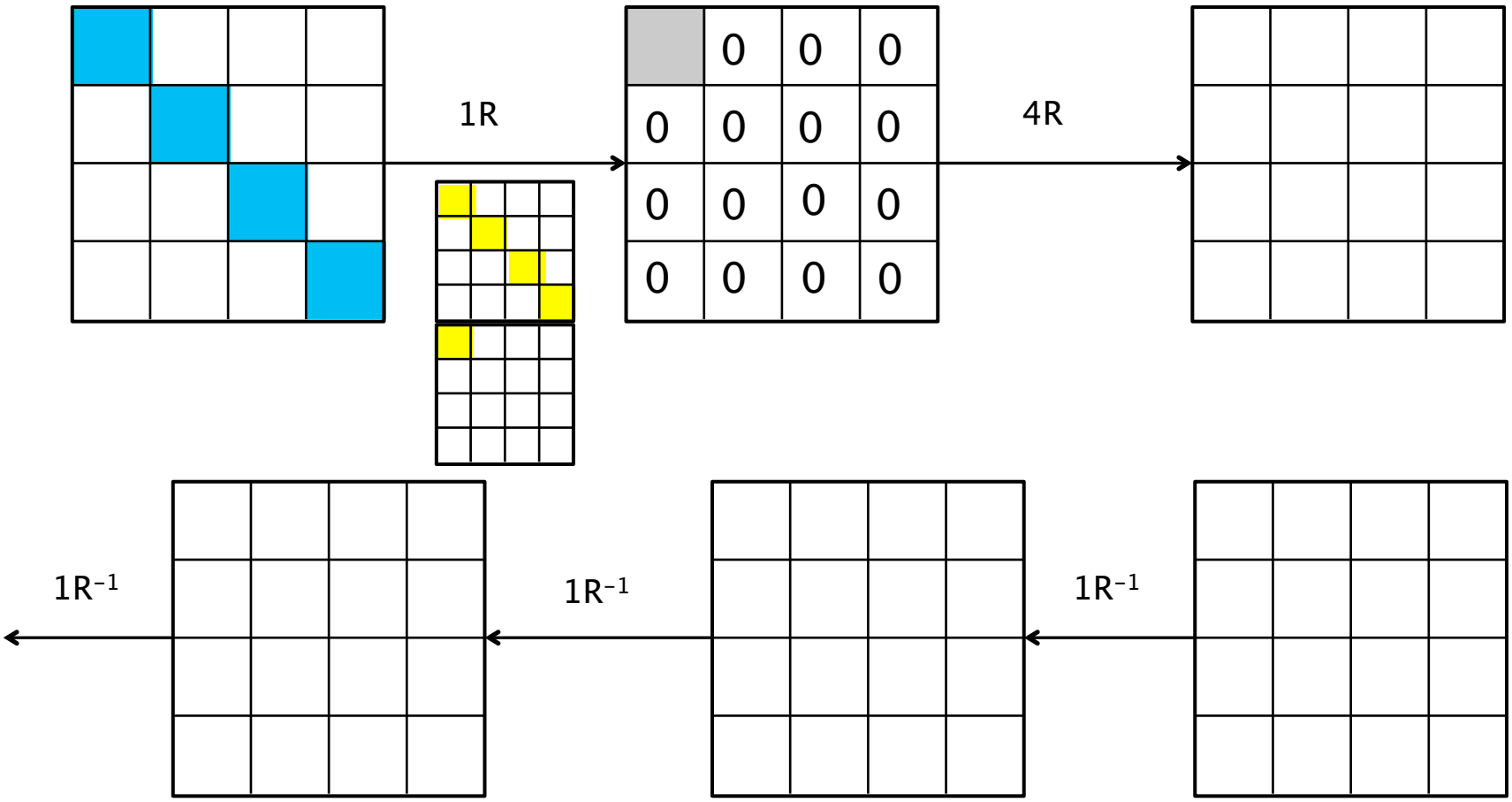
- Demander une structure 2^{32} messages.

Demirci–Selçuk Attack



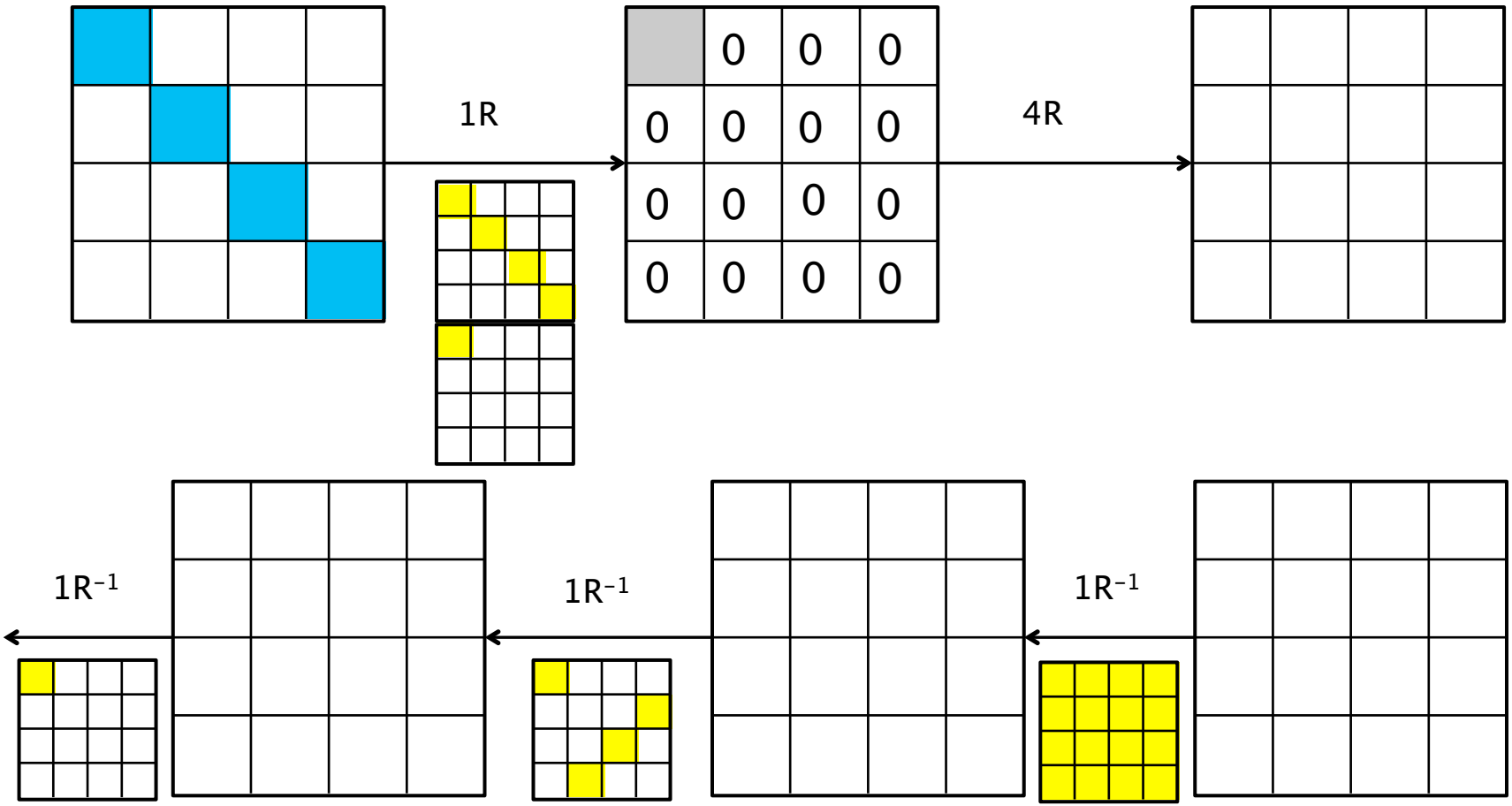
- Deviner 4 octets de K_0 et un de K_1.

Demirci–Selçuk Attack



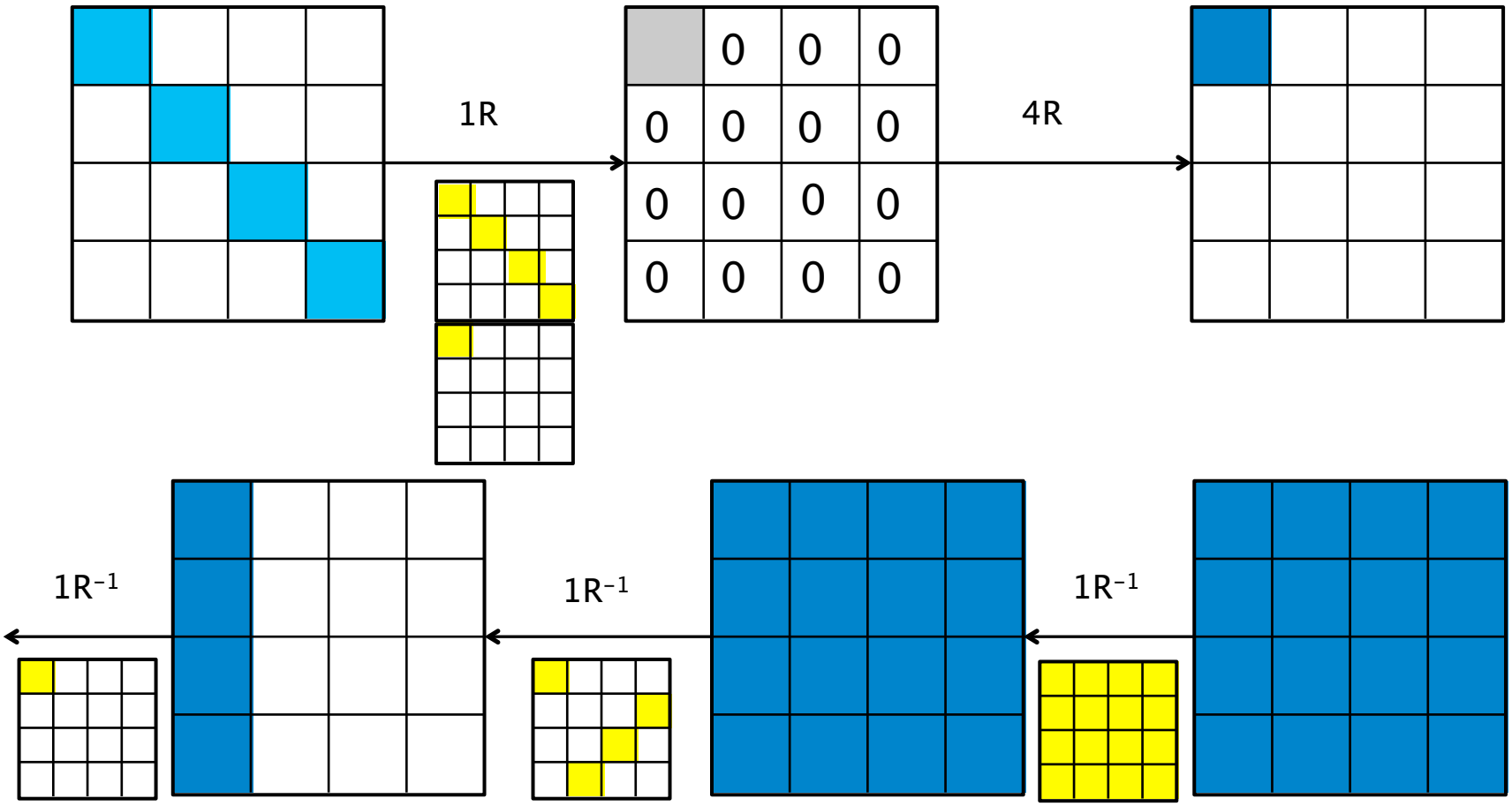
- Identifier un ensemble de 256 de la bonne forme

Demirci–Selçuk Attack



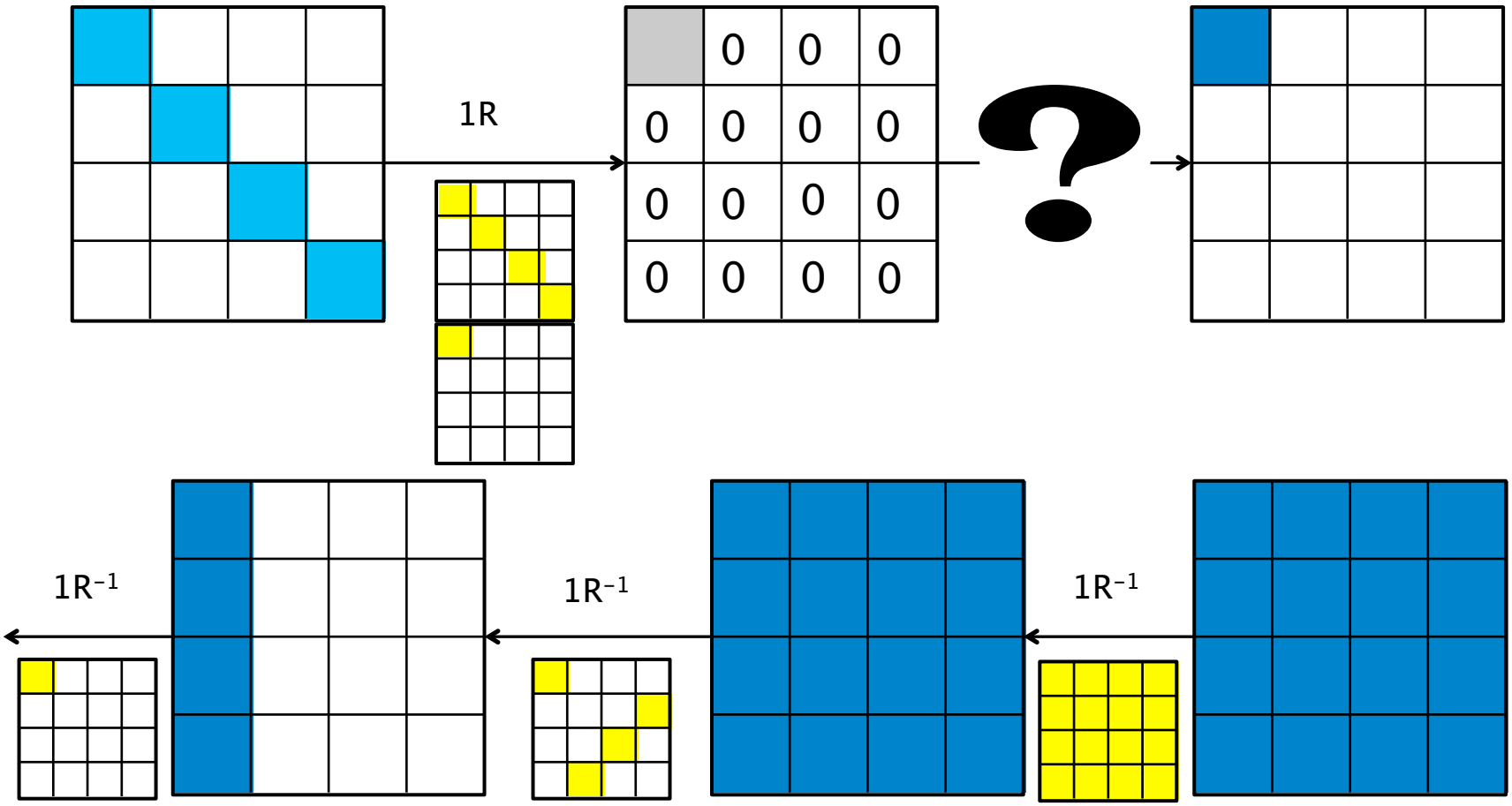
- Deviner 16 octets de U_8, 4 de U_7, et 1 de U_6.

Demirci-Selçuk Attack



- Déchiffrer partiellement les chiffrés.

Demirci–Selçuk Attack



- Vérifier si la séquence est valide.

Demirci-Selçuk Attack

- Réduire la taille de la table en ne stockant qu'une fraction α des séquences possibles
- En contrepartie, l'attaque devra être répétée, multipliant ainsi la complexité en données et en temps par un facteur 1/α
- L'attaque devient probabiliste
- Résultats :

Version	Rounds	Data	Memory (Pre.)	Time	MinMax
AES-192	7	2^{46+n}	2^{192-n}	2^{94+n}	2^{143}
AES-256	7	2^{34+n}	2^{204-n}	2^{82+n}	2^{143}
AES-256	8	2^{34+n}	2^{206-n}	$2^{205,6+n}$	$2^{205,8}$

Commentaire sur Demirci–Selçuk's Attack

- Une structure de 2^{32} messages se divise en 2^{24} ensembles de 256 messages de la bonne forme
- Les données peuvent donc être réutilisée

- On peut diviser la taille de la table par 256 :

$$f_{c_1,\ldots,c_{25}}(i) = g_{c_1,\ldots,c_{24}}(i) + c_{25}$$

Stocker la séquence $[f(0) + f(0), \ldots, f(0) + f(255)]$

Multiset tabulation

- Dans la phase online, l'octet de K_1 est deviné uniquement pour ordonner la séquence

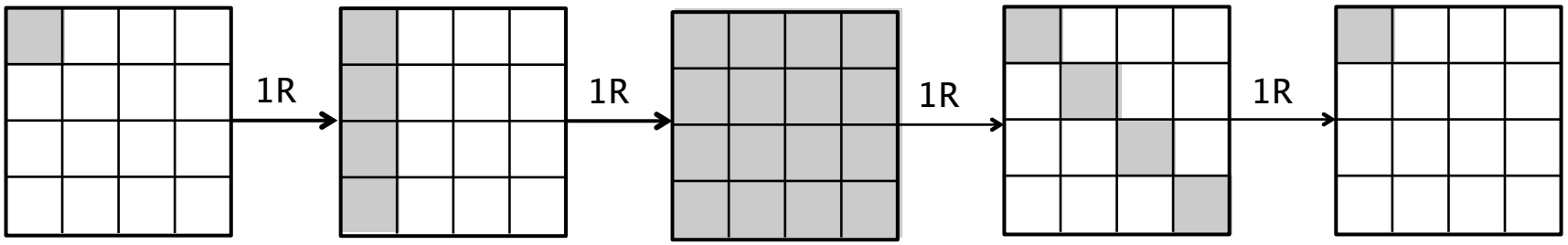
- Idée : considérer les suites non-ordonnées

- Double avantage :
  - Economie d'un guess dans la phase online
  - Petit gain mémoire

- Inconvénient : # suite. non-ordonnées << # seq. ordonnées

mais suffisant dans ce cas

Enumération différentielle



- Seulement 2^{64} valeurs possibles pour l'état avant/après le troisième SubByte.
- 16 paramètres parmi les 24 nécessaires.

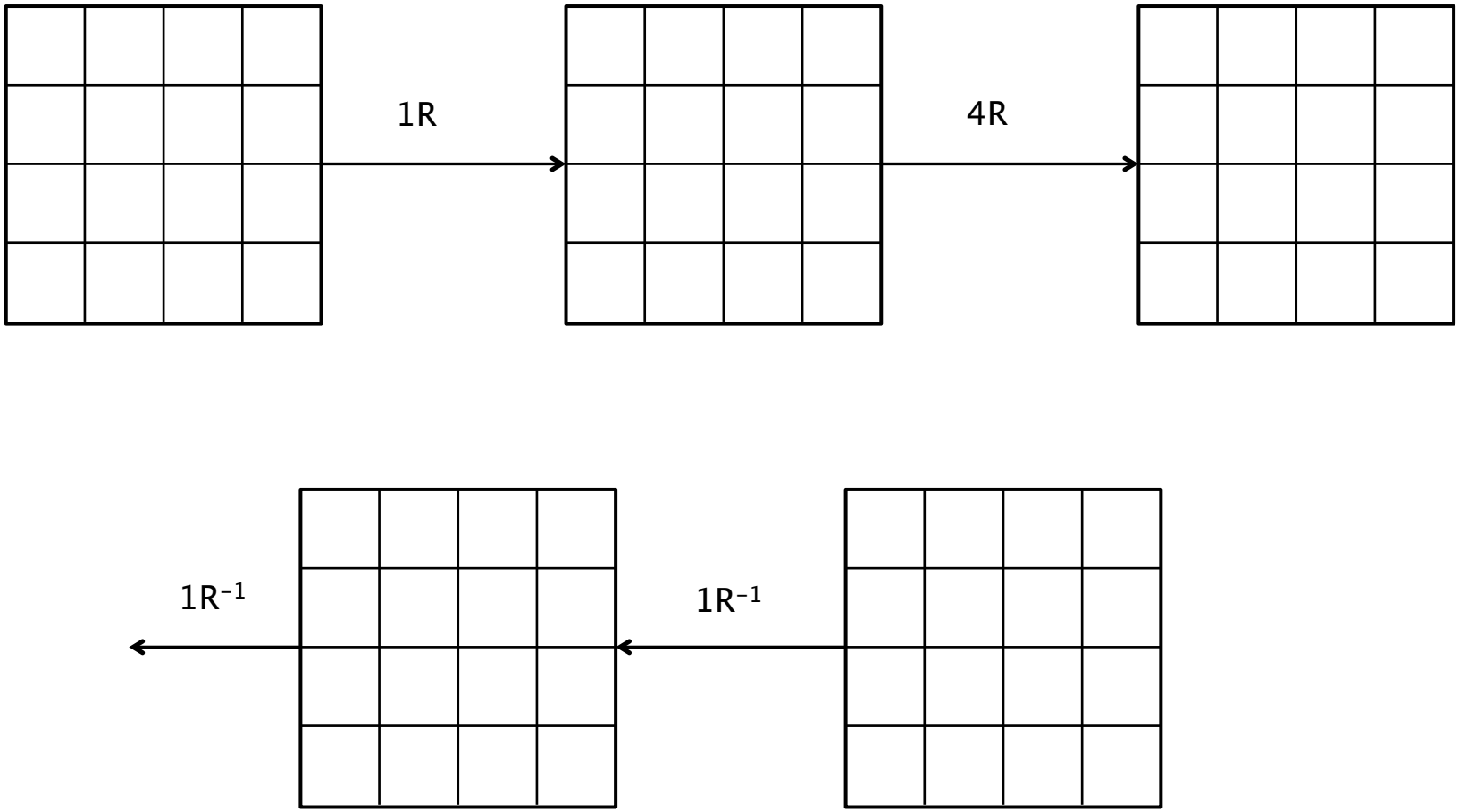
⟹ Nombre de séquences possibles réduit à 2^{129}

- Facteur 2 dû aux deux cas $P_0 = P$ et $P_0 = P'$ dans la construction de la séquence.

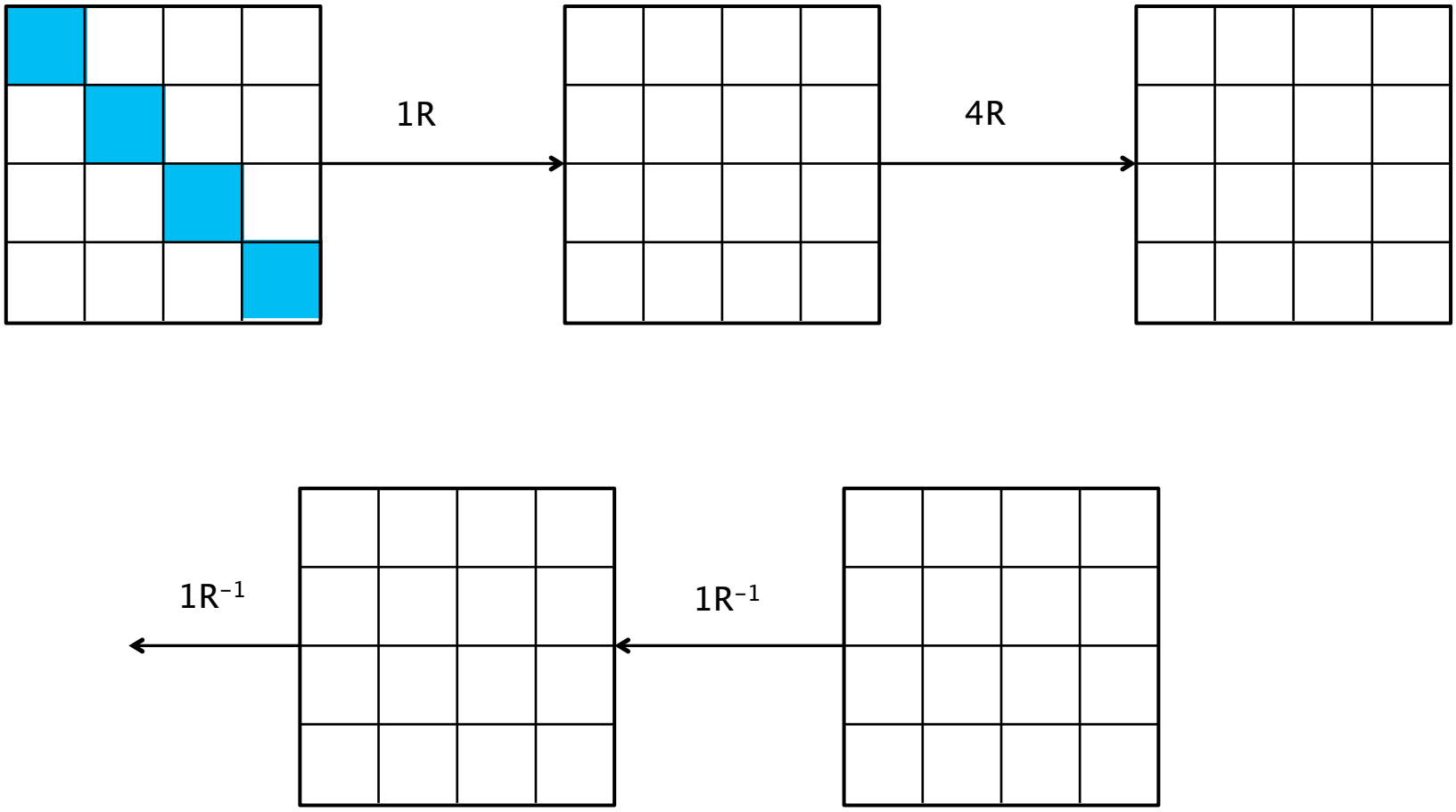
Enumération différentielle

- En contrepartie, on ne peut utiliser que des paires vérifiant ce chemin différentiel
- Comme la paire est contenue dans une structure et que la probabilité de la transition est 2^{-120}, 2^{105} structures sont nécessaires (chacune contenant 2^{15} candidats)
- Identifier les paires est relativement facile

7-Round Attack

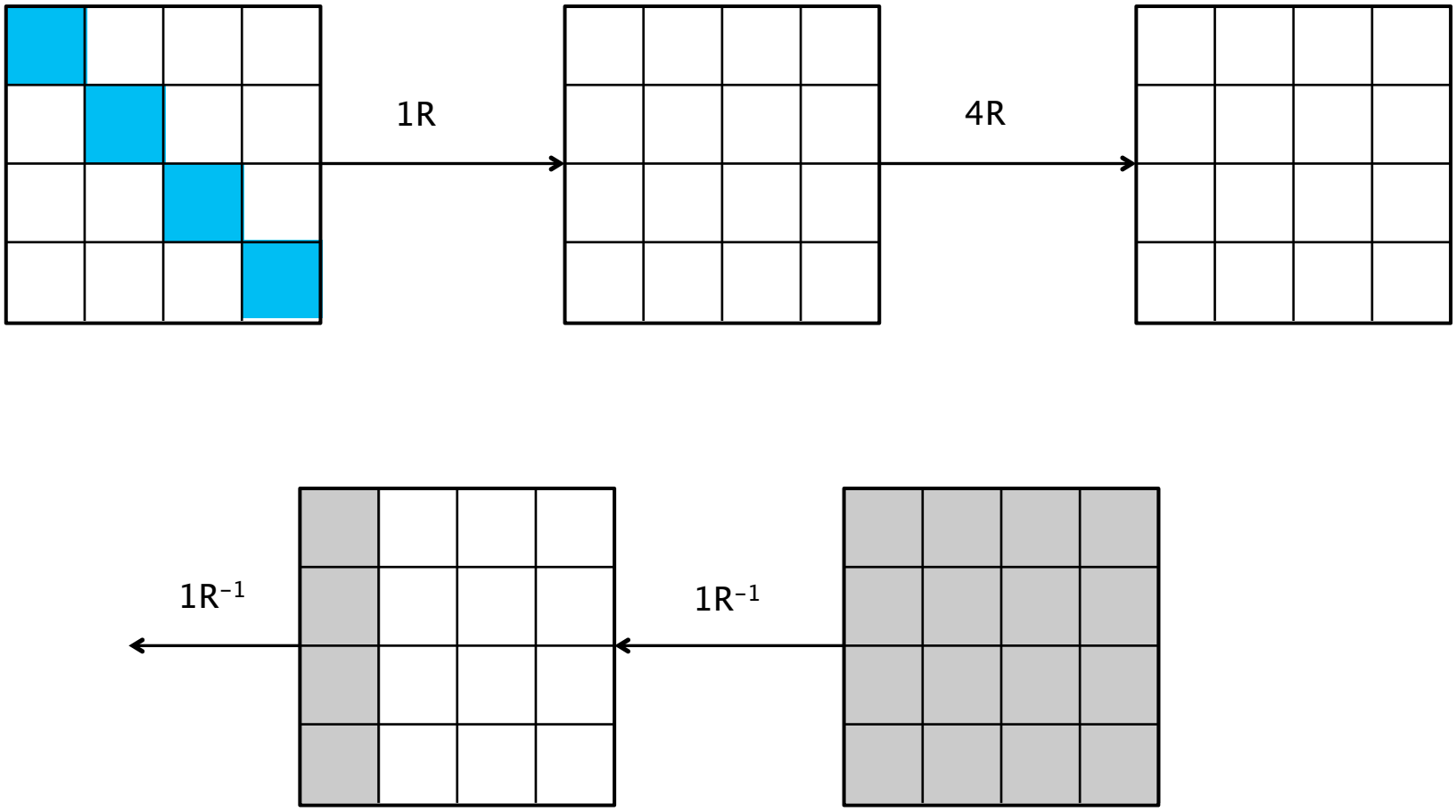


7-Round Attack



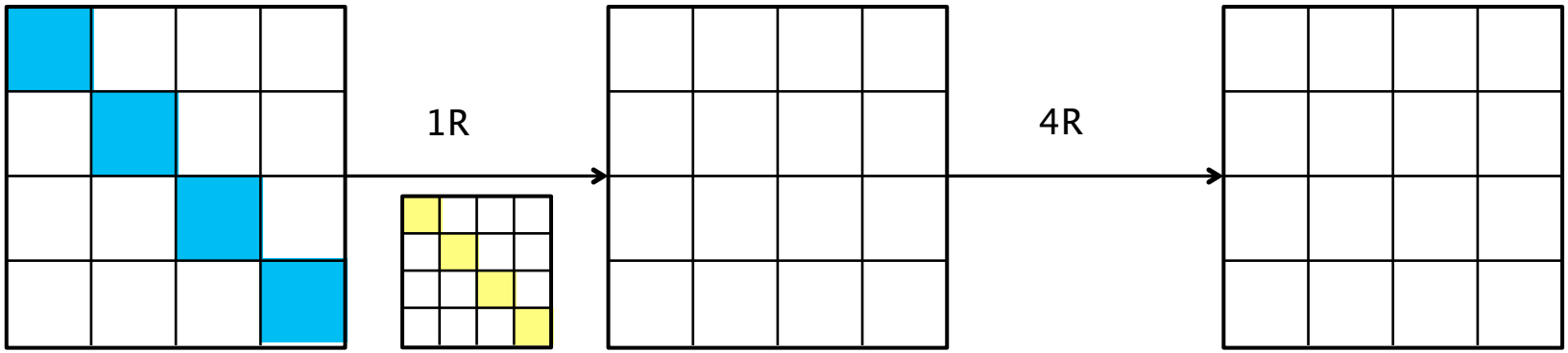
- Demander une structure de 2^{32} messages

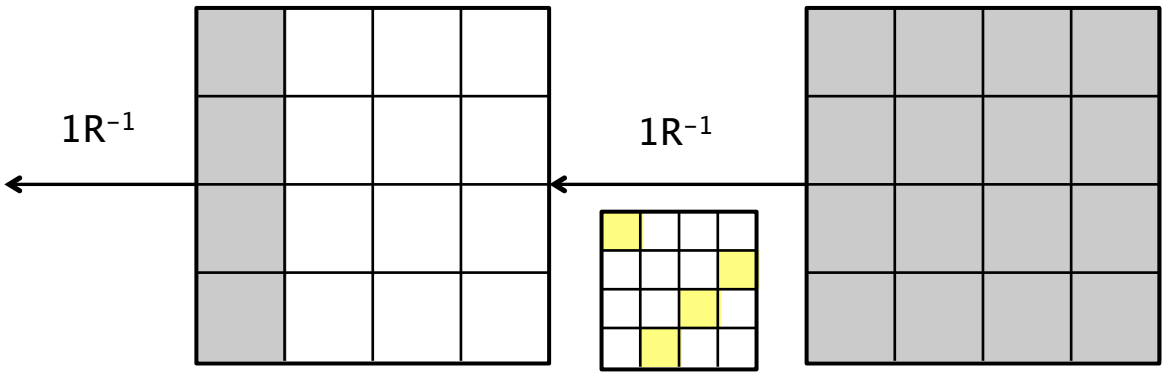
7-Round Attack



- Identifier les paires vérifiant la transition 4 → 16.

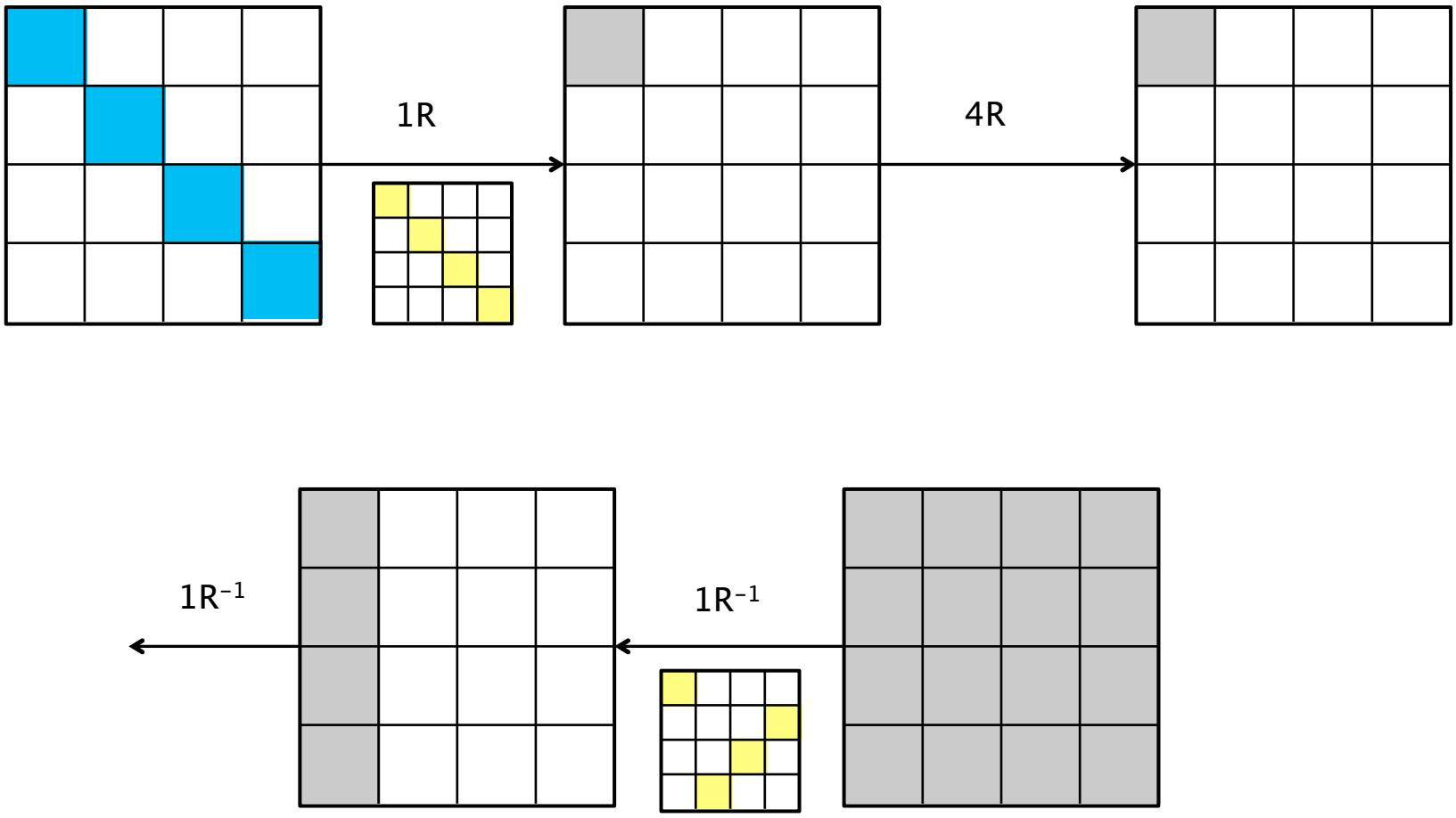
7-Round Attack





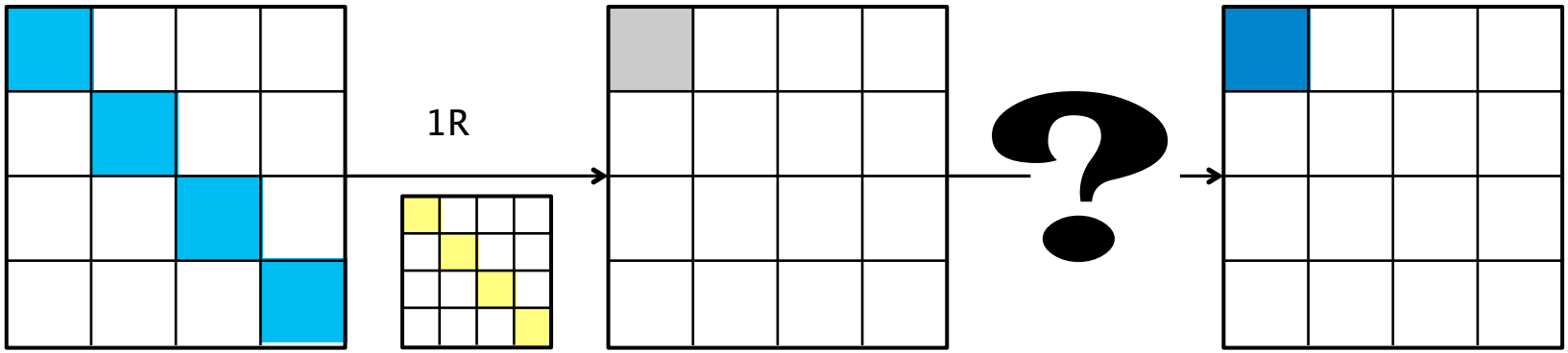
- Identifier les paires vérifiant la transition.

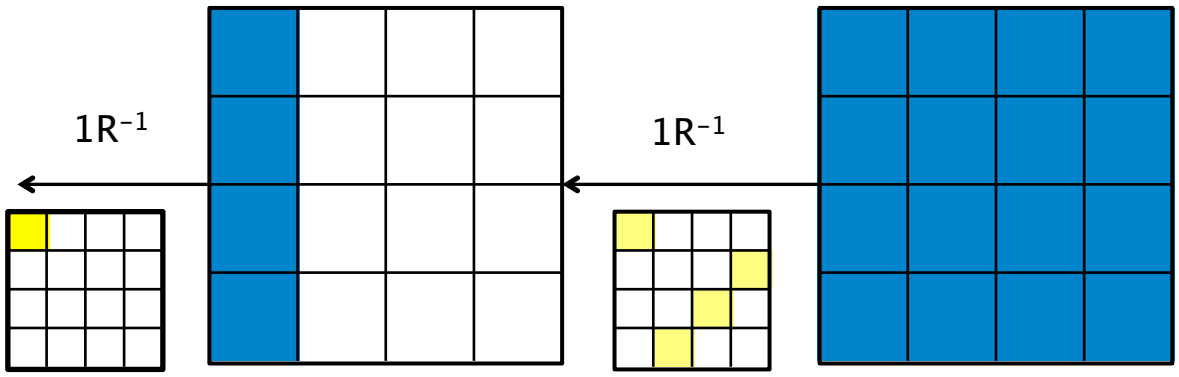
7-Round Attack



- Identifier les paires vérifiant la transition (complète).

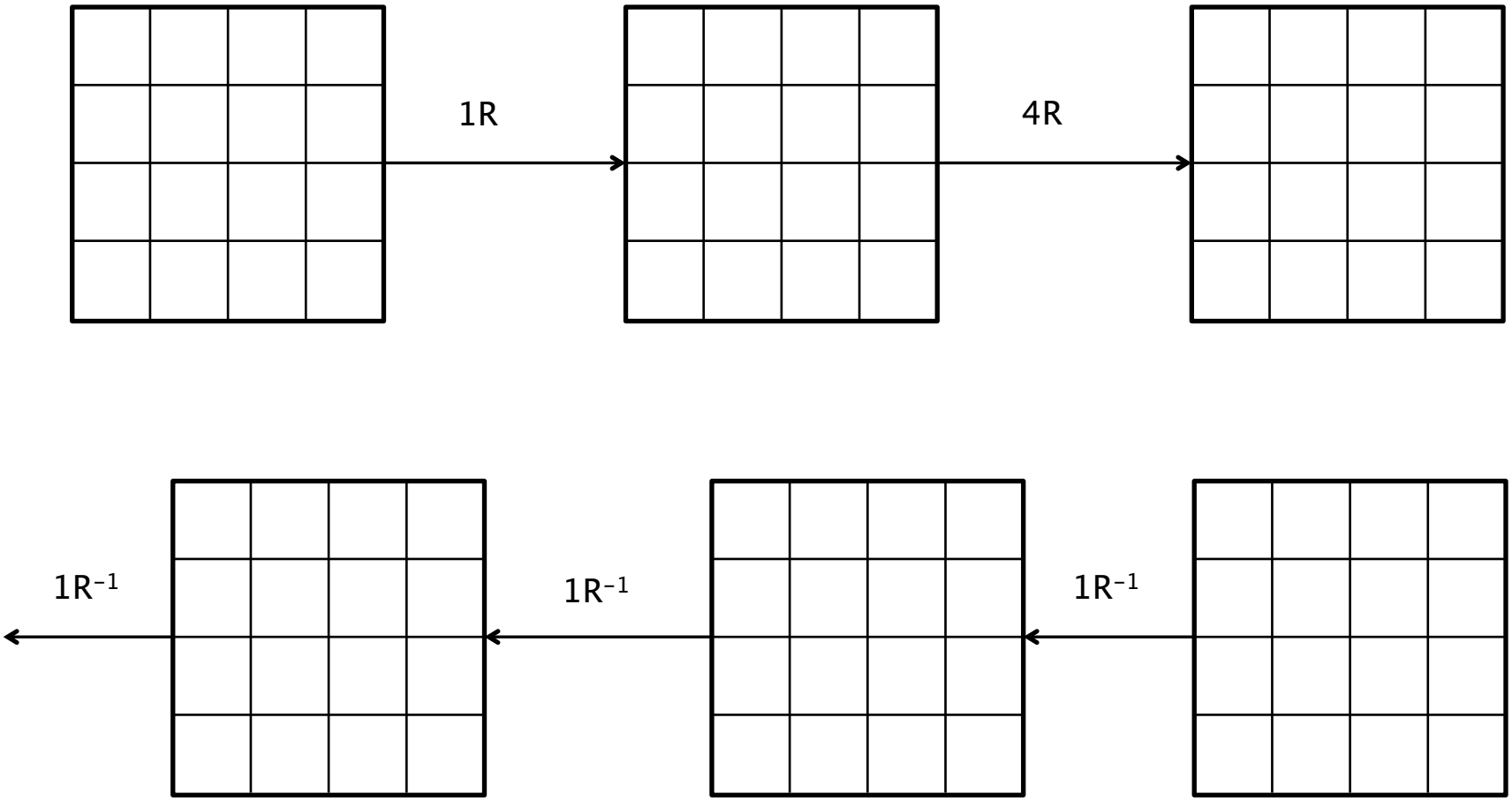
7-Round Attack



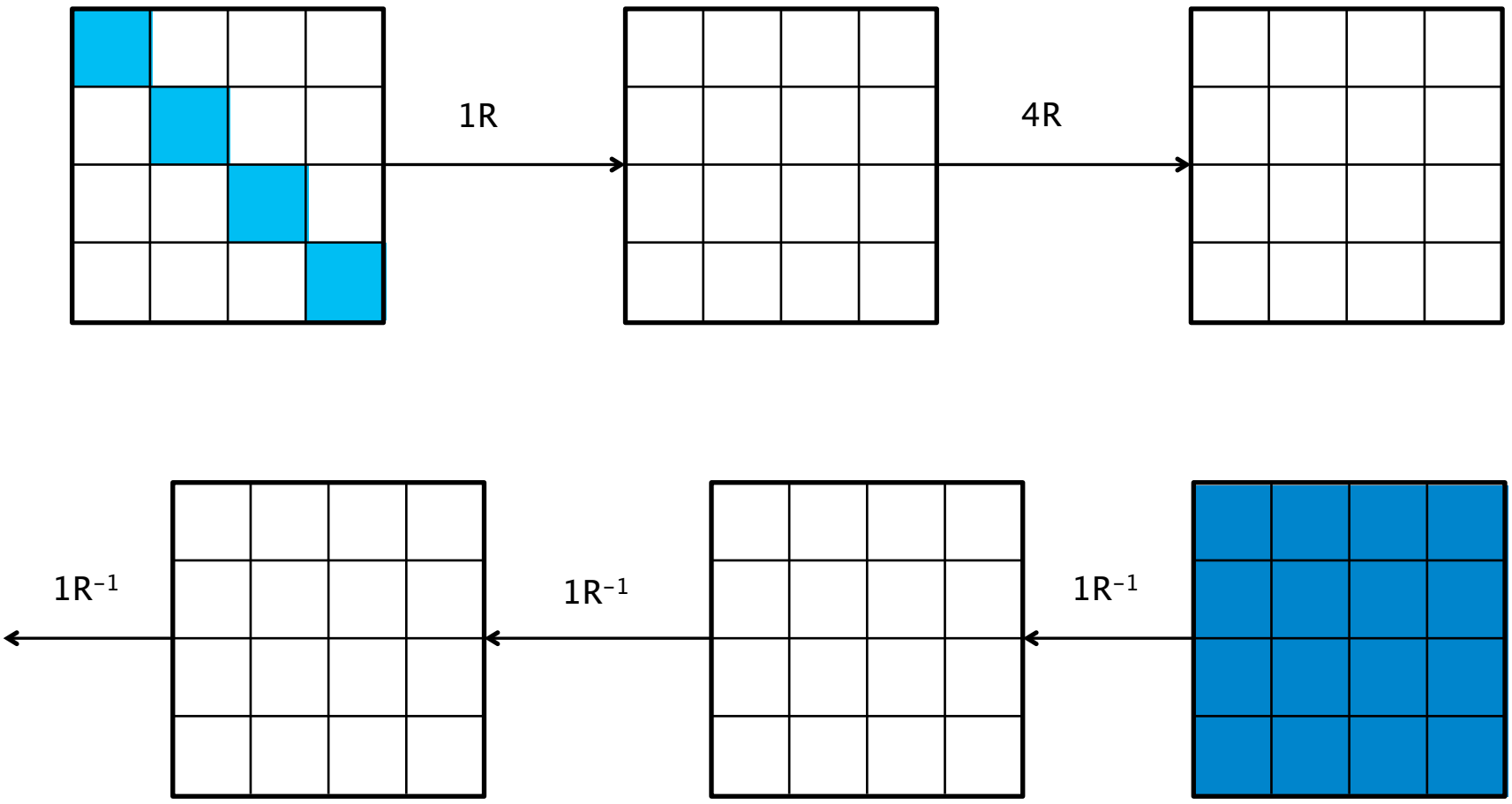


- Pour chaque bonne paire vérifier si la suite est valide

8-Round Attack

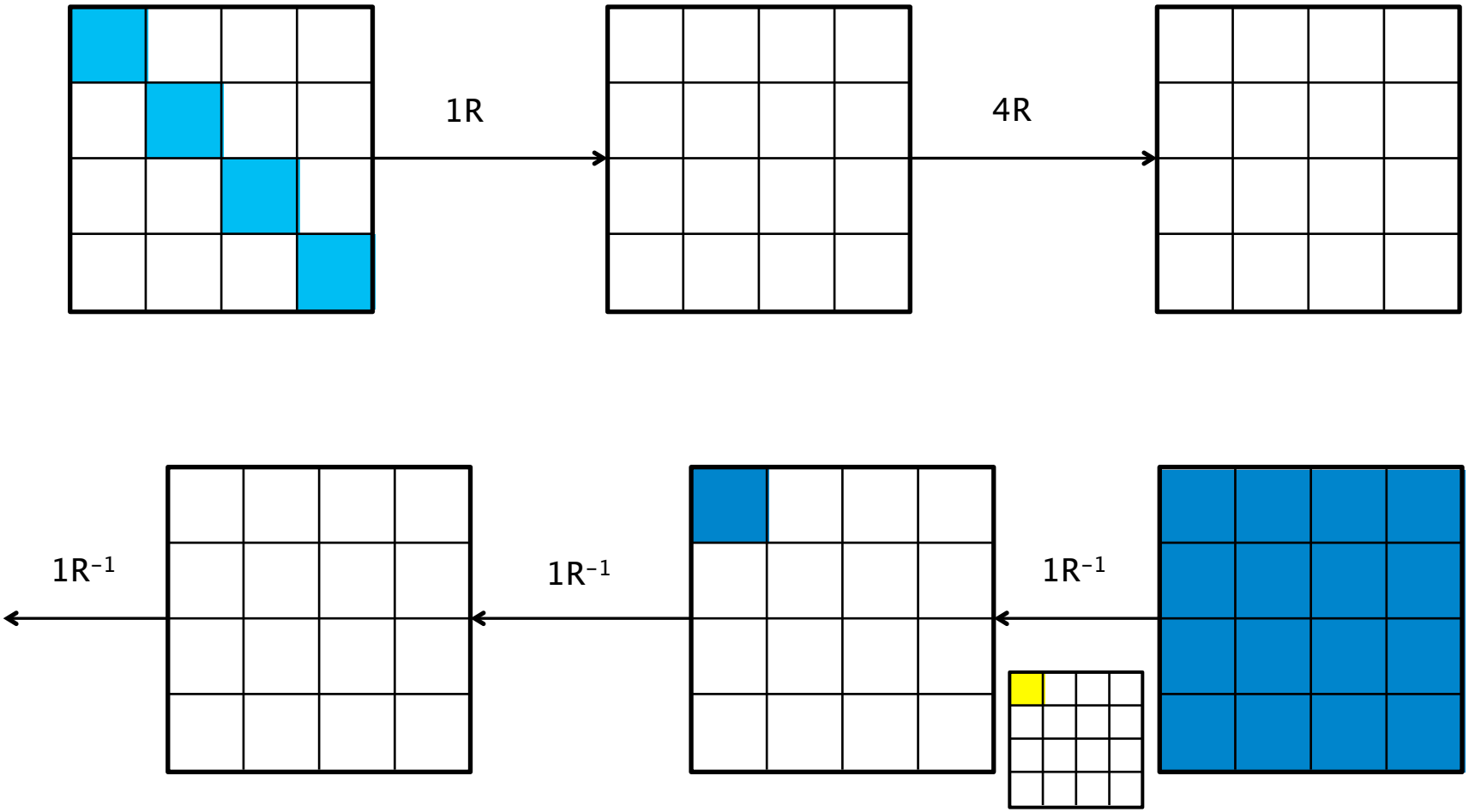


8-Round Attack



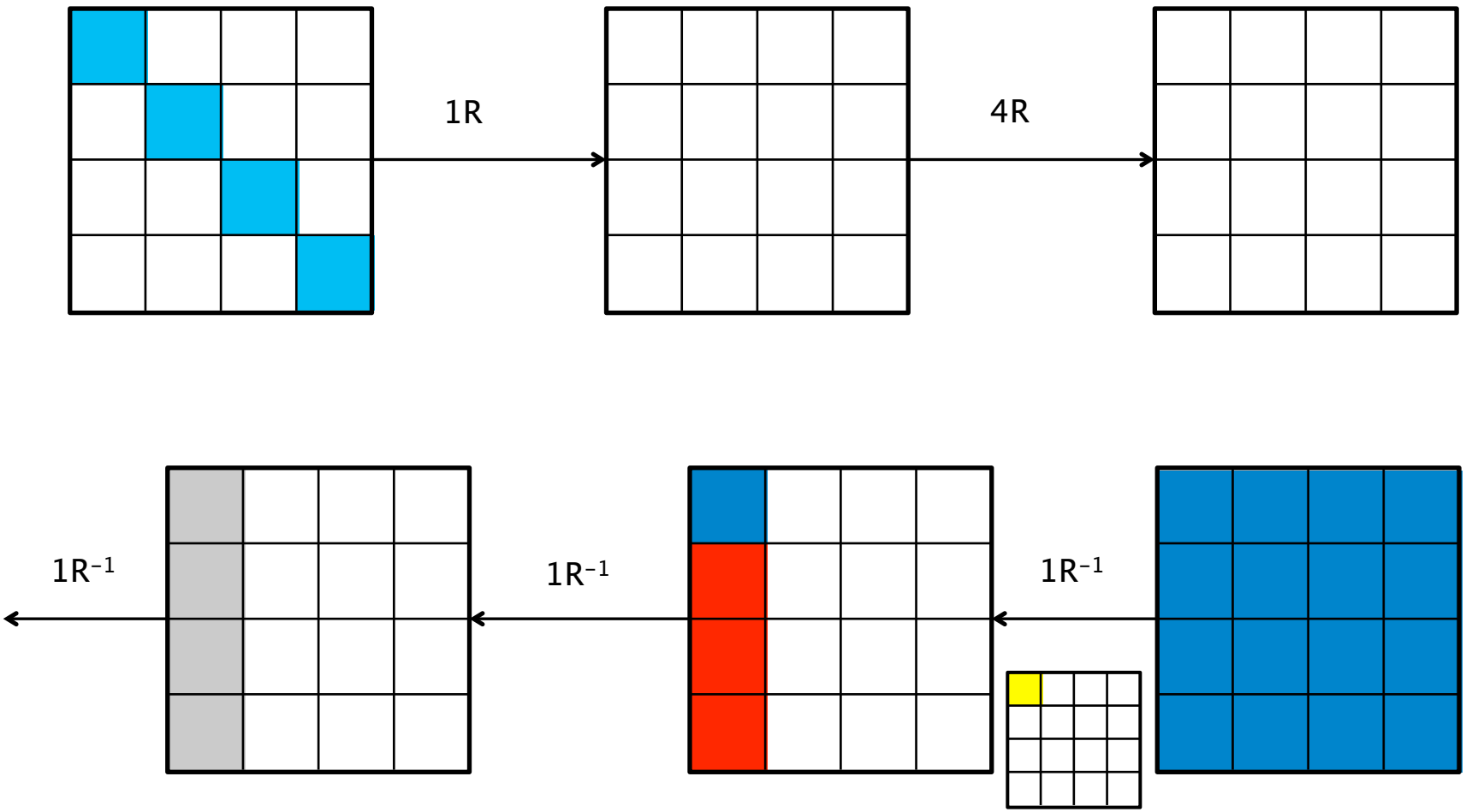
- Demander une structure de 2^{32} messages.

8-Round Attack



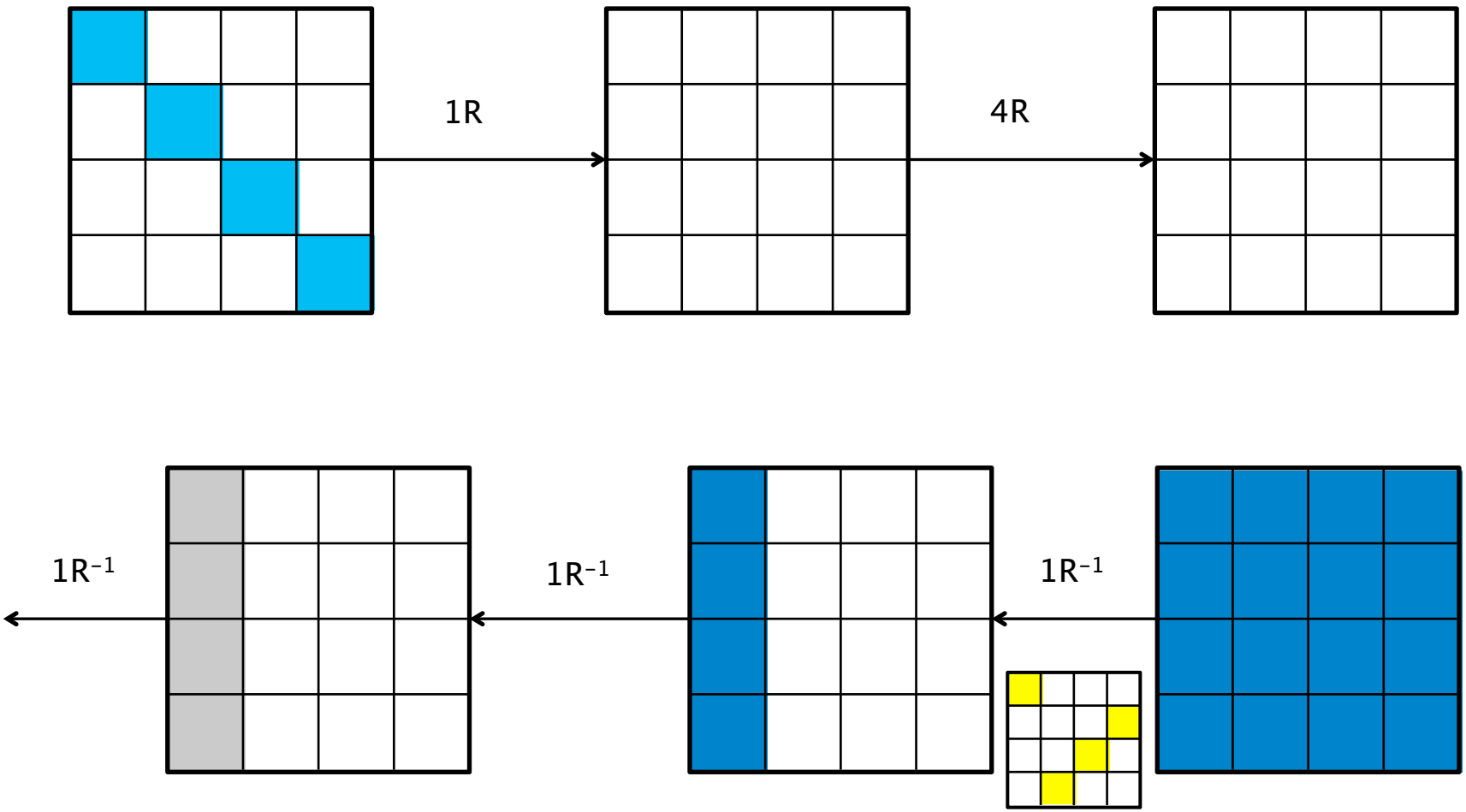
- Examiner chaque paire, et deviner un octet de U_8 (ou K_8)

8-Round Attack



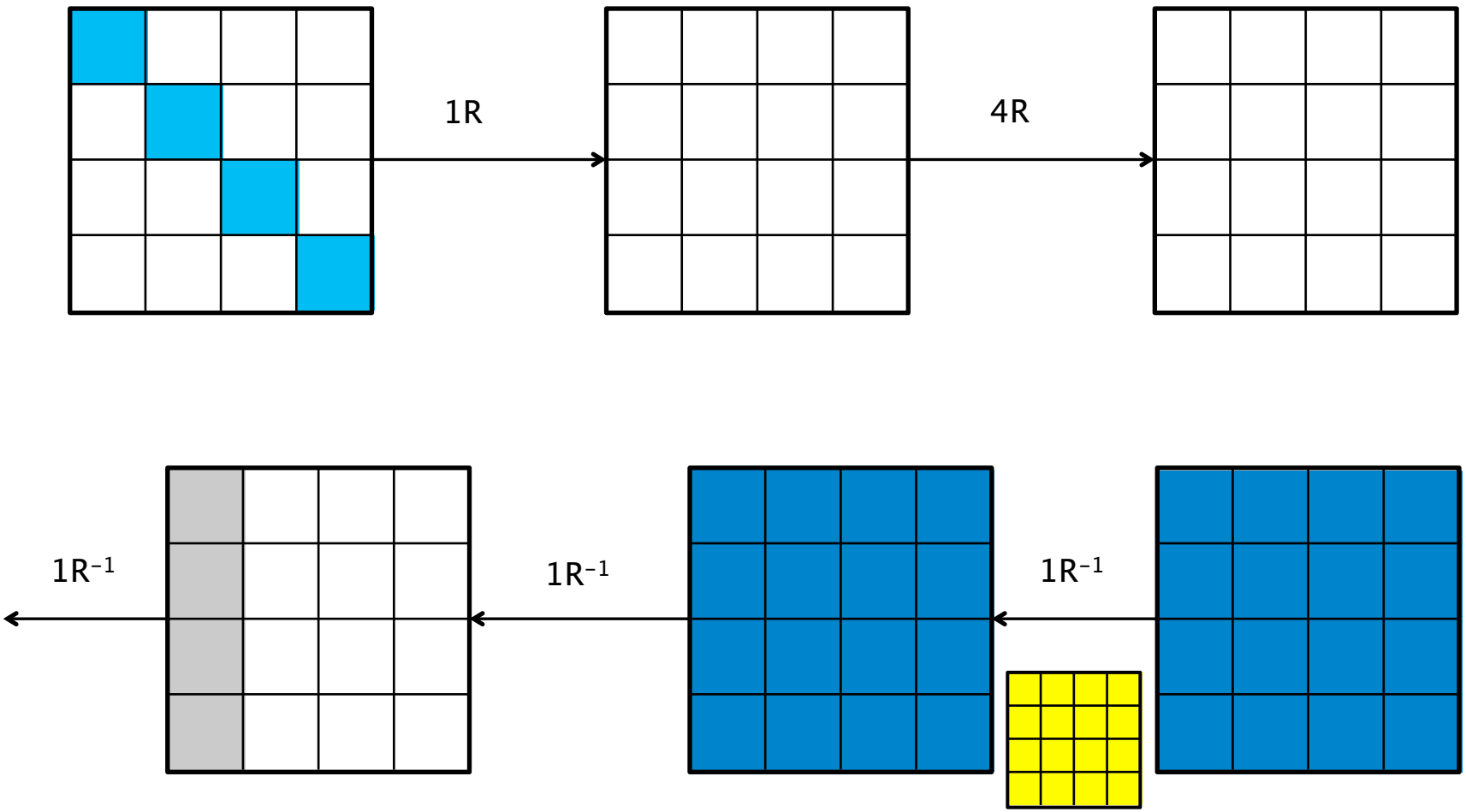
- Obtenir la différence sur la colonne.

8-Round Attack



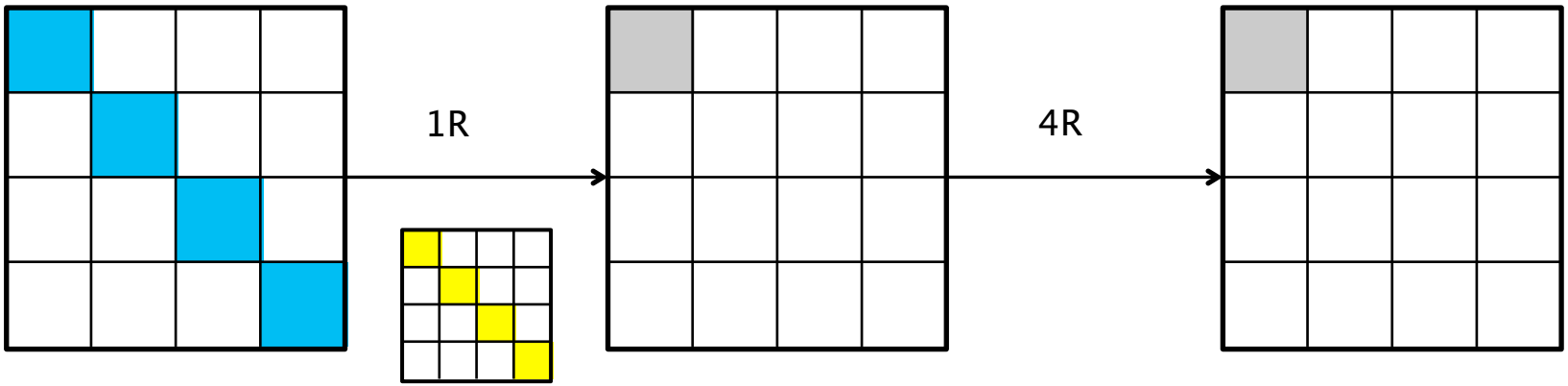
- Retrouver les octets de clé correspondant.

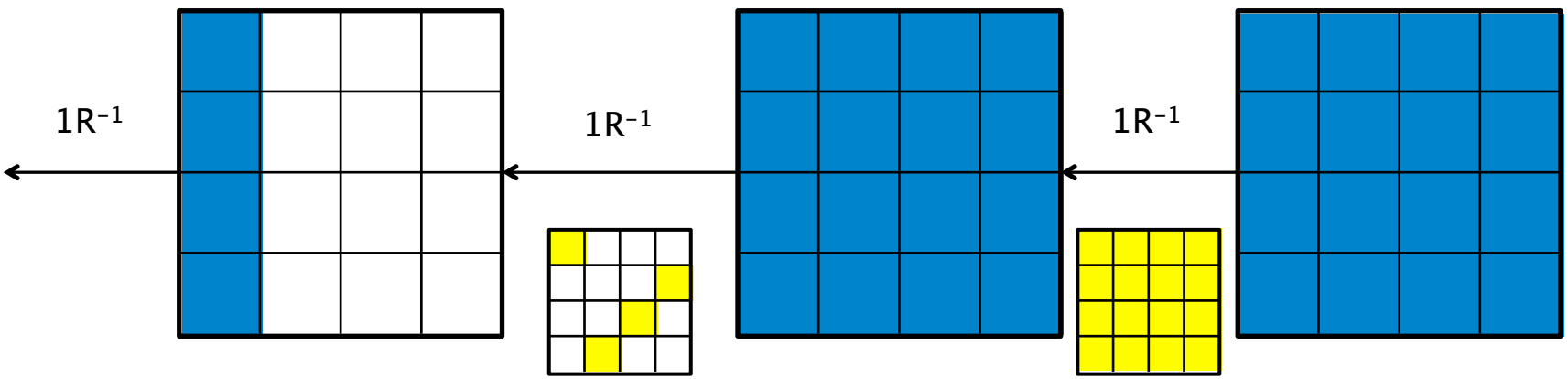
8-Round Attack



- Répéter l'opération sur les autres colonnes pour U_8.

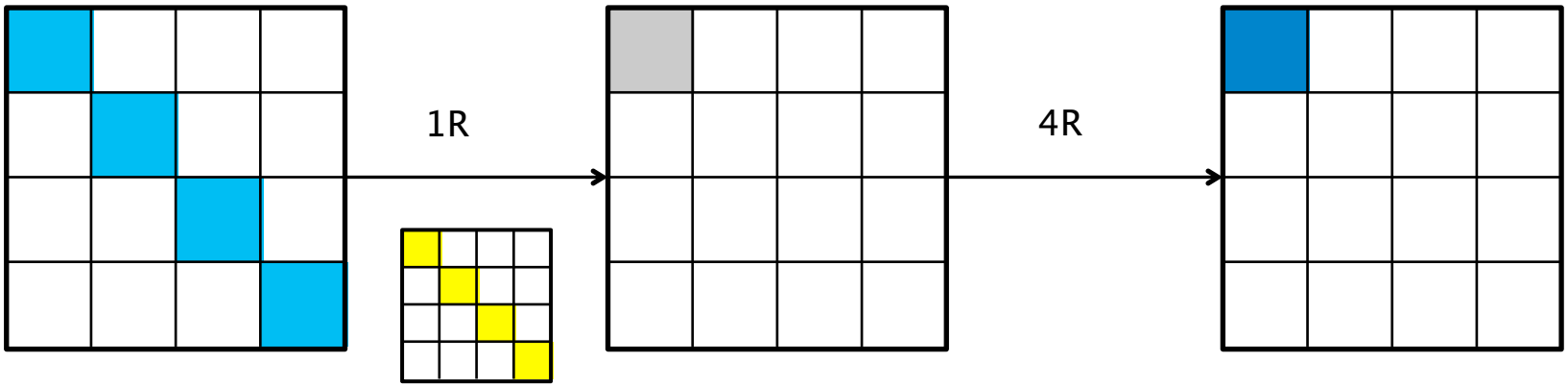
8-Round Attack

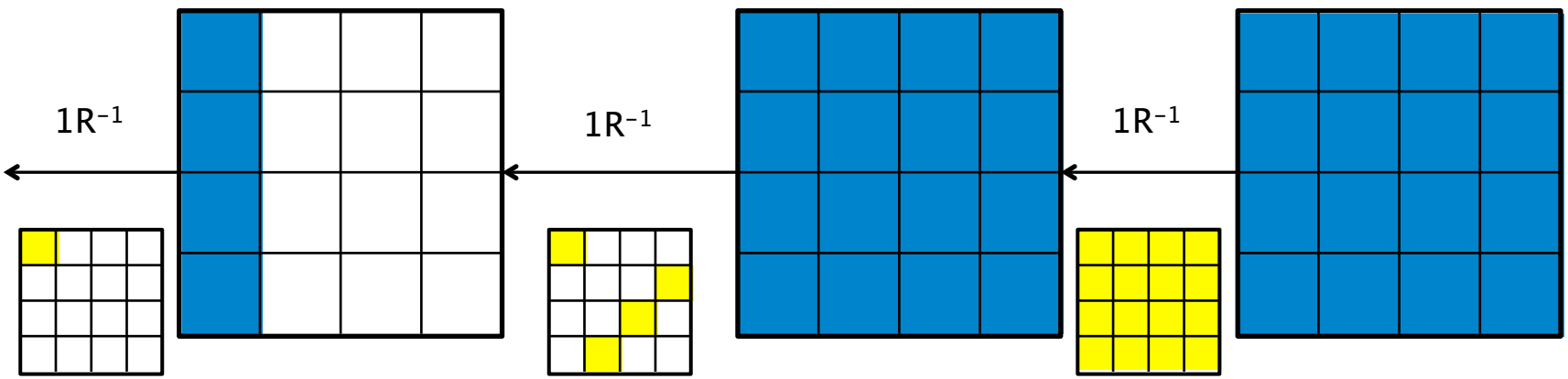




- De manière similaire, obtenir 4 octets de K_0 et 4 de U_7.

8-Round Attack





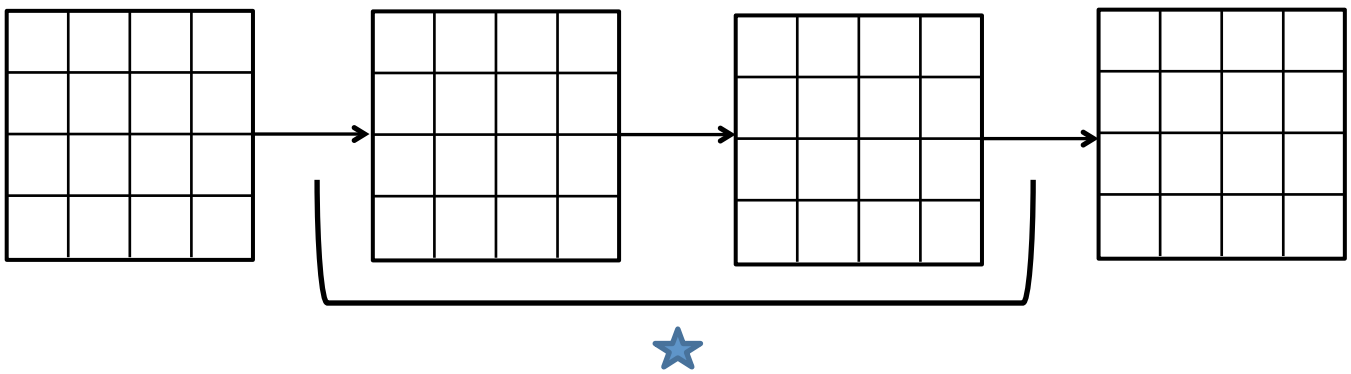
- Construire la suite et vérifier si elle est valide.

Résultats:

Version	Rounds	Data	Memory (Pre.)	Time	MinMax
All	7	2^{103+n}	2^{129-n}	2^{103+n}	2^{116}
AES-192	8	2^{113+n}	2^{129-n}	2^{172+n}	2^{172}
AES-256	8	2^{113+n}	2^{129-n}	2^{196+n}	2^{196}

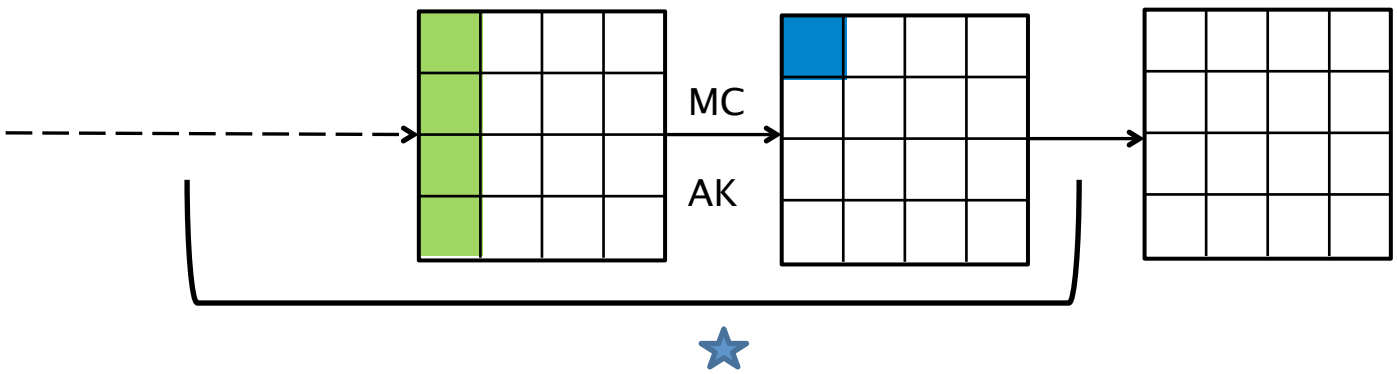
- Découverte d'une nouvelle relation entre les octets de sous-clés séparés par plusieurs tours
- Meilleure attaque connue sur 8 tours d'AES-256 (dans le modèle standard)
- Complexité en données élevée

Squelette des attaques



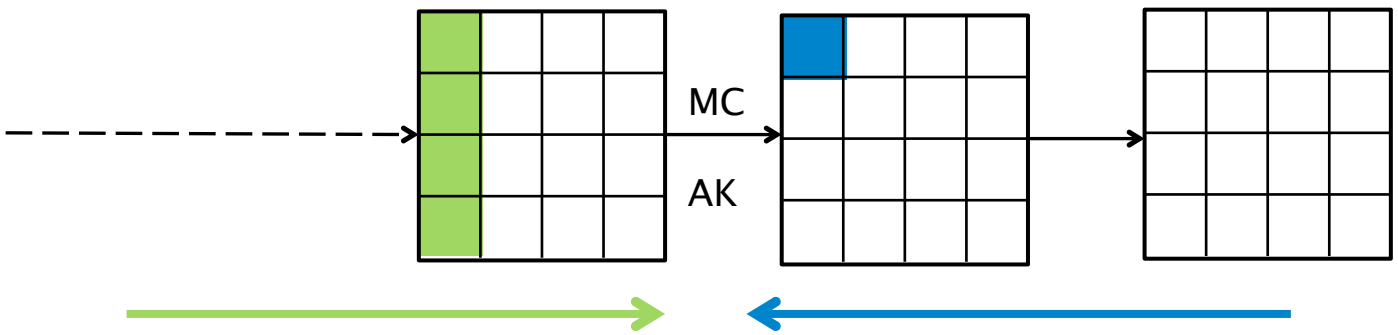
1. Construire la table des séquences possibles quand l'un des messages vérifie une certaine propriété ★
2. Identifier un message vérifiant la propriété
3. Identifier une structure de 256 messages contenant le message
4. Déchiffrer partiellement la structure et vérifier si la suite est dans la table

Squelette des attaques



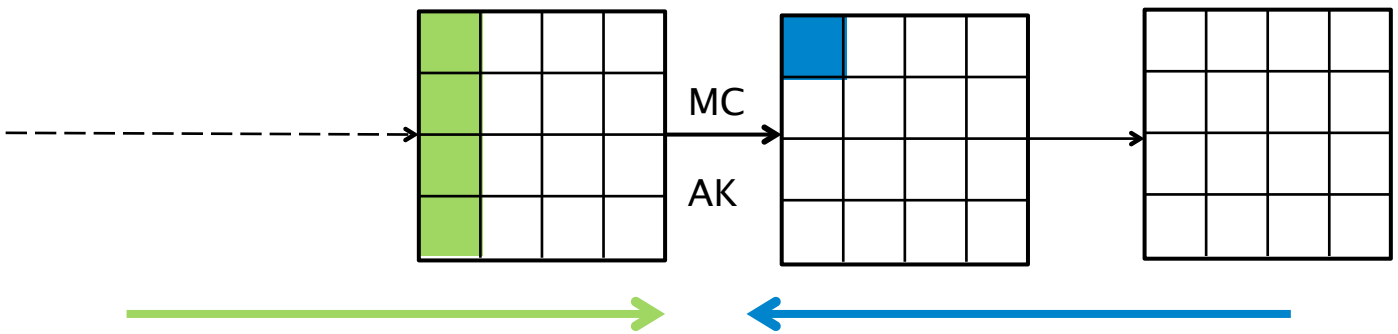
1. Construire la table des suites possibles quand l'un des messages vérifie une certaine propriété ★
2. Identifier un message vérifiant la propriété
3. Identifier une structure de 256 messages contenant le message
4. Déchiffrer partiellement la structure et vérifier si la suite est dans la table

Squelette des attaques



- Equation linéaire entre les différences des octets verts et bleus

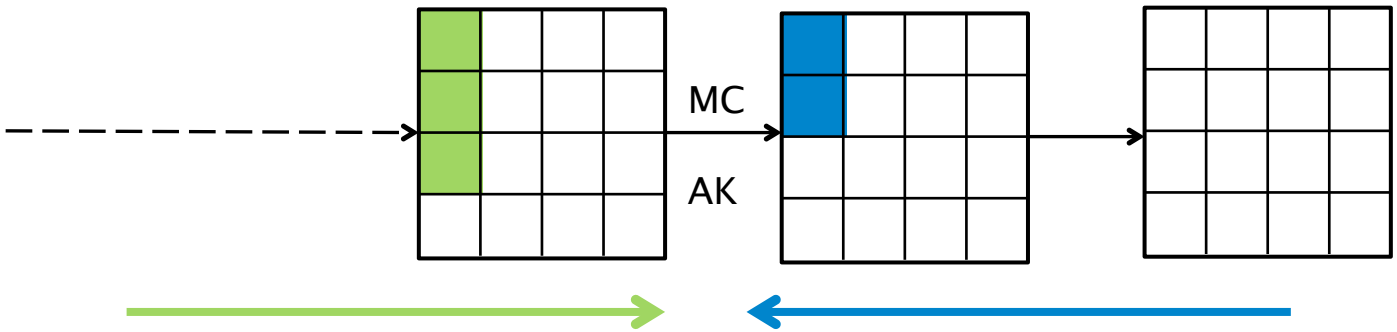
Squelette des attaques



- Equation linéaire entre les différences des octets verts et bleus
- Utiliser la propriété du MixColumn :

⟹ Nouveaux Trade-Off possibles!

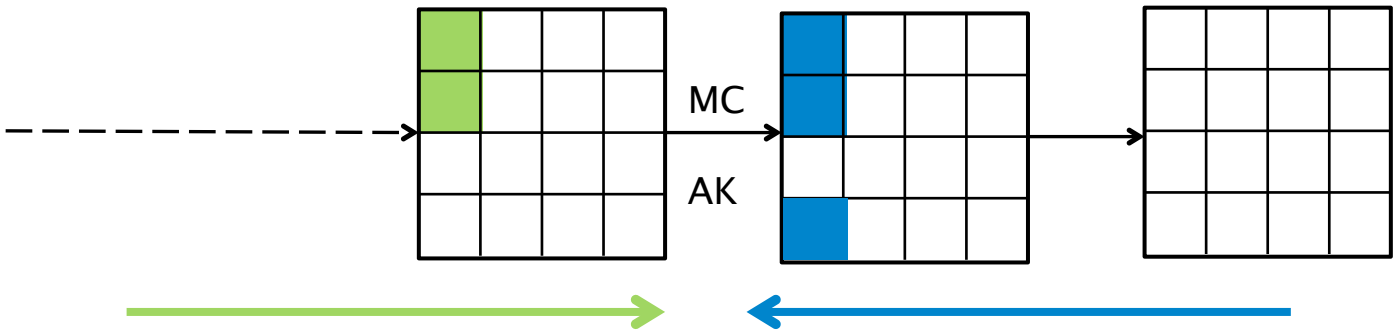
Squelette des attaques



- Equation linéaire entre les différences des octets verts et bleus
- Utiliser la propriété du MixColumn :

⇒ Nouveaux Trade-Off possibles!

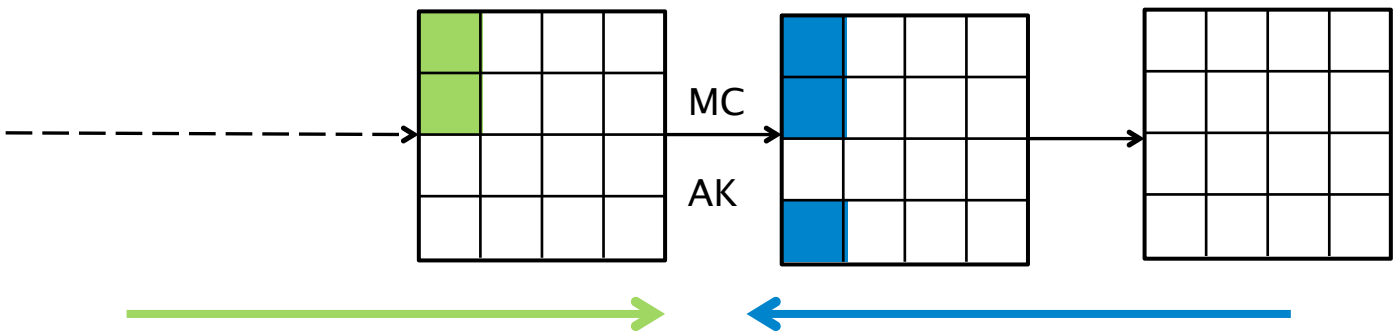
Squelette des attaques



- Equation linéaire entre les différences des octets verts et bleus.
- Utiliser la propriété du MixColumn :

⇒ Nouveaux Trade-Off possibles!

Squelette des attaques



- Equation linéaire entre les différences des octets verts et bleus
- Utiliser la propriété du MixColumn :

  ⟹ Nouveaux Trade-Off possibles!

- Plus de cas possibles → chance d'améliorer les attaques précédentes

Automatisation

- Beaucoup de cas à tester → automatisons la recherche !

1. Construire la table des suites possibles quand l'un des messages vérifie une certaine propriété 
2. Identifier un message vérifiant la propriété
3. Identifier une structure de 256 messages contenant le message
4. Déchiffrer partiellement la structure et vérifier si la suite est dans la table

Automatisation

- Beaucoup de cas à tester → automatisons la recherche !

1. Construire la table des séquences possibles quand l'un des messages vérifie une certaine propriété ★.

2. Identifier un message vérifiant la propriété.
3. Identifier une structure de 256 messages contenant le message.
4. Déchiffrer partiellement la structure et vérifier si la séquence est dans la table.

Automatisation

- Beaucoup de cas à tester → automatisons la recherche !

1. Construire la table des séquences possibles quand l'un des messages vérifie une certaine propriété ★.

2. Identifier un message vérifiant la propriété.
3. Identifier une structure de 256 messages contenant le message.
4. Déchiffrer partiellement la structure et vérifier si la séquence est dans la table.

- Chaque étape nécessite de guesser des octets des états intermédiaires et des sous-clés.
- Problème : Énumérer ces octets sous la contrainte de la propriété ★ et/ou du keyschedule.

Automatisation

- Problème résolu en utilisant l'outil développé par Bouillaguet, Derbez et Fouque.

Algorithme : Outil

Entrée : un système d'équations E en les variables X.

Sortie : le meilleur algorithme pour énumérer les solutions de E.

Algorithme : Fun

Entrée : un système d'équations E en les variables X et un ensemble $Y \subseteq X$.

Sortie : un ensemble de variable $Y \subseteq Z \subseteq X$ et le meilleur algorithme pour énumérer les solutions du sous-système de E en les variables Z.

Heureusement, on dispose d'un meilleur algorithme que la recherche exhaustive !

Résultats (DS)

Version	Rounds	Data	Memory (Pre.)	Time	Ref.
AES-192	7	2^{95}	2^{143}	2^{143}	DS
	7	2^{116}	2^{116}	2^{116}	DKS
	7	2^{32}	2^{130}	2^{130}	New
	8	2^{113}	2^{129}	2^{172}	DKS
	8	2^{32}	2^{182}	2^{182}	New

Version	Rounds	Data	Memory (Pre.)	Time	Ref.
AES-256	7	2^{95}	2^{143}	2^{143}	DS
	7	2^{116}	2^{116}	2^{116}	DKS
	7	2^{32}	2^{134}	2^{134}	New
	8	2^{95}	2^{143}	2^{143}	DS
	8	2^{113}	2^{129}	2^{196}	DKS
	8	2^{32}	2^{193}	2^{195}	New

Résultats

- Des attaques avec encore moins de données :

Version	Rounds	Data	Memory (Pre.)	Time
AES-128	6	2^{8}	2^{106}	2^{106}
AES-192	7	2^{8}	2^{153}	2^{163}
AES-256	7	2^{16}	2^{178}	2^{178}
AES-256	8	2^{8}	2^{234}	2^{234}

- Des attaques par « bicliques » :

Version	Rounds	Data	Memory (Pre.)	Time
AES-128	7	2^{32}	$2^{126,5}$	$2^{126,5}$
AES-256	9	2^{32}	$2^{254,2}$	$2^{254,2}$

Résultats (DKS)

- Les attaques trouvées par Dunkelman, Keller et Shamir sont optimales mais ...
- ... la taille de la table a été largement surestimée :

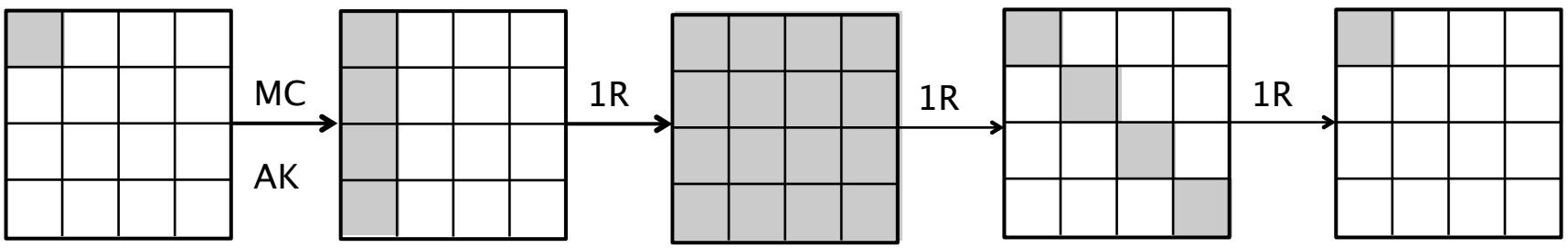
$$2^{129} \rightarrow 2^{80} \ !$$

- Résultats :

Version	Rounds	Data	Memory (Pre.)	Time	Ref.
All	7	2^{116}	2^{116}	2^{116}	DKS
All	7	2^{100}	2^{100}	2^{100}	New
AES-192	8	2^{113}	2^{129}	2^{172}	DKS
AES-192	8	2^{113}	2^{80}	2^{172}	New
AES-256	8	2^{113}	2^{129}	2^{196}	DKS
AES-256	8	2^{113}	2^{80}	2^{196}	New
AES-256	9	2^{119}	2^{206}	2^{206}	New

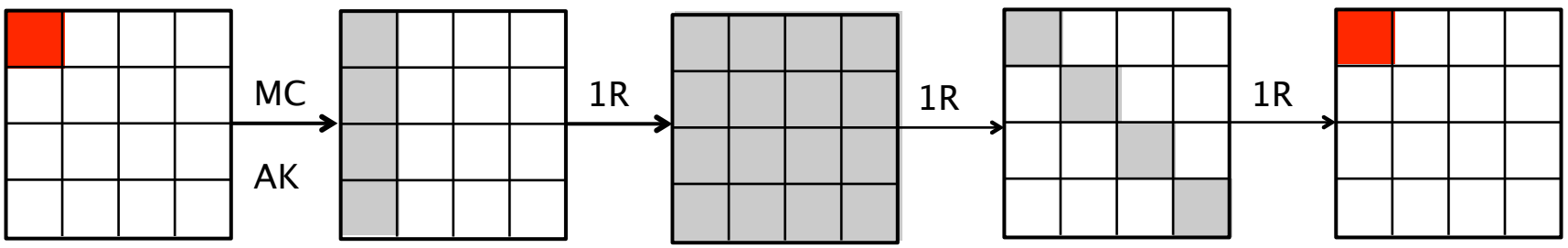
Enumération Différentielle

Considérons une paire de messages (P,P') vérifiant:



Enumération Différentielle

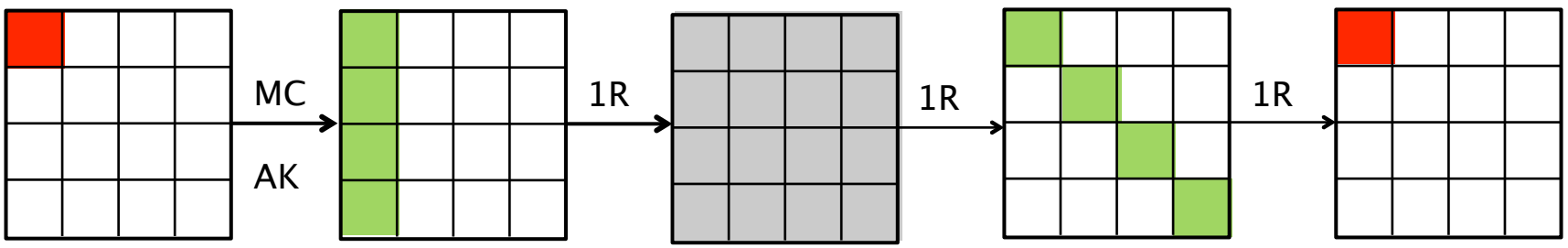
Considérons une paire de messages (P,P') vérifiant:



1. Guesser les différences dans les octets rouges.

Enumération Différentielle

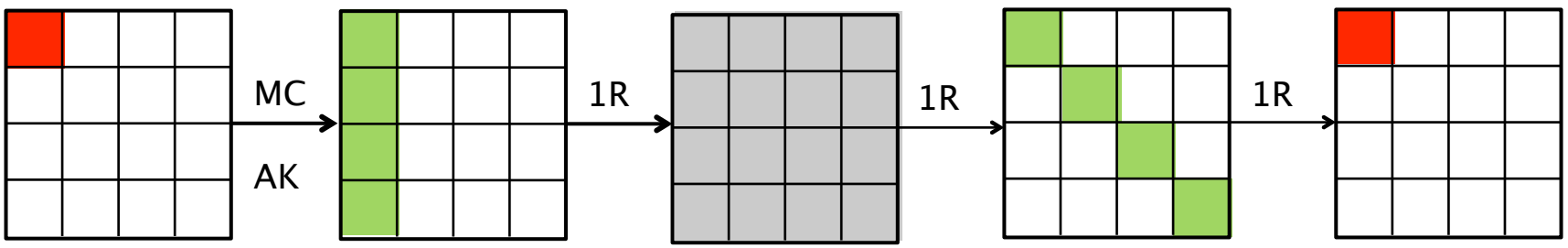
Considérons une paire de messages (P,P') vérifiant:



1. Guesser les différences dans les octets rouges.
2. Guesser les valeurs des octets verts de P.

Enumération Différentielle

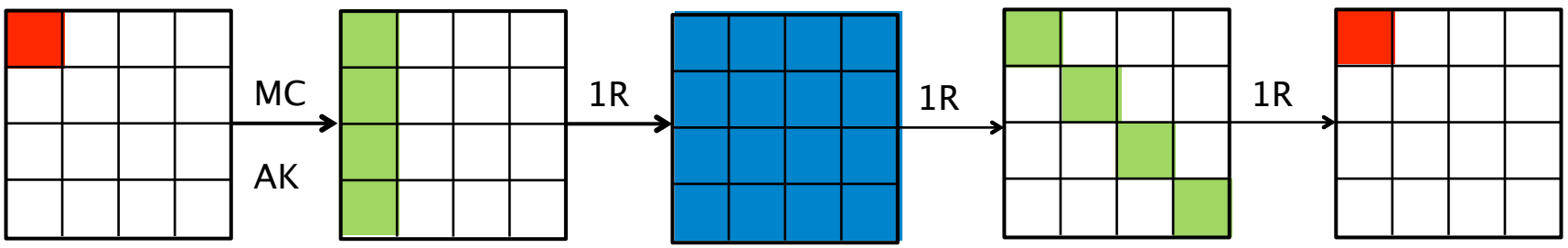
Considérons une paire de messages (P,P') vérifiant:



1. Guesser les différences dans les octets rouges.
2. Guesser la valeurs des octets verts de P.
3. En déduire la valeurs des octets verts de P'.

Enumération Différentielle

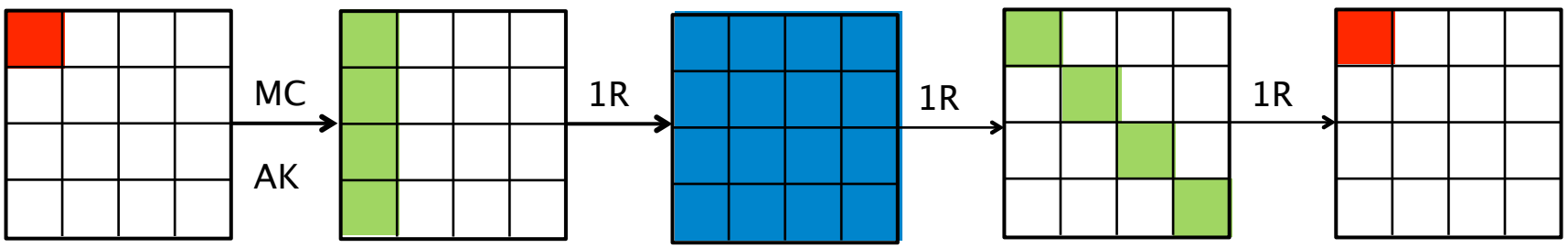
Considérons une paire de messages (P,P') vérifiant:



1. Guesser les différences dans les octets rouges.
2. Guesser la valeurs des octets verts de P.
3. En déduire la valeurs des octets verts de P'.
4. En déduire la valeurs des octets bleus de P et P'.

Enumération Différentielle

Considérons une paire de messages (P,P') vérifiant:



1. Guesser les différences dans les octets rouges.
2. Guesser la valeurs des octets verts de P.
3. En déduire la valeurs des octets verts de P'.
4. En déduire la valeurs des octets bleus de P et P'.

Les octets bleus et verts sont exactement les 24 paramètres !

Conclusion

- Baisse significative de la complexité en données des attaques de Demirci et Selçuk
- Baisse importante de la complexité en espace des attaques de Dunkelman, Keller et Shamir
- Meilleure attaque connue sur 7 tours d'AES
- Attaques trouvées automatiquement

- Problème ouvert : Trouver de nouvelles propriétés améliorant les attaques existantes