



# T28: ワイヤレス・セキュリティ

---

WEP, 802.1X, WPA, そして  
802.11i へ

Internet Week 2003, Yokohama

進藤 資訓

株式会社データコントロール

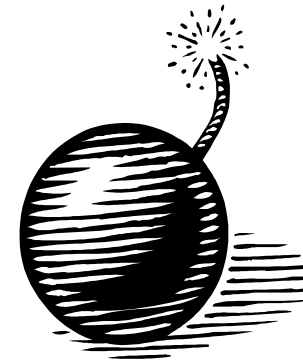
CTO

[shindo@datacontrol.co.jp](mailto:shindo@datacontrol.co.jp)

# 開口一番

---

- ワイヤレスは危ないか？
  - 危ない！
    - もし、正しく使わなければ・・・。
- なぜ危ないか？
  - よく分からない！？
- どれくらい危ないか？
  - よく分からない！？
- どうすれば防げるか？



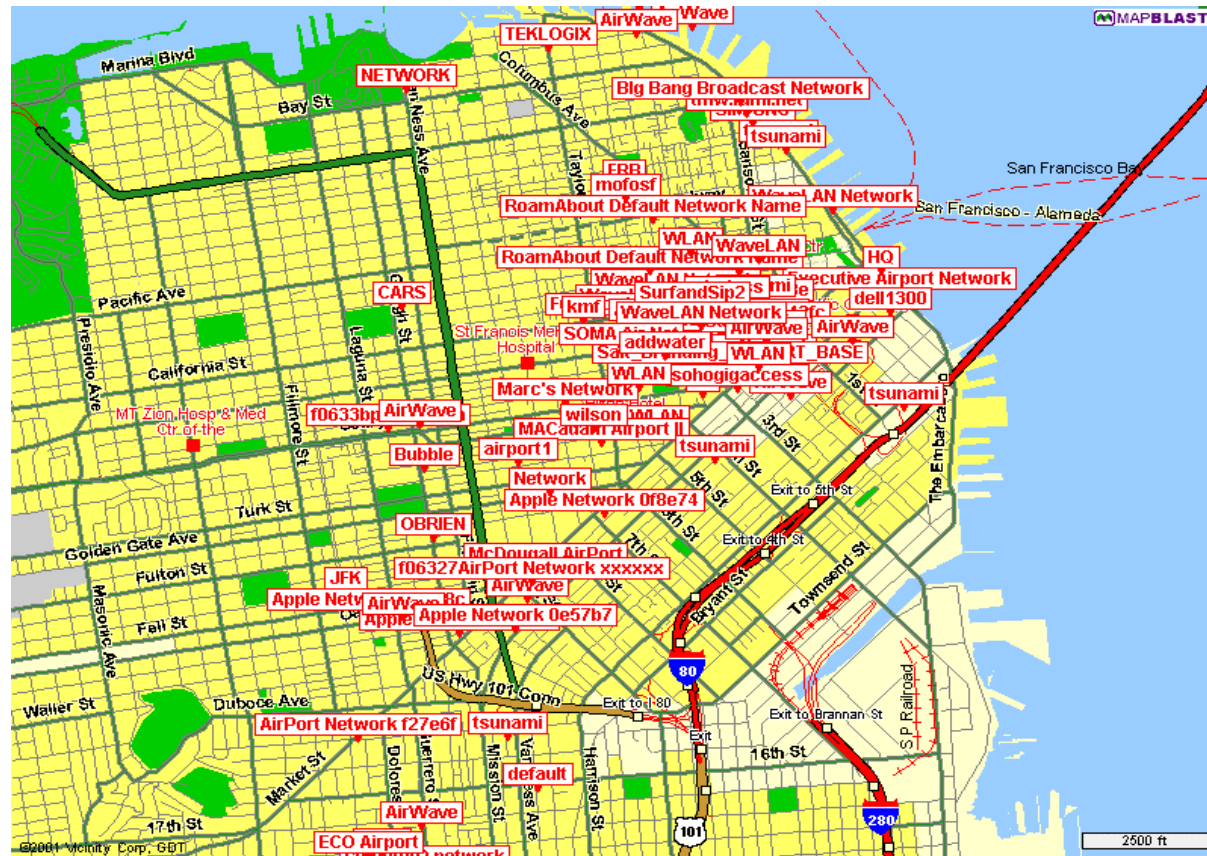
# セキュリティ

---

- 認証 (Authentication)
- 許可 (Authorization)
- 秘匿性 (Confidentiality)
- 完全性 (Integrity)
- 否認防止 (Non-repudiation)



# 攻撃(1) ~ War Driving ~





## 対策(1-1) ～ ESS-ID の隠蔽 ～

---

- 呼び名は色々
  - Closed System or Network
  - ステルス機能
  - ...
- 実装も色々
  - 802.11 のビーコンを止める
  - プロブリクエストに対して、
    - 応答しない
    - 応答はするが、レスポンスに SSID は入れない
    - 自分の SSID に合致する場合のみ応答

## 対策(1-2) ～ MAC 認証 ～

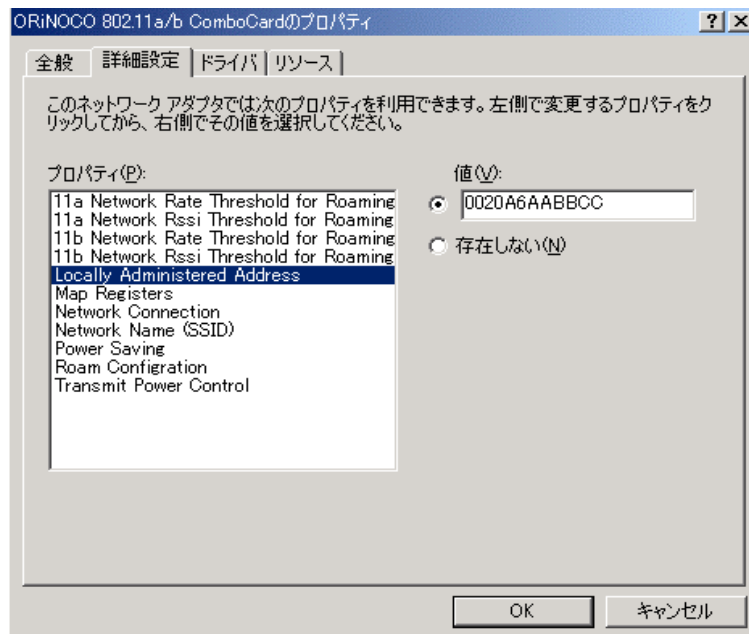
---

- 接続を許可する MAC アドレス(のリスト)を設定
  - AP に静的に設定する
  - RADIUS 等のサーバーに設定する



## 攻撃(2) ～ 詐称(なりすまし) ～

- MAC アドレスの詐称は簡単！
- 正規のMAC アドレスはワイヤレス上で簡単に見つけることができる！



```
# ifconfig eth1 down  
# ifconfig eth1 hw ether 12:34:56:aa:bb:cc  
# ifconfig eth1 up
```



## 対策(2) WEP

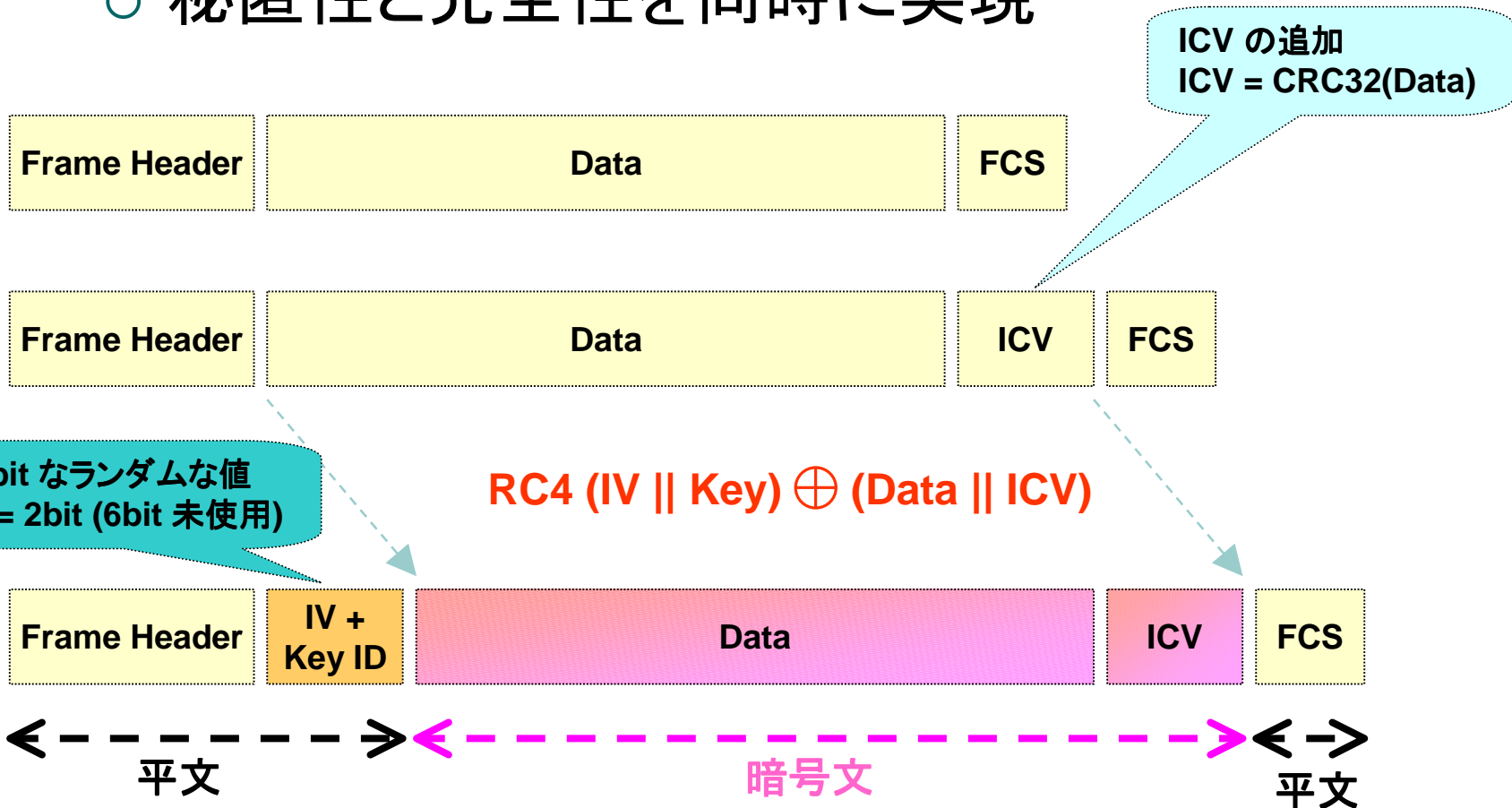
---

- WEP (Wired Equivalent Privacy)
  - 秘匿性 (Confidentiality)
  - 完全性 (Integrity)
  - 認証 (Authentication)
- What on Earth does this Protect?

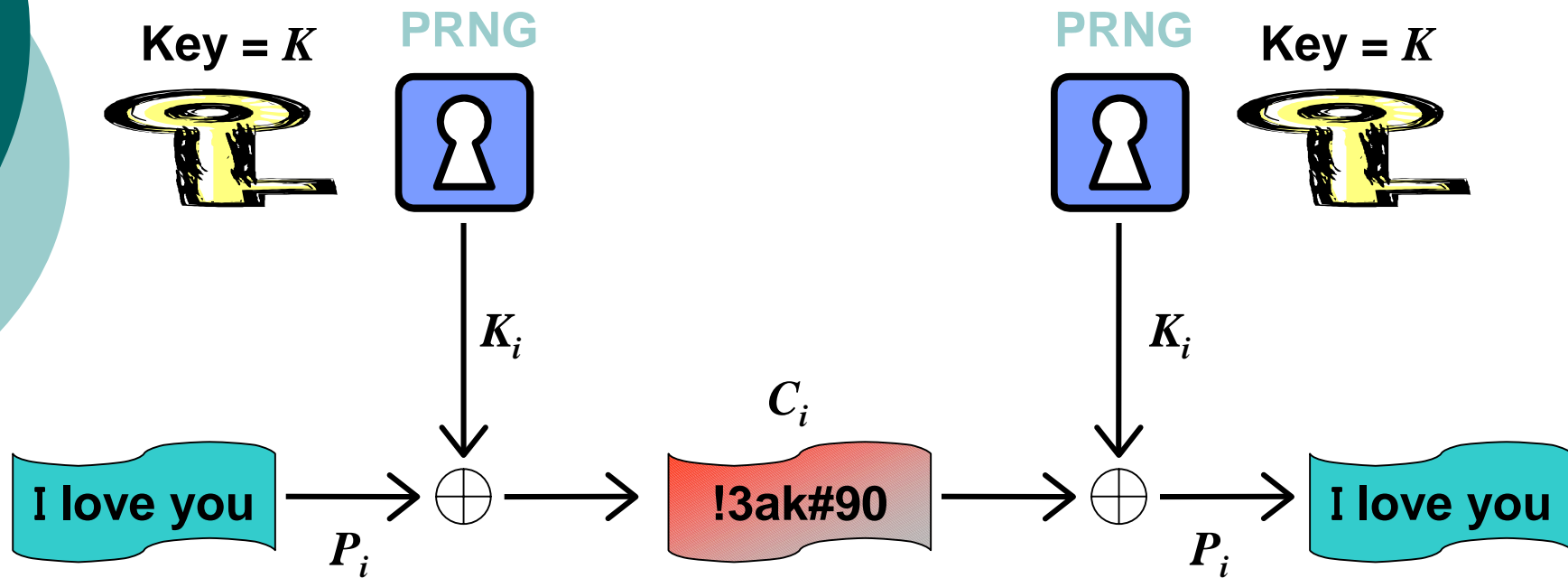


# WEP 処理

- 秘匿性と完全性を同時に実現



# Stream Cipher



**Property 1:** If  $C_i = P_i \oplus K_i$  Then  $P_i \oplus C_i = K_i$

**Property 2:** If  $C_1 = P_1 \oplus K_a$  and  $C_2 = P_2 \oplus K_a$   
Then  $C_1 \oplus C_2 = (P_1 \oplus K_a) \oplus (P_2 \oplus K_a) = P_1 \oplus P_2$



# WEPの問題点

---

- 鍵長が 40bit と短い
  - Brute Force で破れる。
  - 最近ではほとんどの場合長い鍵 (e.g. 104 or 128 bits) が利用可能。
- ICV に CRC32 を用いている
  - ICVは暗号化対象ではあるが、CRC自体は暗号的強度はない。
  - 鍵と組み合わせられていない。
- 一つの鍵を使い続ける
  - どんなに強力な暗号アルゴリズムでも1つの鍵を長く使うのは望ましくない。



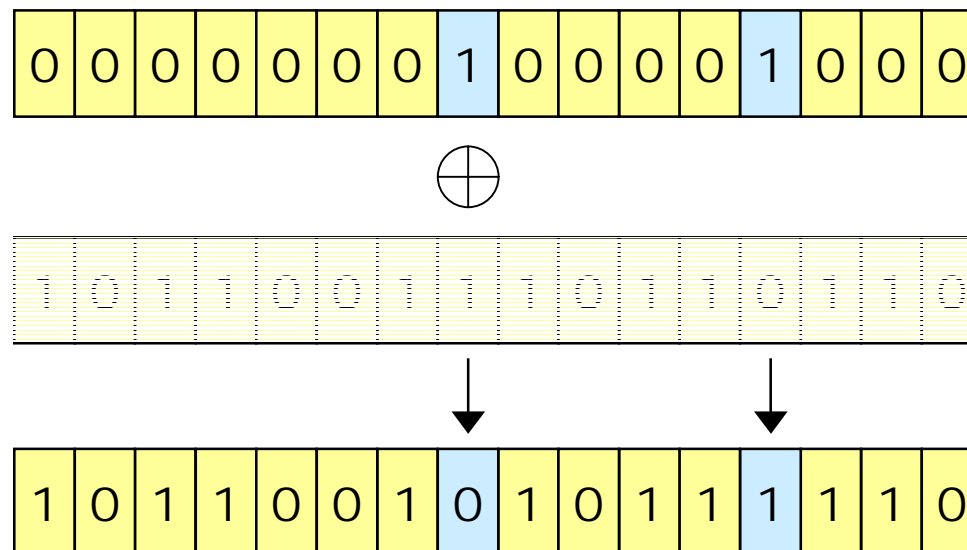
## WEPの問題点(cont'd)

---

- 鍵の配布メカニズムがない
  - スケールしない。
- IV の空間が小さい(i.e. 24bit)
  - 扱い方が規定されていない。
  - フレームごとに1増やす場合、200 bytes/packet, 10% utilized で 14 時間で再利用される。
- リプレイ攻撃に無力
- FMS 攻撃

# Bit Flipping Attack

- CRC は XOR に対して線形である！
  - $\text{CRC}(M \text{ XOR } \Delta) = \text{CRC}(M) \text{ XOR } \text{CRC}(\Delta)$
- M 中の任意の bit を set したり、clear したりすることはできないが、bit を反転させることはできる！





# FMS 攻撃

---

- S. Fluhrer, I. Mantin, A. Shamir, *Aug. 2001*
- Key Recovery
- 条件
  - 生成される RC4 stream の最初のバイトが判っていて、
  - IV がある種の条件を満たす場合、Key Byte を5%の確率で guess できる
    - 代表的 Weak IV:  $(B+3, 0xff, M)$
- key の長さに比例しかしない！
- 4,000,000 ~ 6,000,000 パケットで 40bit WEP を解読できる
- 更なる最適化で 1,000,000 パケット程度で解読可能
  - 5Mbps, 200 bytes/packet で、3125 秒



# RC4 は脆弱か？

---

- 若干の脆弱性はあるが、一般的にはほとんど問題ない
- WEP が脆弱なのは RC4 の使い方を少々間違えたからである
- RC4 を正しく使えば安全
  - IV を MD5 や SHA1 でハッシュする
    - 例) SSL or TLS
  - 最初の数百バイト(例えば 256 バイト)を捨てる
    - 例) GTK over EAPOL



## 攻撃(3) ~ WEP Cracking ~

---

- AirSnort
  - <http://airsnort.shmoo.com>
- WEPCrack
  - <http://wepcrack.sourceforge.net>
- bsd-airtools
  - <http://dachb0den.com/projects/bsd-airtools.html>



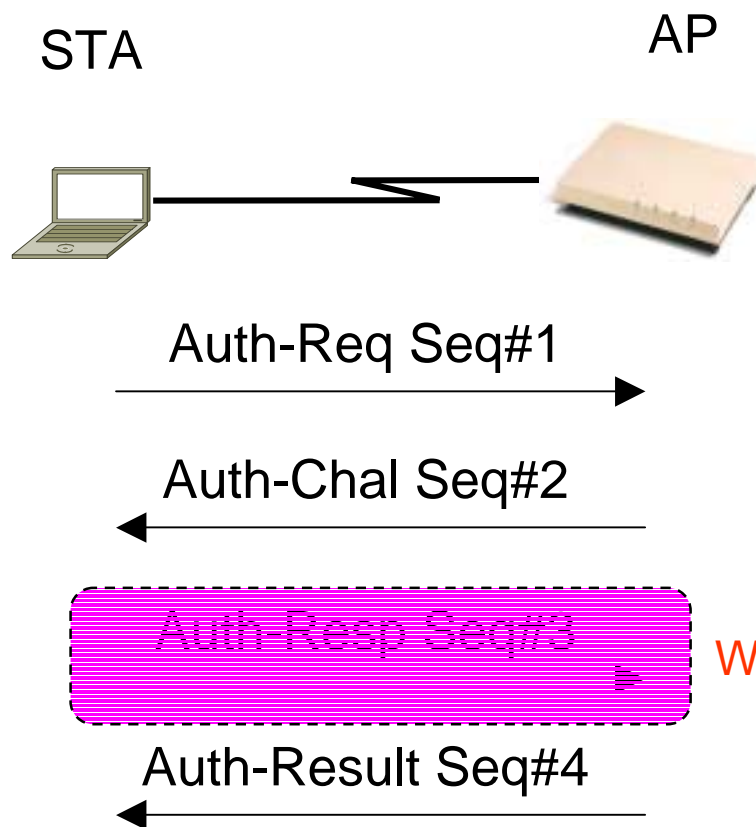


## 対策（3-1）WEP plus

---

- Agere 802.11b Firmware 8.10 or later
  - Weak IV を避ける
  - 最初の IV をランダムに決める
- メリット
  - FMS 攻撃は避けられる
- デメリット
  - IV の空間をさらに小さくする
  - チップセット依存

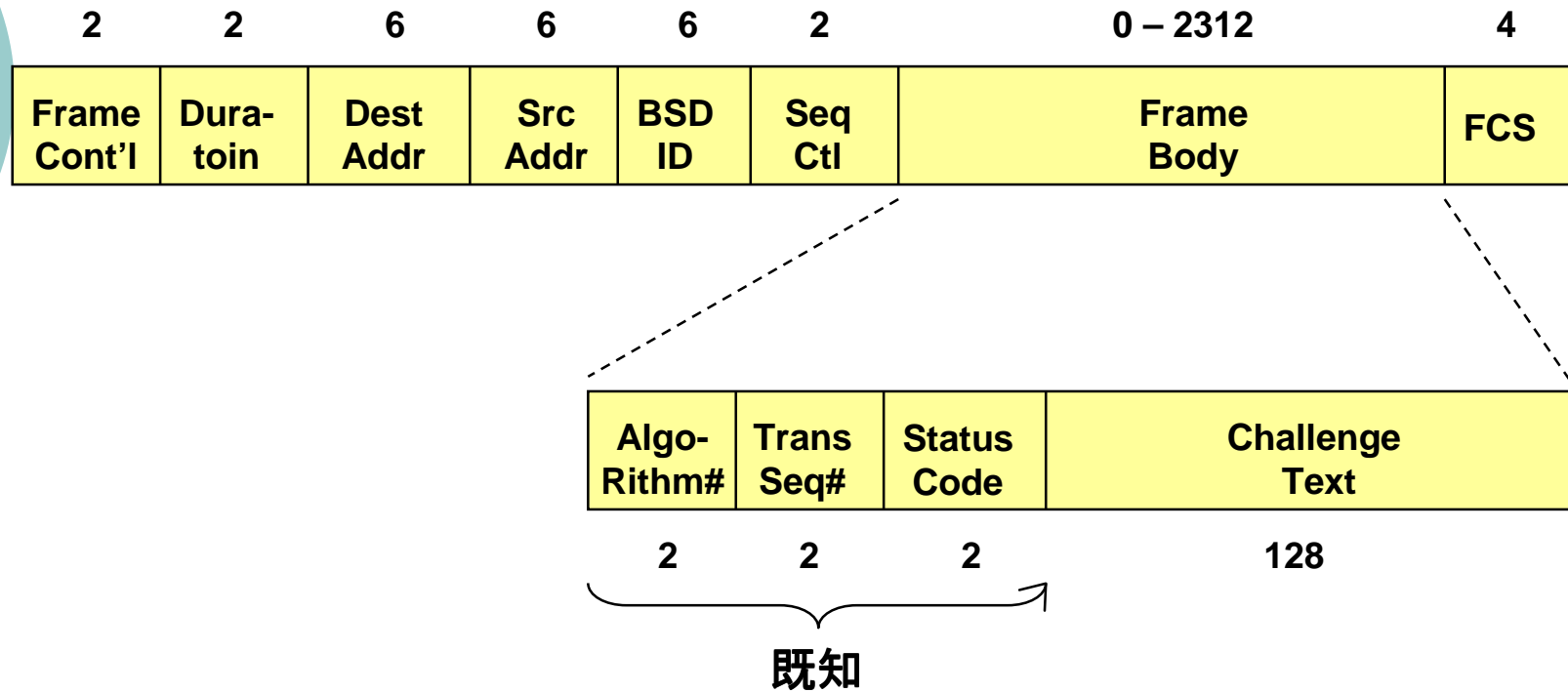
# 802.11 の認証



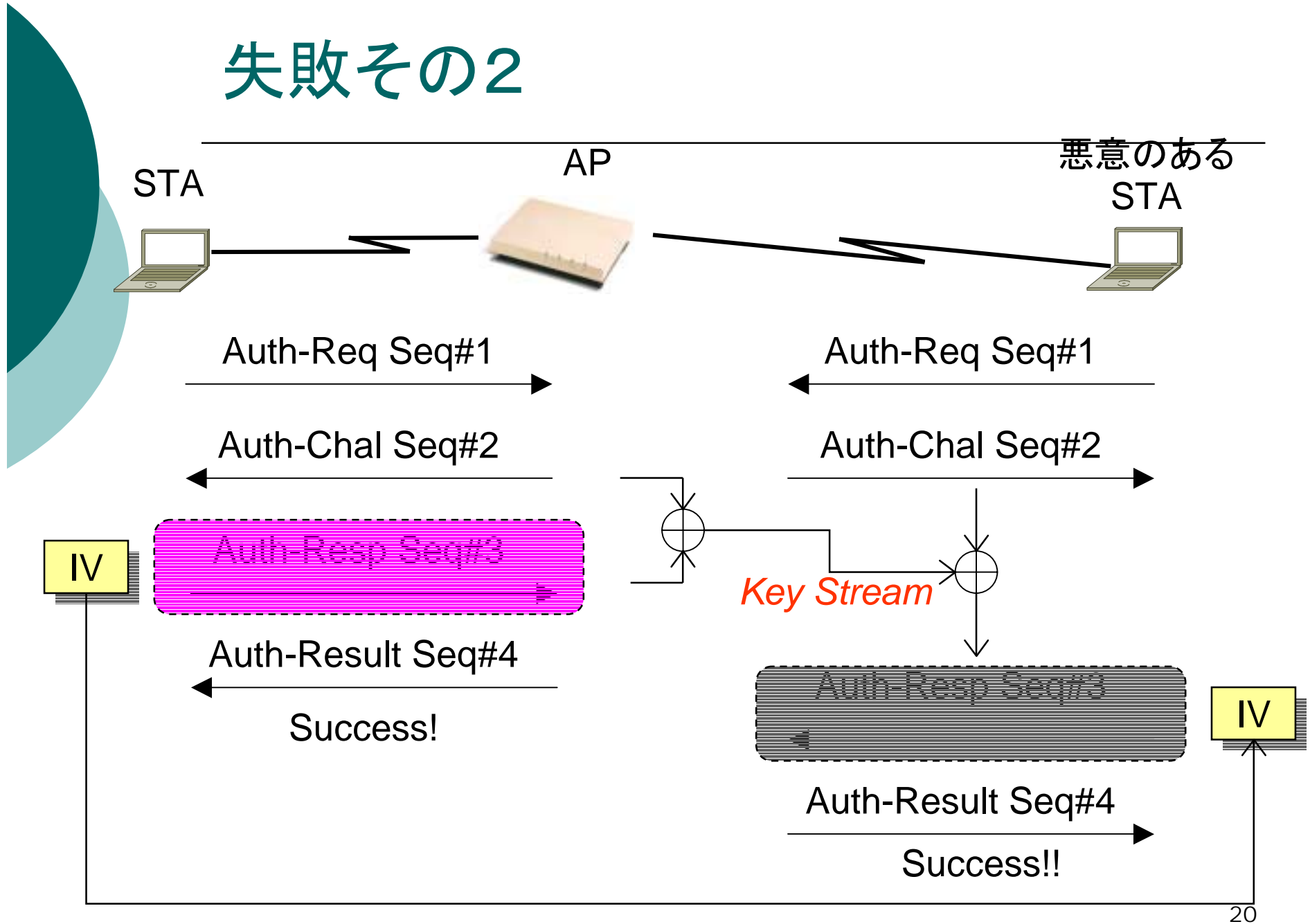
## ○ WEP を使う！

- AP は Challenge (128bytes) を送出
- STA はそれを WEP で暗号化して AP へ送る
- AP はそのフレームの整合性をチェック

# 802.11 Authentication Management Frame



# 失敗その2



# WEP is completely broken!!

---

秘匿性



完全性



認証



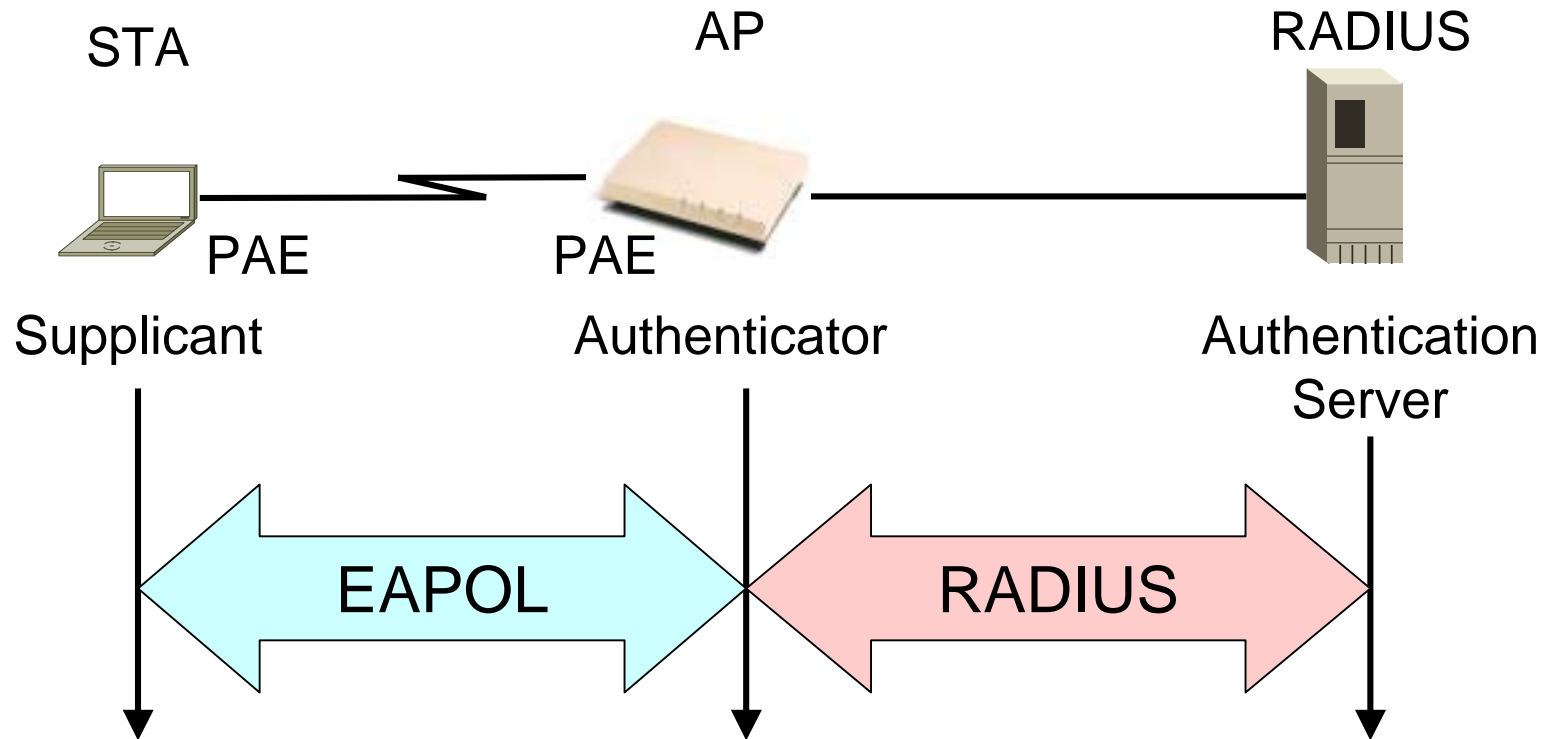


## 対策(3-2) 802.1X

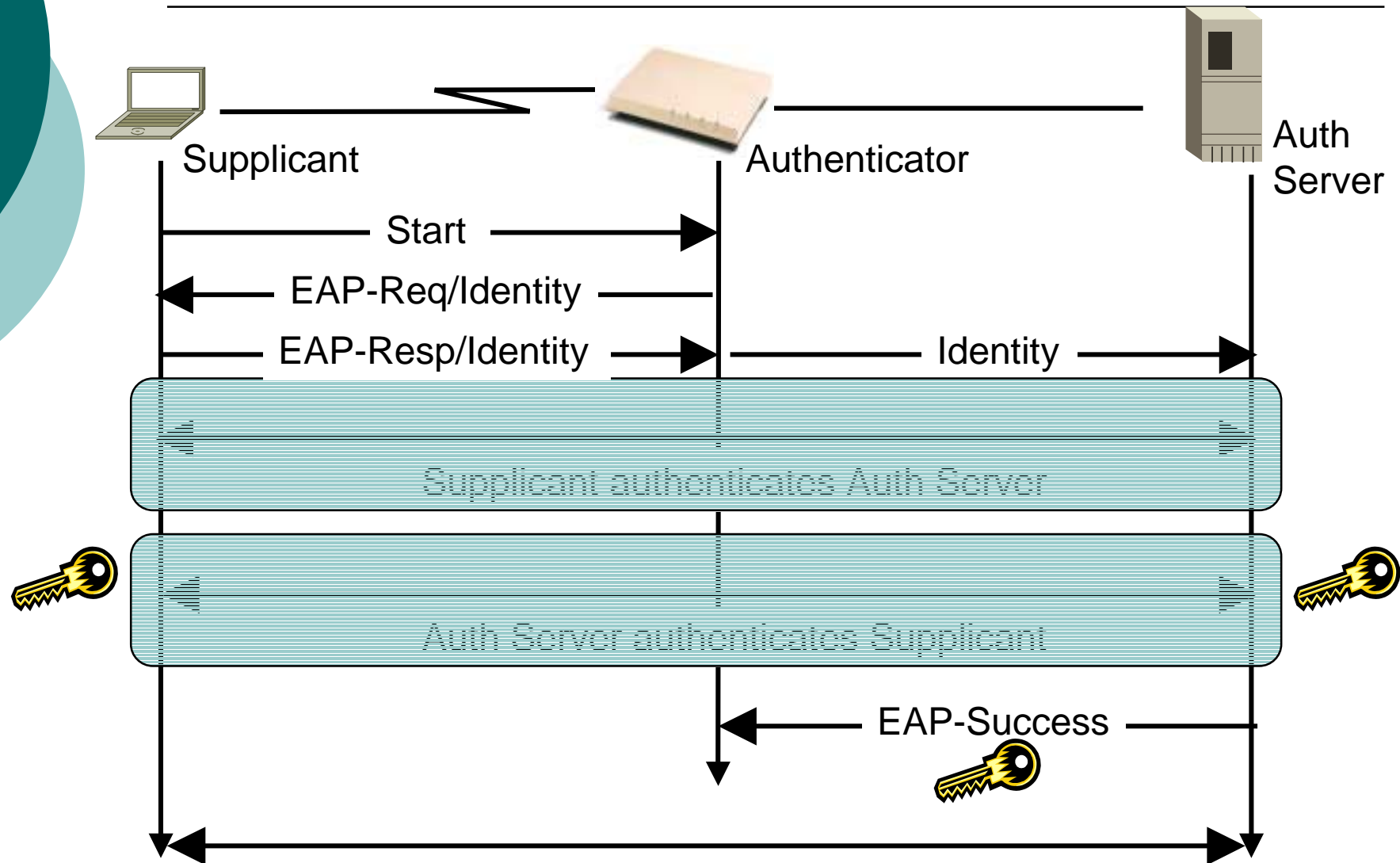
---

- Port-Based Network Access Control
- a.k.a 802.1aa
- 認証を「ユーザーベース」できちんとしよう！
- 鍵配送の仕組みを提供しよう！
  - 管理上のスケーラビリティ
  - 暗号化方式の脆弱性を「和らげる」

# 802.1X の構成要素



# 802.1X の動き







# TLS (Transport Layer Security)

---

- TLS Version 1.0
  - a.k.a SSL version 3.1
- Certificate ベース
- なぜ TLS ??
  - 相互認証
  - セッション鍵
  - 広く受け入れられているから



# EAP (Extensible Authentication Protocol)

---

- PPP から生まれたプロトコル
- 基本的に何でもあり！
  - 単純な Request – Response 型のプロトコル
- この上で激しく色々なことができる
  - MD5 (4)
  - EAP-TLS (13)
  - EAP-Cisco Wireless (17)
  - EAP-TTLS (21)
  - EAP-3Com Wireless (24)
  - PEAP (25)
  - MS-EAP-Auth (26)

# EAP Type

---

EAP Type	Open/ Proprietary	Mutual Auth	Authentication Credentials		Key Material	User Name In Clear	RFC
			Supplicant	Authenticator			
MD5	Open	No	Username/Pwd	None	No	Yes	1321
TLS	Open	Yes	Certificate	Certificate	Yes	Yes	2716
TTLS	Open	Yes	Username/Pwd	Certificate	Yes	No	IETF Draft
PEAP	Open	Yes	Username/Pwd	Certificate	Yes	No	IETF Draft
LEAP	Proprietary	Yes	Username/Pwd	None	Yes	Yes	NA



## WPA の目標

---

- 暗号的脆弱性の排除
- ユーザーベースの認証
- 鍵の配布をサポートすること
- 動的なユーザー・セッション・パケット毎の鍵を使用
- 認証サーバーを強要しないこと
- 2003年中に利用可能になること
- ソフトウェアアップグレード可能



# WPA (Wi-Fi Protected Access)

---

- 802.11i のサブセット
- 認証
  - 802.1X + EAP
- 秘匿性(暗号化)
  - 802.1X 動的鍵配布
  - TKIP
- 完全性
  - Message Integrity Check (MIC) “Michael”

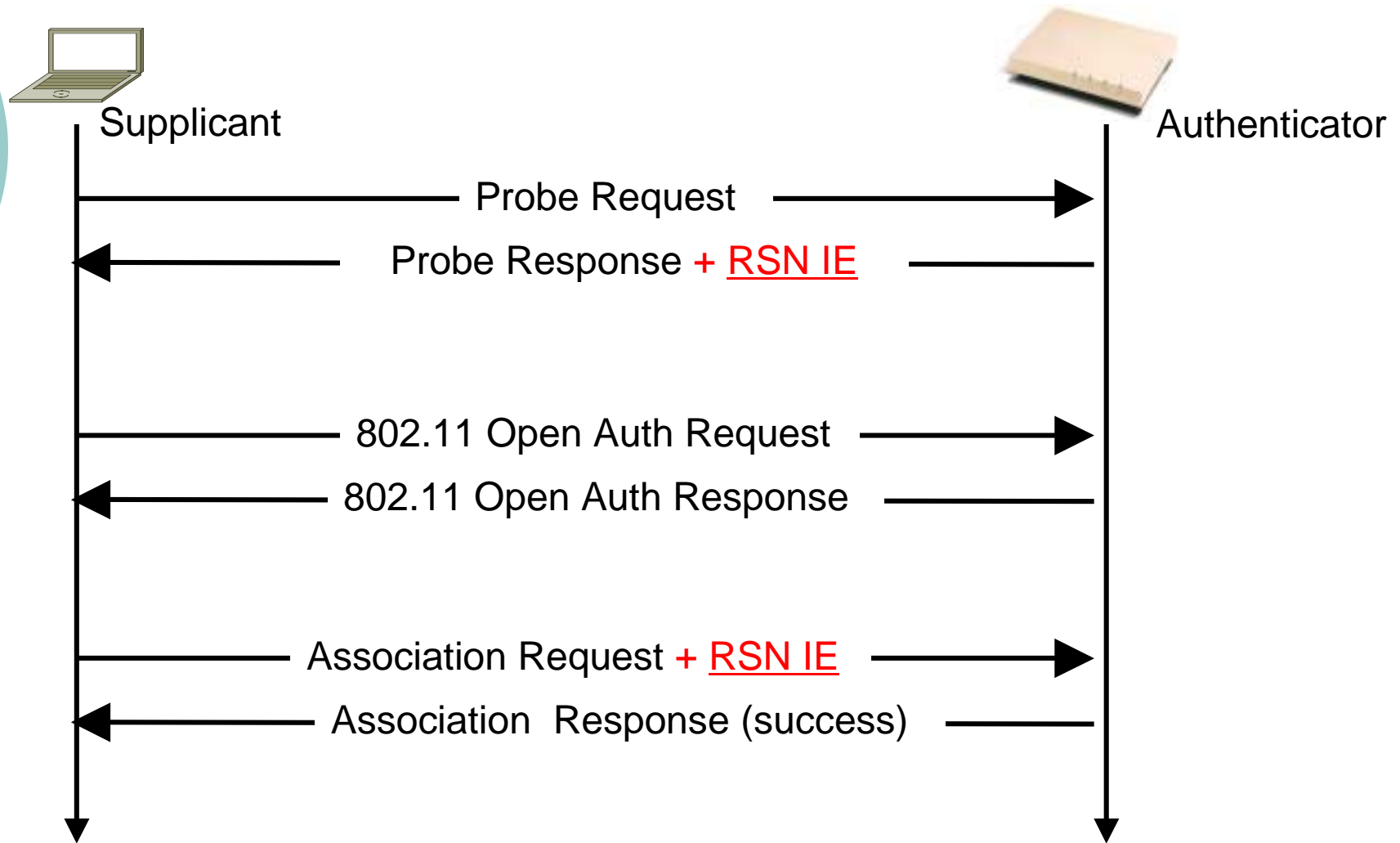


## WPA ステップ

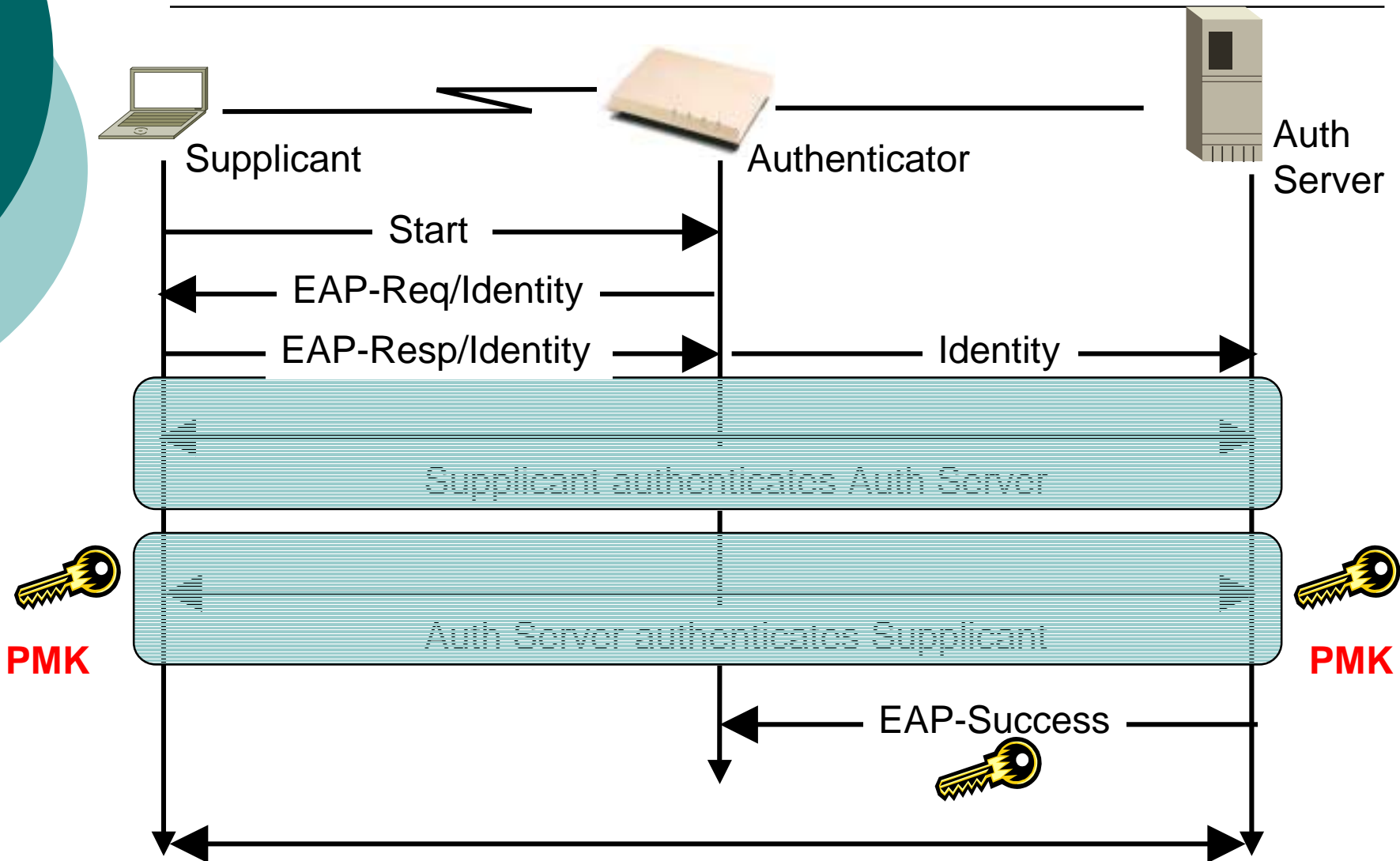
---

- アソシエーションとキーパビリティの確認
- 802.1X 認証と PMK (Pairwise Master Key) の配布
- TK (Temporal Key) の導出
- GK (Group Key) の導出
- 暗号化および整合性チェック

# アソシエーションとケーパビリティの確認

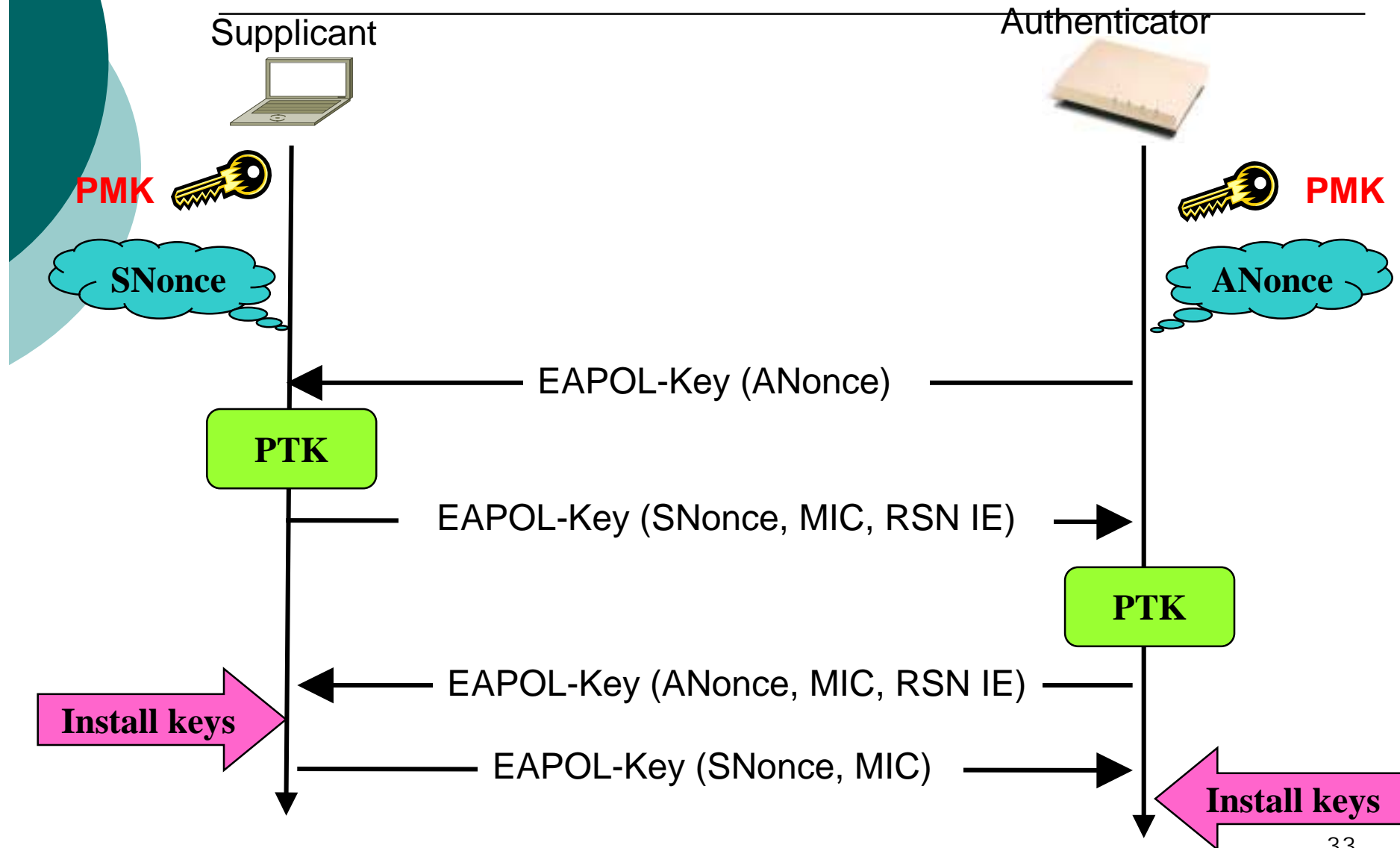


# 802.1X 認証と PMK の配布

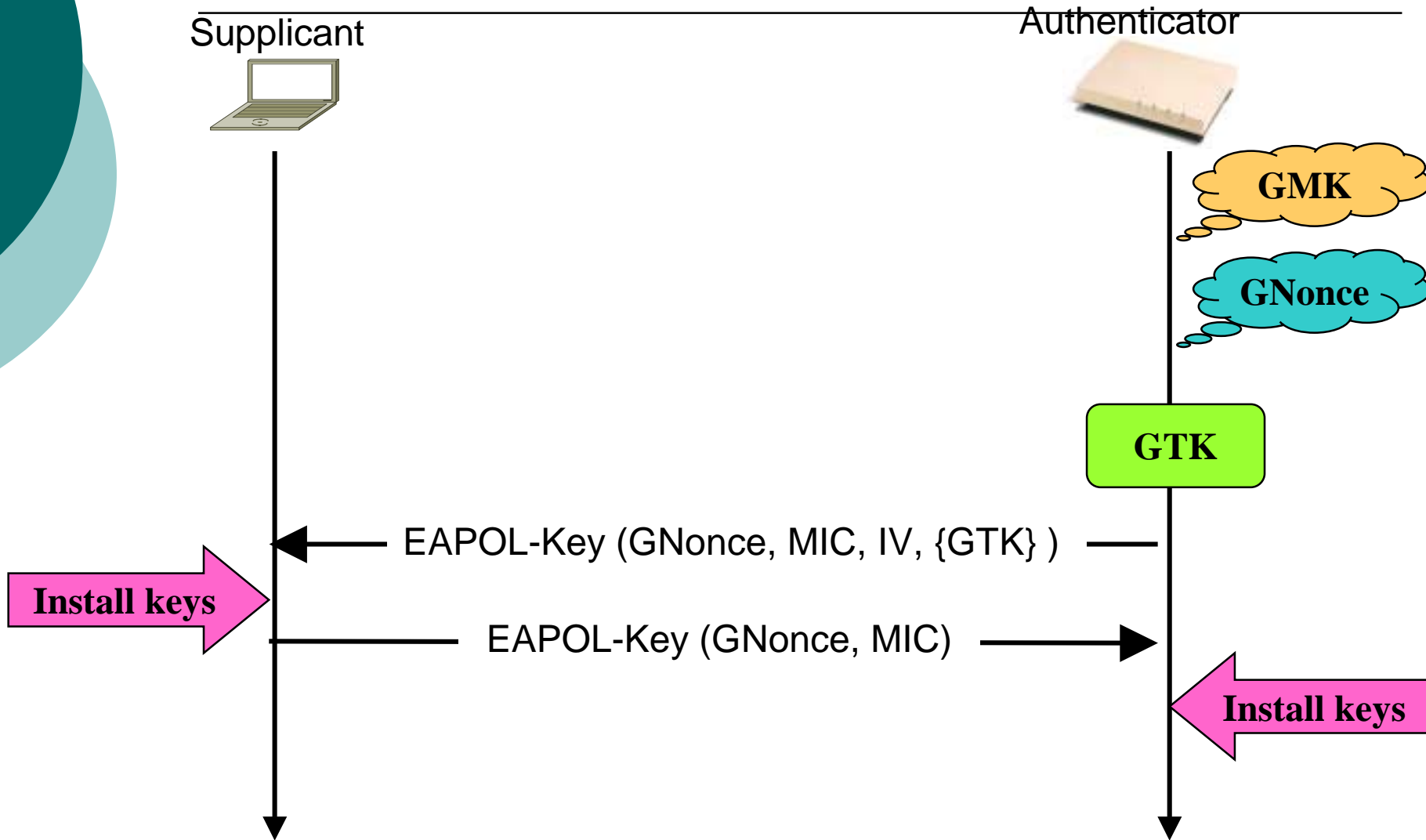




# Temporal Key の導出 ~ 4 way handshake ~



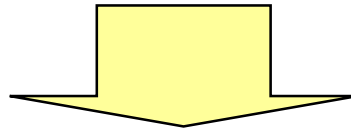
# Group Key の導出 ~ 2 way handshake ~



# Pairwise Key Hierarchy (for TKIP)

---

**Pairwise Master Key  
(PMK)  
256 bits**



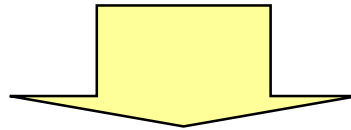
**Pairwise Transient Key (PTK)  
512 bits**

<b>EAPOK-Key MIC Key 128 bits</b>	<b>EAPOL-Key Encryption Key 128 bits</b>	<b>Temporal-Key 128 bits</b>	<b>Data MIC key 128 bits</b>
-------------------------------------------	--------------------------------------------------	----------------------------------	--------------------------------------

# Group Key Hierarchy (for TKIP)

---

**Group Master Key  
(GMK)  
128 bits**



**Group Transient Key (GTK)  
256 bits**

**Temporal-Key  
128bits**

**Data  
MIC key  
128bits**



# PRF (Pseudo Random Function)

---

**H-SHA-1( $K, A, B, X$ )**

**$\leftarrow \text{HMAC-SHA-1}(K, A \parallel 0 \parallel B \parallel X)$**

**PRF- $n(K, A, B) = \text{PRF}(K, A, B, n)$**

**where  $n$  be 128, 192, 256, 384, or 512**

**PRF( $K, A, B, Len$ )**

**for  $i \leftarrow 0$  to  $(Len + 159) / 160$  do**

**$R \leftarrow R \parallel \text{H-SHA-1}(K, A, B, i)$**

**return  $L(R, 0, Len)$**



# PRFの使用例

---

## ○ Nonce

- PRF-256 (Random number, “Init Counter”, Local MAC Address || Time)

## ○ PTK for TKIP

- PRF-512(PMK, “Pairwise key expansion”, Min(AA, SA) || Max(AA, SA) || Min(ANonce, SNonce) || Max(ANonce, SNonce))
- $n = 384$  for CCMP, WRAP and WEP

## ○ GTK for TKIP

- PRF-256(GMK, “Group key expansion”, AA || GNonce)
- $n = 128$  for CCMP, WRAP, and WEP

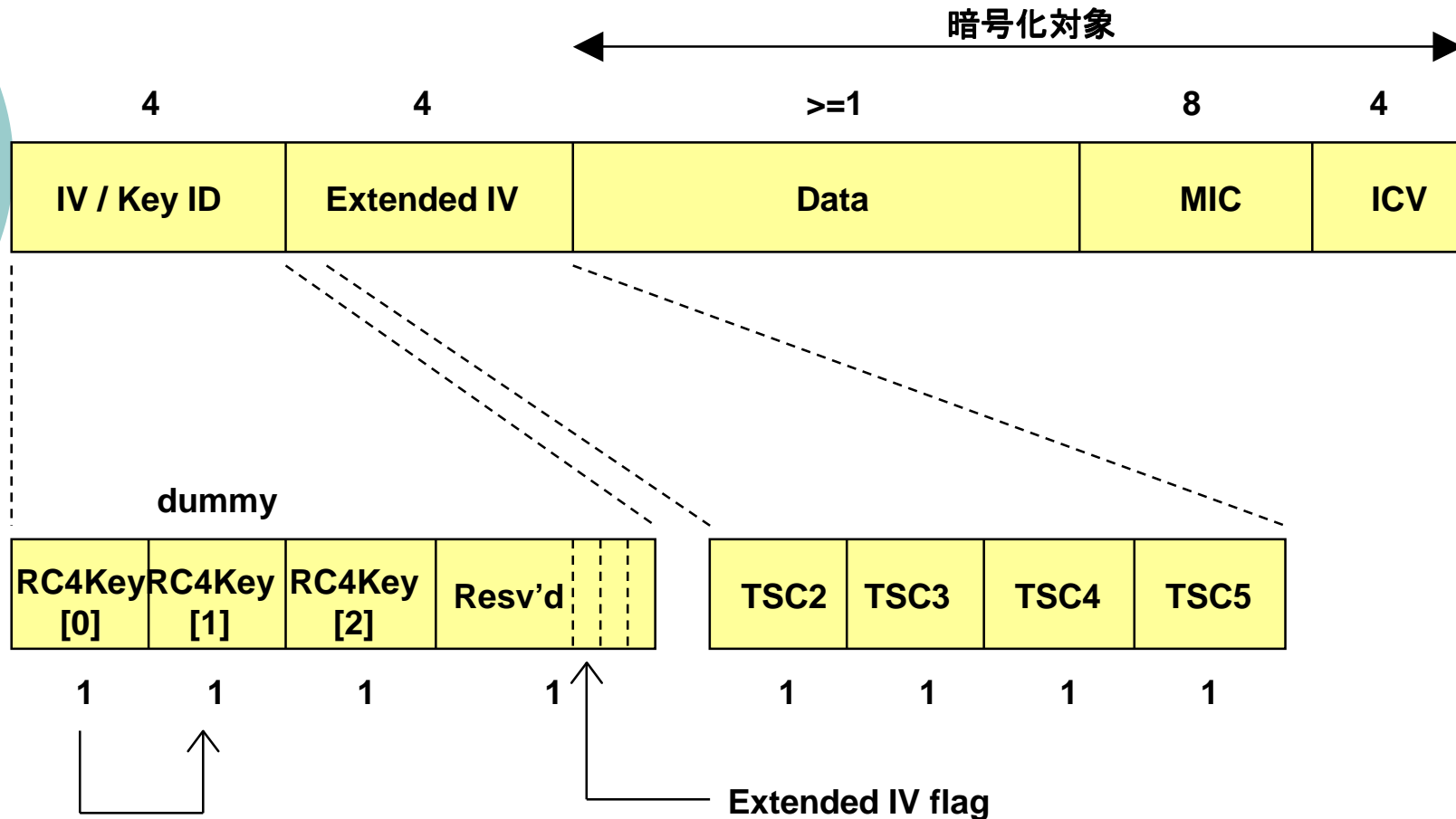


# TKIP (Temporal Key Integrity Protocol)

---

- IV 空間の拡張 (24 -> 48 bits)
- IV シーケンス処理の規定
- Per-packet-mixing Function
- Michael MIC (Message Integrity Code)

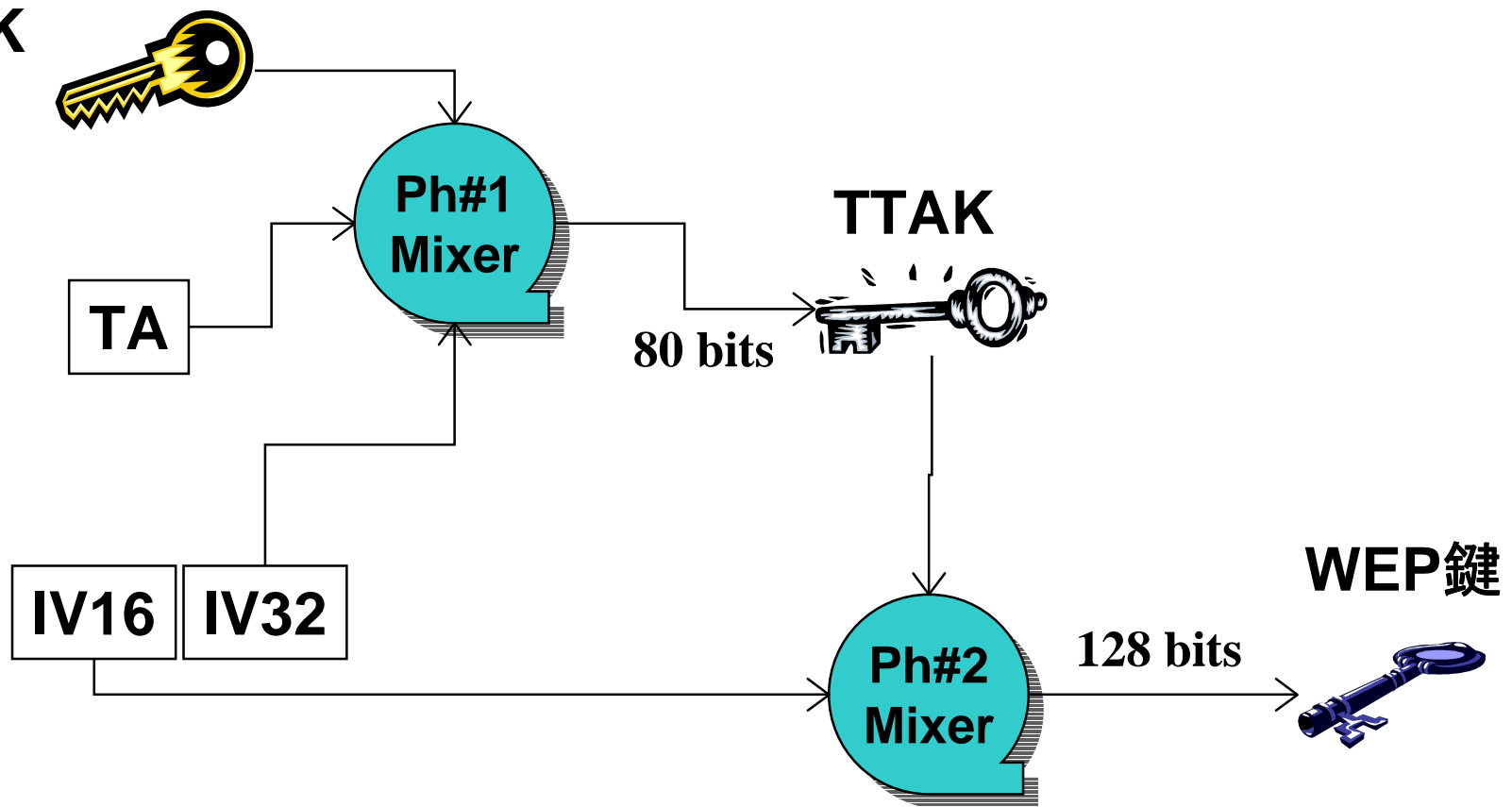
# TKIP Frame Format



Bit 5 is set and  
Bit 4 is cleared



# Per-packet-mixing function





# What's Michael ?

---

- Niels Ferguson によって考えられたメッセージダイジェスト関数の一種
- 8 octets の hash 値を生成
- MSDU に対して行われる
- 守られるのは、
  - Destination MAC address
  - Source MAC address
  - Data



# Why Michael ?

---

- 与えられた CPU サイクルはごく僅か
  - MD5 や SHA-1 は使えない
  - 演算を慎重に選ぶ必要あり
- 設計上のゴールは 20 bits の強度を持つこと
  - 現在知られている最も強力な攻撃は  $2^{29}$  個のメッセージを使った差分暗号解析
- Countermeasure が必要



# Michael Countermeasure (AP)

---

- Multicast Frame の MIC Failure
  - 1) Group Key を捨て、マルチキャストの送信を止め、ログを記録し、blackout timer (60秒) を開始する。
  - 2) blackout 中に再度 MIC failure があった場合は、blackout が解けるまで Group Key の生成を待つ。
  - 3) 2 way handshake による Group Key の生成。
- Unicast Frame の MIC Failure
  - 1) ログを記録し、blackout timer (60秒) を開始。
  - 2) 802.1X フレーム以外の送受信をストップ。
  - 3) blackout 中に再度 MIC failure があった場合は、blackout が解けるまで Pairwise Key の生成を待つ。
  - 4) 4 way handshake による Pairwise Key の生成。



# Michael Countermeasure (STA)

---

- Multicast Frame の MIC Failure
  - 1) Group Key の削除。
  - 2) Access Point に新しい Group Key をリクエスト。
  - 3) ログの記録。
- Unicast Frame の MIC Failure
  - 1) 802.1X フレーム以外のフレームの送受信をストップ。
  - 2) Access Point に新しい Pairwise Key をリクエスト。
  - 3) ログの記録。



# Is Michael subject to DoS ??

---

- 理論的には可能
- 実際にはちょっと面倒
  - IV replay protection をかいくぐり、
  - ICV のチェックをパスしなければならない。
- もっと簡単な DoS があるじゃない！
  - Disassociation or Deauthentication 攻撃
  - RF jammer



# PreShared Key (PSK) Mode

---

- RADIUS を使用しない(用意できない)場合を想定
  - ホームユース
- 802.1X で実現していた部分を手動設定で代替
  - 認証
  - PMK の配布
  - 802.1X 以降の動き(4 and 2 way handshake, 鍵の導出、TKIP、等)は non-PSK 時と同様
- PMK (256bits) を AP, STA 双方に設定
- Pass Phrase から 256 bits PMK を生成する際の推奨方法も別途規定
  - PKCS#5 PBKDF2 (Password-Based Key Derivation Function)




## WPA PSK は安全か？ ～active attack 編～

---

- (WEP と同様)仕組み的には per-user で適用できるが、(これまた WEP と同様)ほぼ全ての実装で ESS 内で共通の PSK を用いる
  - PSK を知っていれば他のユーザーのトラフィックは解読できる





## WPA PSK は安全か？ ~passive attack 編~

---

- PSK = PBKDF2(PassPhrase, SSID, SSID length, 4096, 256)
- Pass Phrase : 8 ~ 63 文字
- $n$  文字の Pass Phrase のエントロピー
  - $2.5 * n + 12$  bits
  - 64bit のエントロピーを得るためには21文字程度、104bit のエントロピーを得るには37文字程度必要
- 十分な長さのない Pass Phrase を使うと dictionary attack 可能！



# PSK in IPsec vs PSK in WPA

---

- IPsec の PSK は認証にしか使わない！
  - 鍵はあくまで DH 鍵交換によって得られる
  - 仮に PSK が分っていても、passive に decrypt することはできない
- WPA の PSK (PMK) は鍵の生成に関与する
  - $PTK = PRF-512(PMK, \text{"Pairwise key expansion"}, \text{Min}(AA, SA) || \text{Max}(AA, SA) || \text{Min}(ANonce, SNonce) || \text{Max}(ANonce, SNonce))$
  - Nonce はアソシエーション時に交換される
  - それを取り逃がしてしまったら deassociation attack すれば良い！



# Pass Phrase からの WEP key 生成

---

- 標準ではないが、多くのベンダーが実装している
- LCG-based derivation for 40bits key
- MD5-based derivation for 104bits key

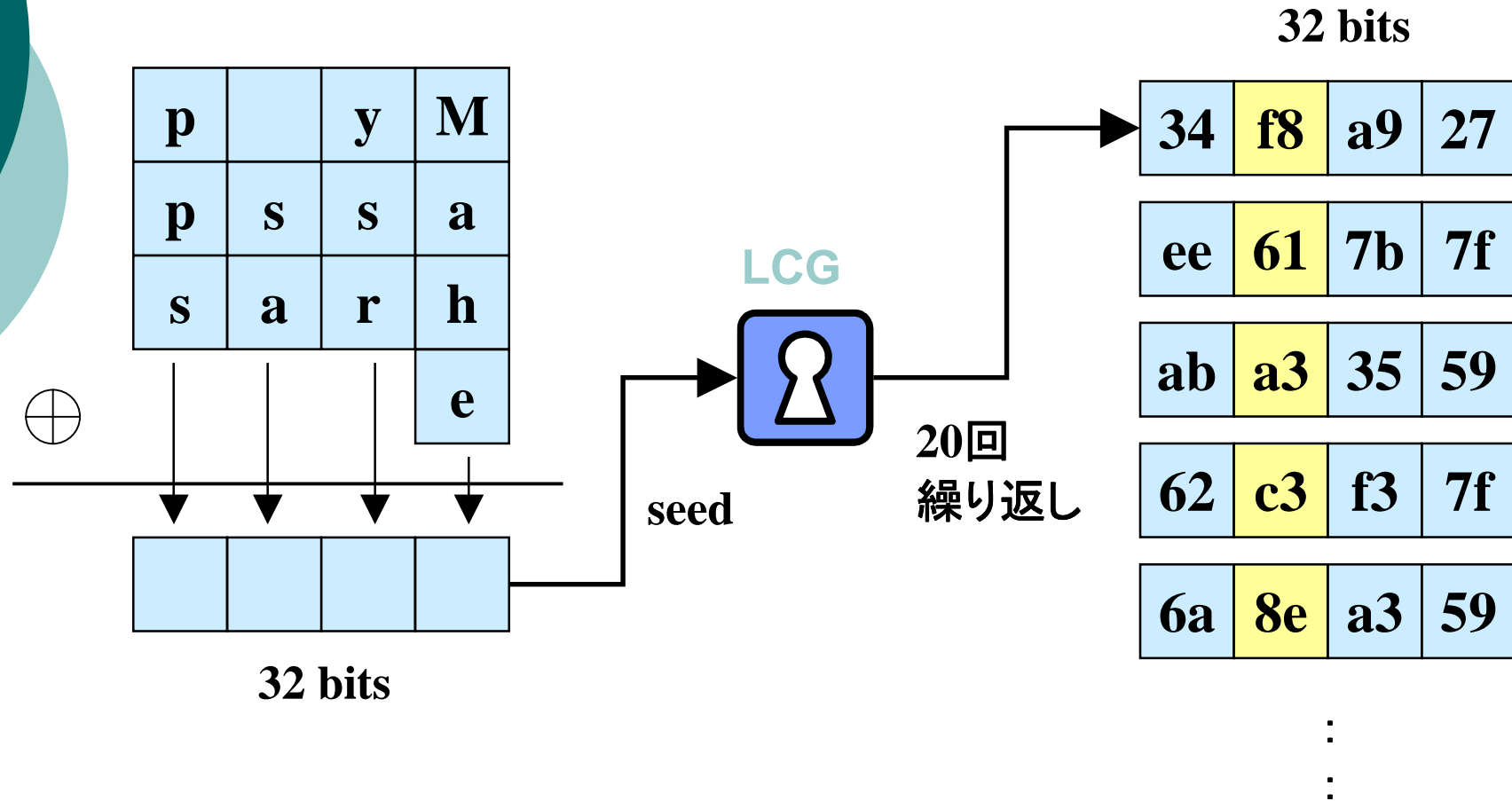


# Linear Congruential Generator

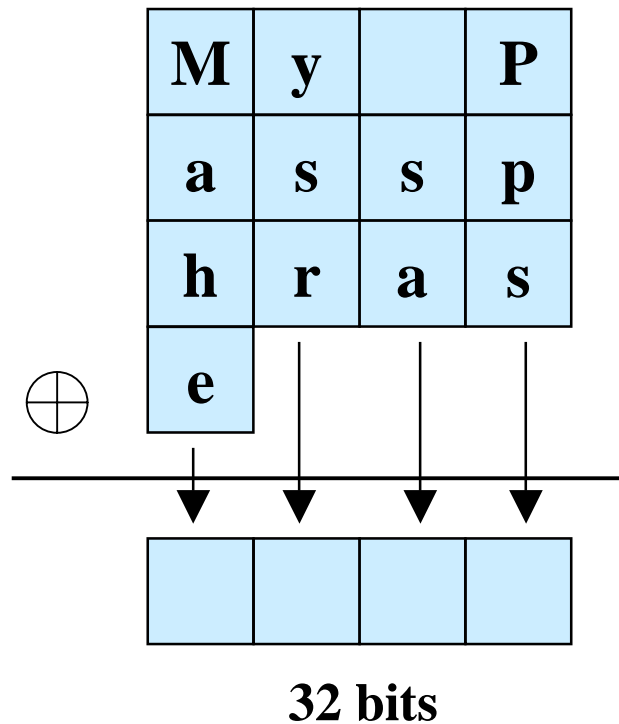
---

- LCG  $(m, a, b, y_0) \leftarrow y_{n+1} = a * y_n + b \pmod{m}$ 
  - $m, a, b, y_0$  は任意(だが注意深く選ぶ必要あり)
- 多くのライブラリの `rand()` で使われている
- LCG  $(2^{31}-1, 7^5=16807, 0, \text{seed})$   
aka MINSTD
  - Microsoft

# 40bit Key Derivation from Pass Phrase

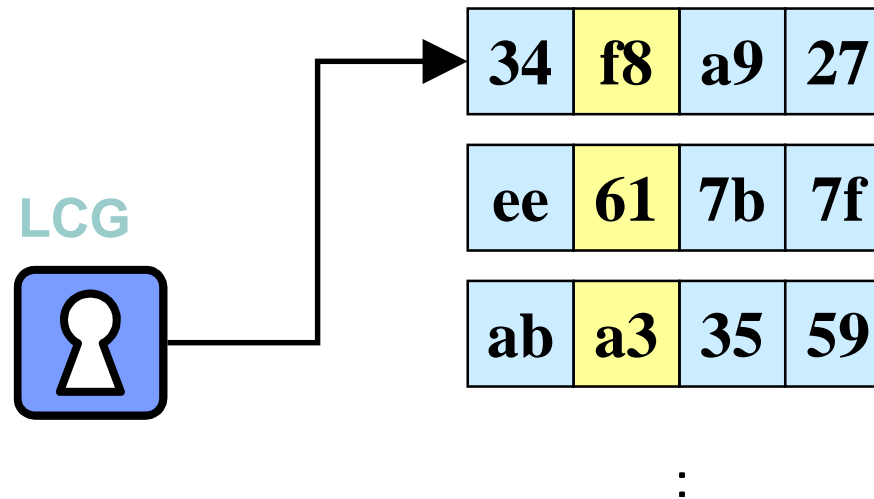


# エントロピーの低下(1)



- Seed の各 octet の MSB は必ず 0 になる !
- 00:00:00:00 ~ 7f:7f:7f:7f

## エントロピーの低下(2)



- Seed の Bit 24~31 は無関係
- 00:00:00:00 ~ 00:ff:ff:ff



## 結果的に

---

- 00:00:00 ~ 00:7f:7f:7f (21bits) のシードだけ調べればよい！
- 総当たり攻撃に対する耐性が 40 bits から 21 bits への低下





## WEPの問題点(再掲)

---

- 鍵長が 40bit と短い
- ICV に CRC32 を用いている
- 一つの鍵を使い続ける
- 鍵の配布メカニズムがない
- IV の空間が小さい(i.e. 24bit)
- リプレイ攻撃に無力
- FMS 攻撃



## 802.11i (a.k.a WPA2)

---

- CCMP (Counter-mode with CBC MAC Protocol)
  - AES が前提
- WRAP (option)
- TKIP (option)
- Secure IBSS
- Secure fast handoff
- Pre-Authentication
- Security Capability Discovery
- ...

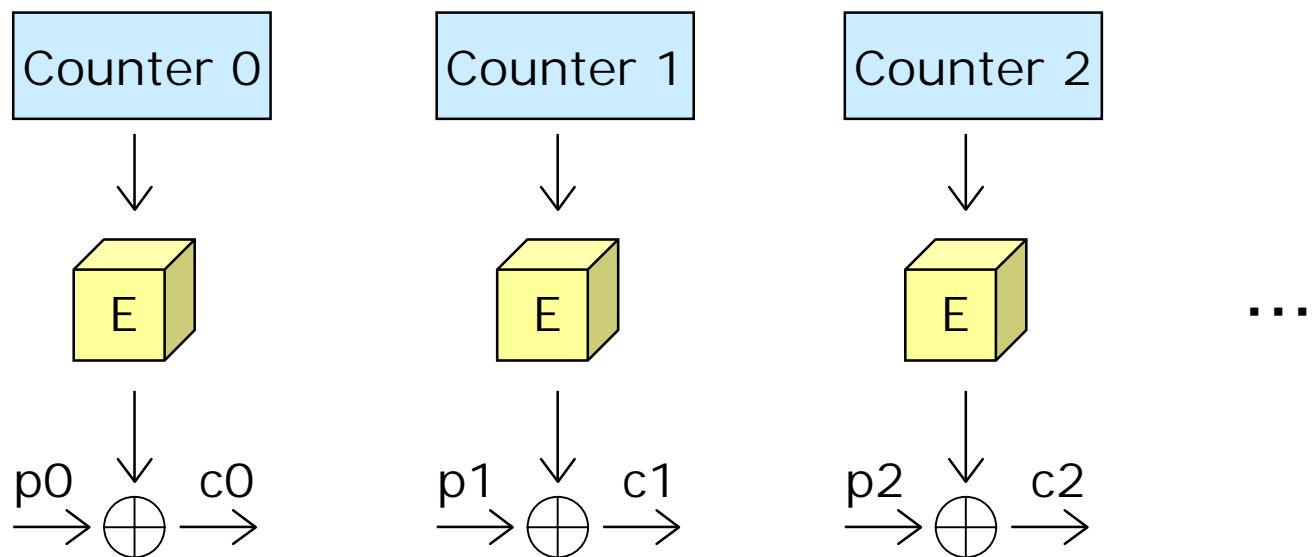


# CCMP

---

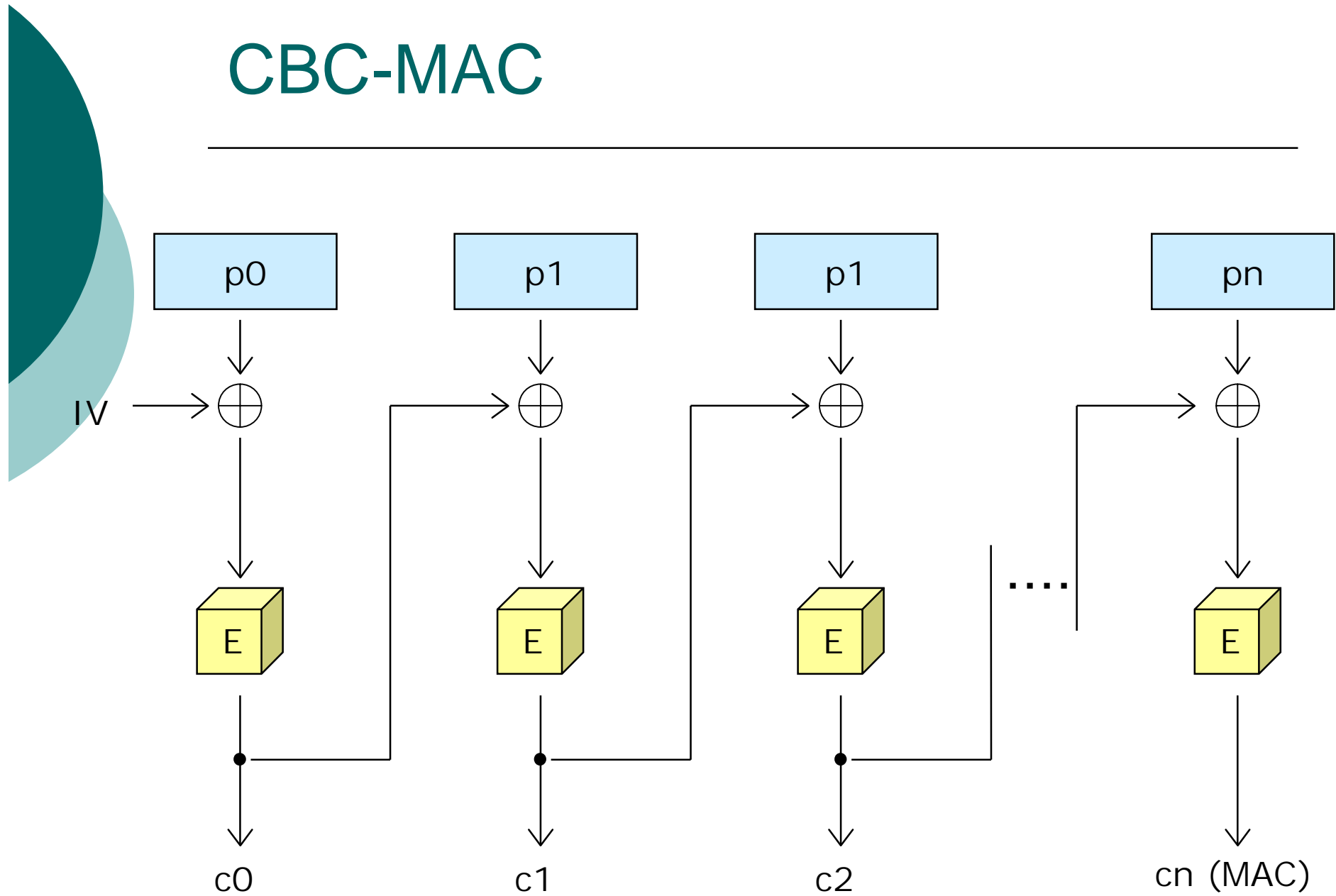
- Counter-mode CBC-MAC Protocol
  - AES を “Counter mode” で使用
  - AES で “CBC-MAC” も計算
- 暗号化と整合性検証を同時に実現する！
- RFC 3610

# Counter-Mode



- 復号化も全く同じプロセスで良い
- 並列化可能
- ランダムアクセス
- 事前に計算しておける
- メッセージはブロックサイズに依存しない

# CBC-MAC



# WEP, TKIP and CCMP

	WEP	TKIP	CCMP
暗号化アルゴリズム	RC4	RC4	AES
鍵長 (bits)	40, 104 or 128	104 (encrypt) 64 (auth)	128
IV (bits)	24	48	48
データ部の完全性	CRC32	Michael	CCM
ヘッダ部の完全性	なし	Michael	CCM
Anti-Replay-Attack	なし	あり	あり



# 結論

---

- ワイヤレスは安全??
  - 危ない!
    - もし、使い方を誤れば
- 今日ワイヤレスを使っても大丈夫?
  - YES!
    - 使い方を誤らなければ
  - 道具は揃っている!
- 人間は過ちを犯すもの
  - 直せばよい!
  - でも時間がかかる
  - 十分に時間が経ってきた!
- 「リスク」と「利益」をよく考えよう!



# 略語一覽

---

AES	Advanced Encryption Standard	PKCS	Public Key Cryptographic Standard
AP	Access Point	PMK	Pairwise Master Key
CBC	Cipher Block Chaining	PPP	Point-to-Point Protocol
CCMP	Counter-mode CBC MAC Protocol	PRF	Pseudo Random Function
CFB	Cipher Feedback	PRNG	Pseudo Random Number Generator
CRC32	Cyclic Redundancy Check 32bits	PSK	PreShared Key
DoS	Denial of Service	PTK	Pairwise Transient Key
EAP	Extensible Authentication Protocol	RADIUS	Remote Access Dial-Up System
EAPOL	EAP over LAN	RC4	Rivest Code (or Cipher) 4
ECB	Electronic Code Book	RSN	Remote Secure Network
ESS	Extended Service Set	SHA1	Secure Hash Algorithm 1
FCS	Frame Check Sum	SSID	Service Set Identifier
GK	Group Key	STA	Station (client)
GMK	Group Master Key	TA	Transmit (MAC) Address
ICV	Integrity Check Value	TK	Temporal Key
IE	Information Element	TKIP	Temporal Key Integrity Protocol
IV	Initialization Vector	TLS	Transport Layer Security
LCG	Linear Congruential Generator	TTAK	TKIP-mixed Transmit Address and Key
LEAP	Lightweight EAP	TTLS	Tunneled TLS
MAC	Message Authentication Code	WEP	Wired Equivalent Privacy
MD5	Message Digest 5	WPA	Wi-Fi Protected Access
MIC	Message Integrity Code	XOR	Exclusive OR
OFB	Output Feedback		
PAE	Port Authentication Entity		
PBKDF	Password-Based Key Derivation Function		
PEAP	Protected EAP		