認証暗号の基礎と最近の研究事例

情報セキュリティ・セミナー

共通鍵暗号の安全な使い方～暗号利用モードの基本とリスク～

NEC セキュアシステムプラットフォーム研究所　峯松 一彦

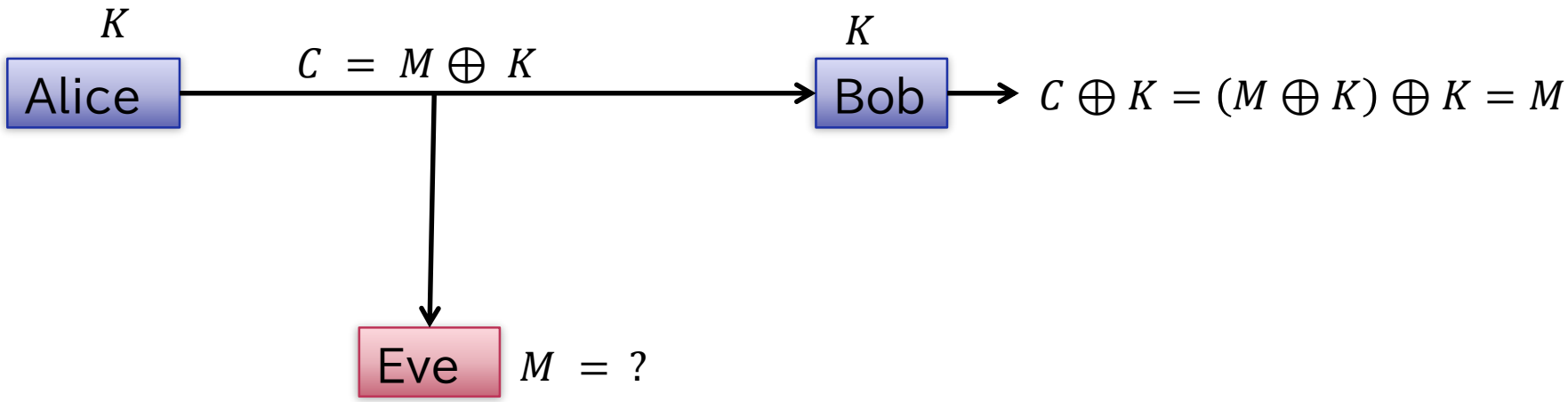
2024年12月16日

目次

認証暗号の基礎

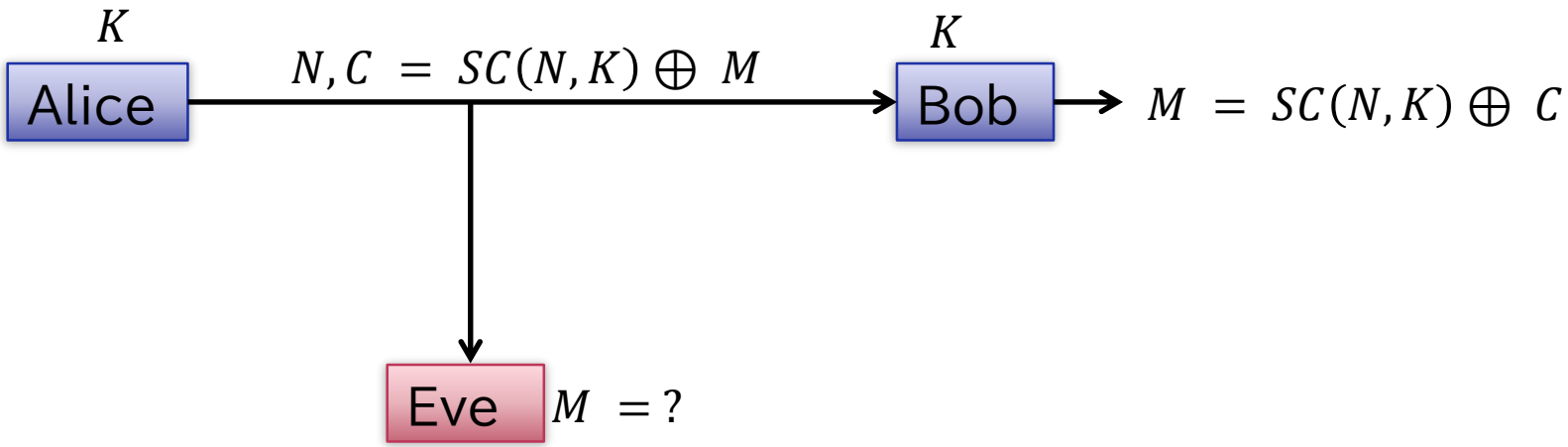
(共通鍵)暗号化とは

- AliceとBobで同じ鍵 K を共有し、通信する
- 古典的な暗号化の例: one-time pad
  - 送信したいメッセージ M、鍵 K、共に 同じビット長
  - 暗号文 $C = M \oplus K$ (ビットごとの排他的論理和, XOR). 復号はKをXORする
  - Eveは盗聴ができない:C からはMの情報は得られない(情報理論的安全性)



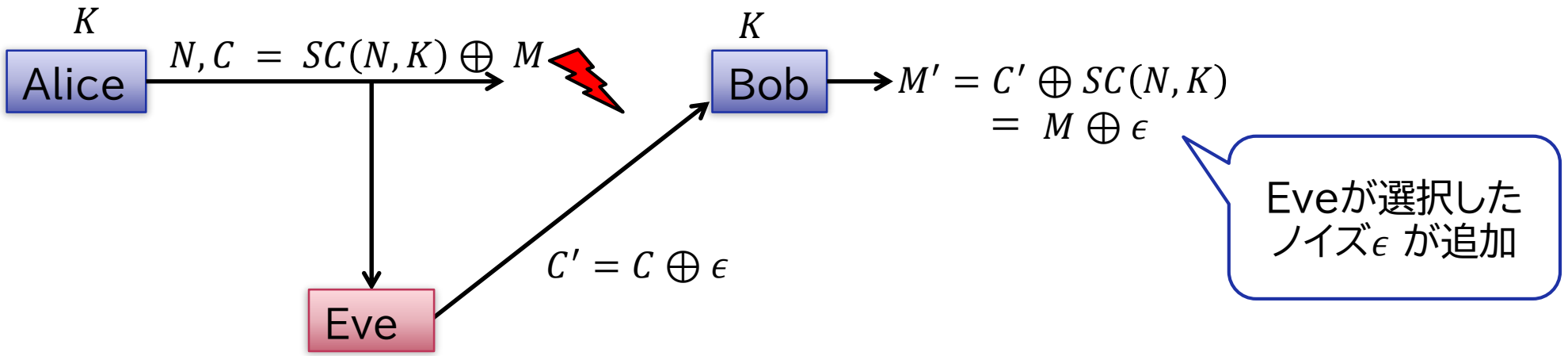
実用的な暗号化の例

- OTP は一般に非実用的(鍵より長い平文を送れない)
- ストリーム暗号
  - ナンス N と鍵 K を用いて可変長の鍵ストリームを生成
  - N は暗号化ごとにユニークな値(**nonce.** カウンターなど). 通信に含める
- ストリーム暗号が計算量的安全なら盗聴困難



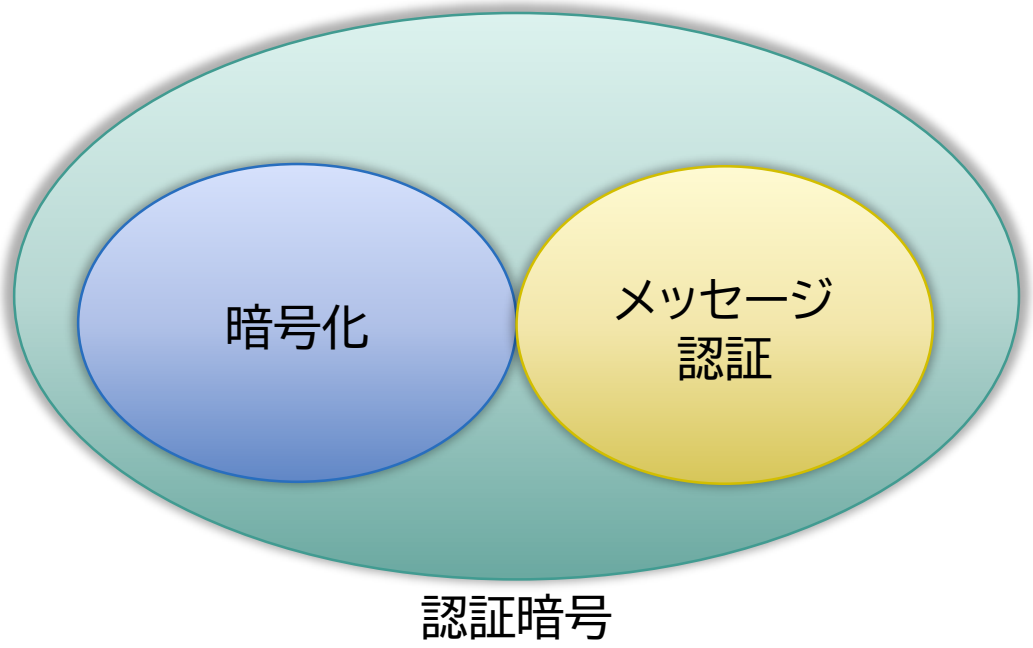
メッセージの真正性

- Eve は改ざんが可能：任意の差分 ϵ を復号結果に適用できる
- 一般に暗号化単体では改ざんを防げない
  - 平文の任意のビットを反転可能. 例えば平文が送金情報（金額）なら狙った桁を反転できる…
  - 原則気づきようがない（復号結果を見て“それっぽい”かどうかで判断するのはリスクがある. ）

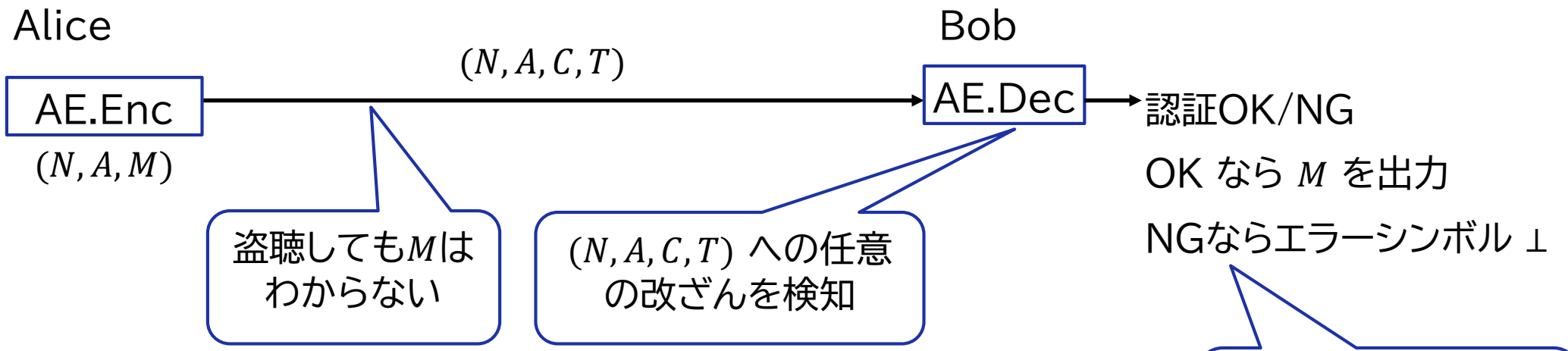


認証暗号

- 認証暗号 ＝ 平文秘匿と改ざん検知を統合した暗号化のこと
  - Authenticated Encryption（AE）
  - 改ざん検知＝メッセージ認証コード　Message authentication code（MAC）



AEのインタフェース ※AD付きAE = AEAD (AE with Associated data)とも呼ぶ

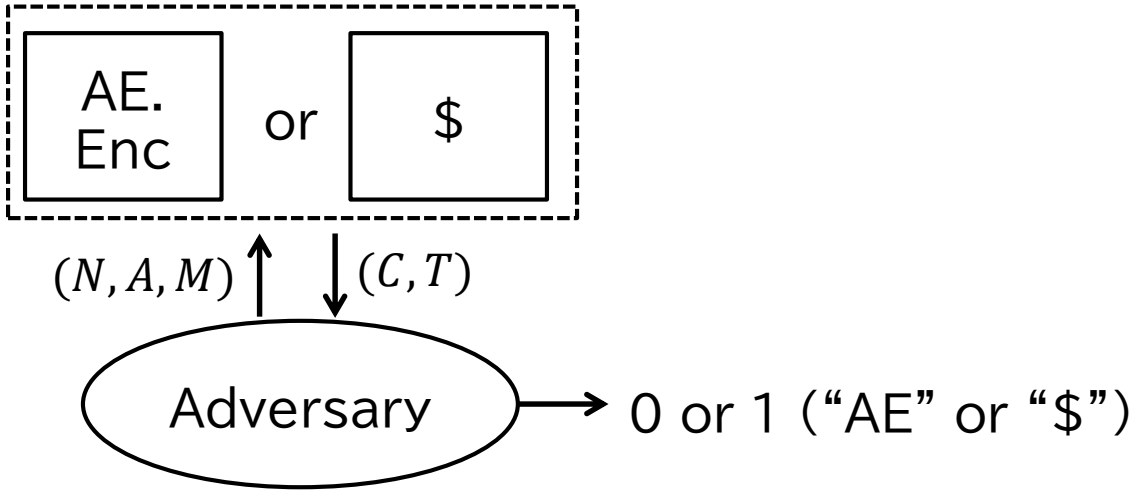


- N：Nonce
- A：暗号化しないが認証は必要な情報（Associated data, **AD**）

  ※プロトコルヘッダ. 受信先アドレスなど
- M：平文(暗号化も認証も必要)
- C：暗号文
- T：認証用の**タグ**

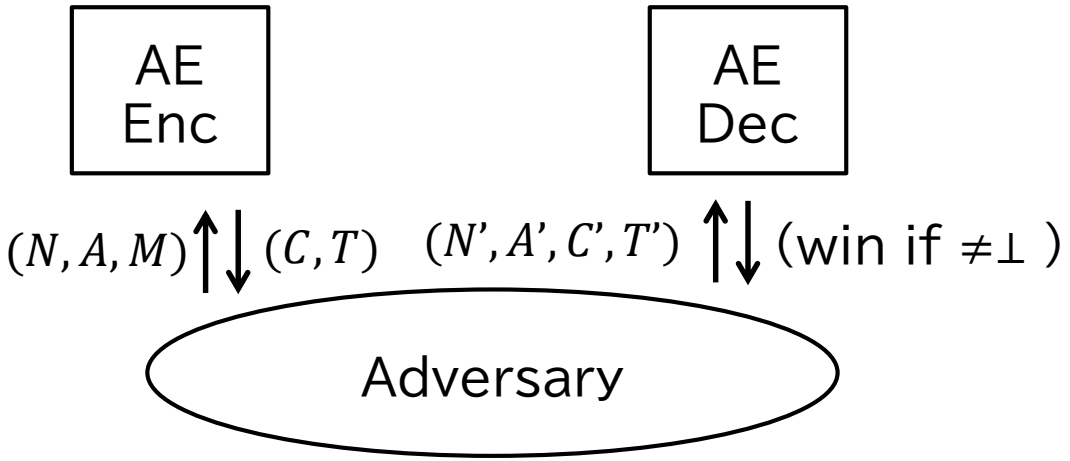
認証暗号の安全性(秘匿性, PRIV)

- PRIV：選択平文攻撃のもと、暗号文と真の乱数との判別困難性
  - $：AE.Enc$(N, A, M)$と同じ長さの乱数を返すオラクル
  - (N, A, M) は攻撃者が選択可能、ただしN の重複は認めない(nonce-based encryption)
  - 攻撃者$\mathcal{A}$の**利得(advantage)** $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{AE}}(\mathcal{A}) = |\Pr[\mathcal{A}^{AE.Enc} = 1] - \Pr[\mathcal{A}^{\$} = 1]|$
    - 利得≒攻撃者の判別成功確率. 小さいほど安全



認証暗号の安全性(真正性, AUTH)

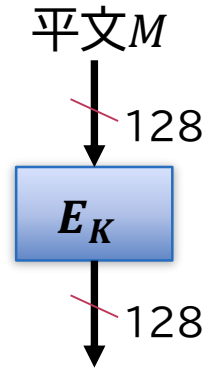
- AUTH：選択暗号文攻撃のもと、**非自明な**偽造の困難性
- 利得(偽造成功確率) $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AE}}(\mathcal{A}) = \Pr[\mathcal{A}^{AE.Enc,AE.Dec} \text{ forges}]$
- $(N_1, A_1, M_1, C_1, T_1), \ldots, (N_q, A_q, M_q, C_q, T_q)$をEncから得たもとで$(N', A', C', T') \neq (N_i, A_i, C_i, T_i), i \in \{1, \ldots, q\}$ をDecにクエリしてエラー(⊥)以外をもらえばforge(偽造成功)
- 復号クエリは複数回可能



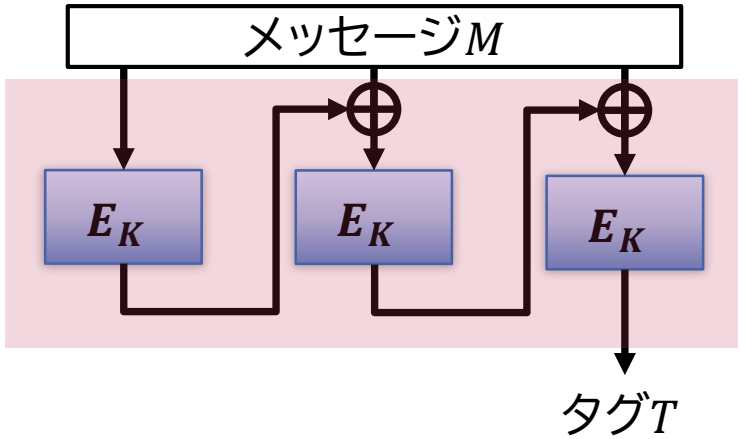
認証暗号の構成方法

背景技術

- 共通鍵暗号の暗号コア(プリミティブ)
  - ブロック暗号: 短い固定長平文を暗号化
  - 例: 128-bit ブロック暗号AES
- 暗号利用モード: 暗号コアを用いた高機能な暗号化
  - 可変長暗号化、改ざん検知、認証暗号etc.
- 多くの認証暗号がプリミティブ＋モードの形をとる
  - NIST標準 GCM/CCM: AESのモード



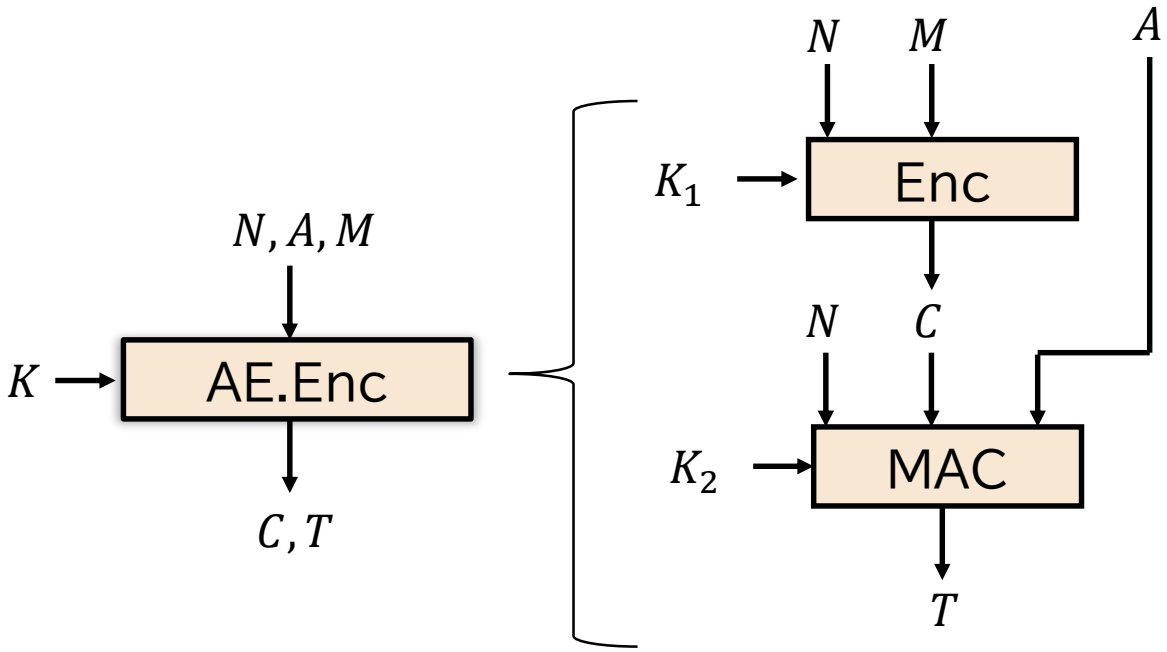
ブロック暗号E



CBC-MAC$[E]$

汎用結合

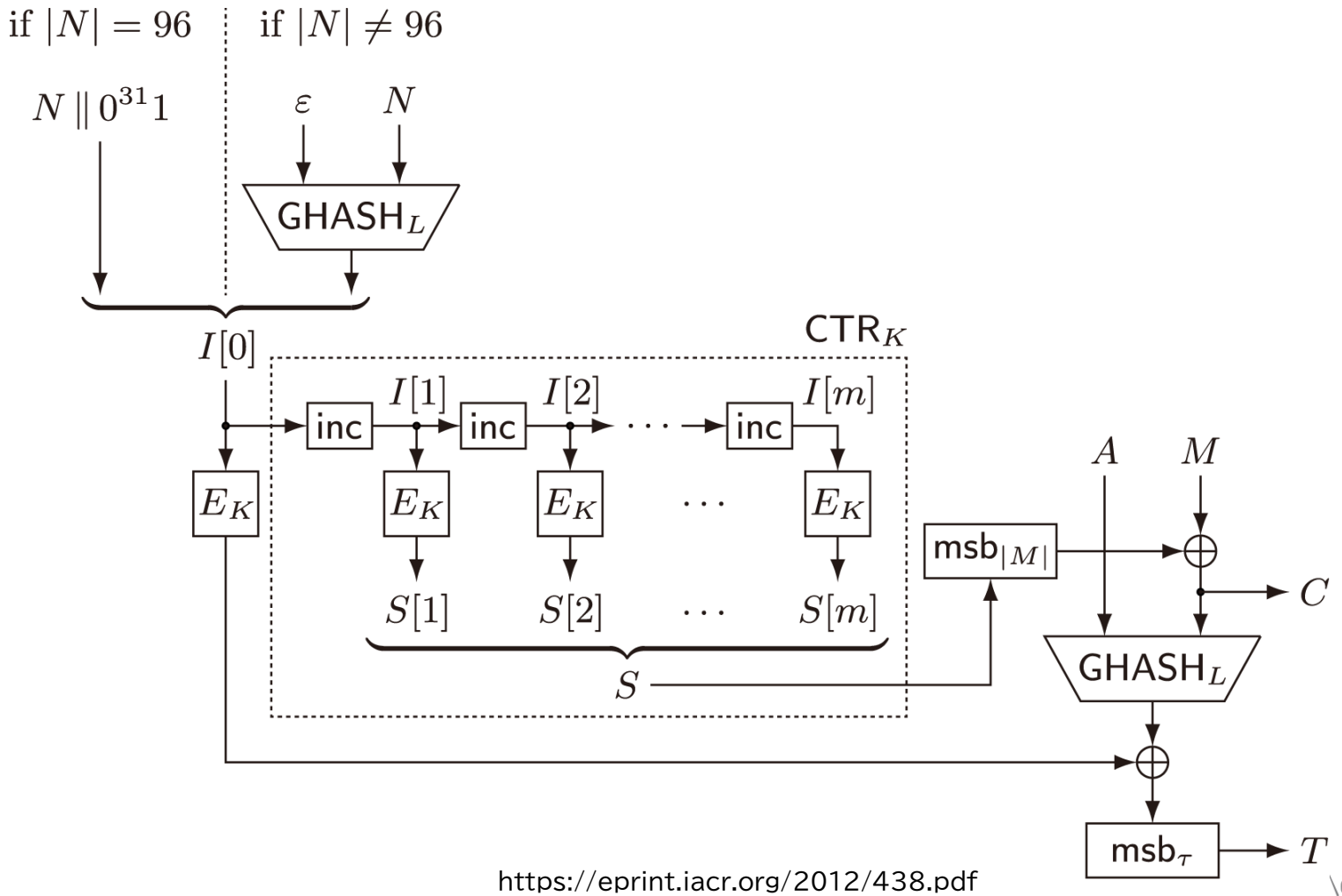
- もっとも古典的なAE構成方法
- 安全な暗号化と安全なMACの組み合わせで実現可能
  - 独立な鍵を二つ使う
  - 全体の安全性は結合方法とベースの仮定による
  - Enc-then-MAC, MAC-then-Enc, Enc-and-MAC



Enc-then-MAC構成

認証暗号の例(GCM)

- NIST推奨方式の一つ. もっとも普及している方式
- AES利用



https://eprint.iacr.org/2012/438.pdf

認証暗号の安全性証明と攻撃

安全性証明とバウンド

- ブロック暗号Eを用いた認証暗号$F[E]$
- 認証暗号$F[E]$が証明可能安全= $F[E]$の安全性をEの安全性（疑似ランダム性）に帰着できること. “$F[E]$を破るにはEを破るしかない”
- 典型的な安全性バウンドの形:

$$\underbrace{\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E]}(\mathcal{A})}_{\text{GCMのPRIV for 敵}\mathcal{A}\text{（乱数との識別成功確率）小さいほど安全}} \leq \underbrace{\mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A}')}_{E_K\text{の敵}\mathcal{A}'\text{に対する安全性}} + \underbrace{\frac{\sigma^2}{2^n}}_{\sigma\text{: }\mathcal{A}\text{がクエリした総ブロック数. }n\text{:ブロックサイズ}}$$

AESなど信頼性の高いEならば小さい（計算量的仮定）

$\sigma \ll 2^{\frac{n}{2}}$ ならば$\ll 1$

両方の項が$\ll 1$ ならばGCM$[E]$はPRIV安全

安全性証明とバウンド

- ブロック暗号Eを用いた認証暗号$F[E]$
- 認証暗号$F[E]$が証明可能安全= $F[E]$の安全性をEの安全性(疑似ランダム性)に帰着できること. “$F[E]$を破るにはEを破るしかない”
- 典型的な安全性バウンドの形:

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A}') + \frac{\sigma^2}{2^n}$$

GCMのPRIV for 敵$\mathcal{A}$
(乱数との識別成功確率)
小さいほど安全

E_Kの敵$\mathcal{A}'$に対する安全性

σ: $\mathcal{A}$がクエリした総ブロッ

AESなど信頼性の高いEならば
小さい(計算量的仮定)

$\sigma \ll 2^{\frac{n}{2}}$ ならば$\ll 1$

逆に言えば、どちらかの項が~1となる状況では安全性が保証されない

両方の項が$\ll 1$ ならばGCM$[E]$はPRIV安全

証明可能安全な認証暗号への攻撃

- 正攻法: ブロック暗号Eを破る(難しいですよね...).

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E_K]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E_K}(\mathcal{A}') + \frac{\sigma^2}{2^n}$$

- 実際には別の”破れ方”が起きうる
  - 1. 誤用(misuse): 本来想定しない利用方法

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E_K]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E_K}(\mathcal{A}') + \frac{\sigma^2}{2^n}$$

  - 2. バウンドの第2項が~1になる攻撃(マッチングアタック)
    - 一般的に大量のデータを必要とする

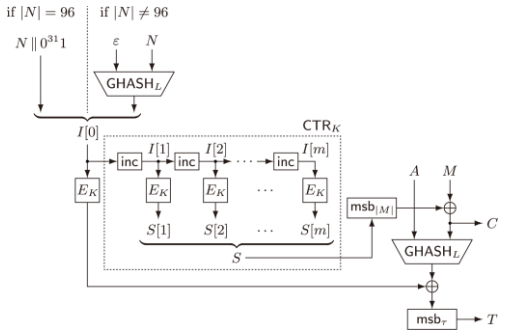
$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E_K]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E_K}(\mathcal{A}') + \frac{\sigma^2}{2^n}$$

  - 3. 証明の誤り. そもそも証明可能安全でなかった！

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{GCM}[E_K]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E_K}(\mathcal{A}') + \frac{\sigma^2}{2^n}$$

- いずれもEに依存しない攻撃. EがAESでも関係ない

誤用による攻撃



- 暗号化でのNonce重複による攻撃 “nonce-misuse attack”
- GCMでは致命的（Joux, 2006）
  - PRIVへの攻撃: two-time pad. 平文差分が漏洩する
  - AUTHへの攻撃: 内部のGHASH処理の鍵 L が漏洩する
    - $T_1 = \text{GHASH}_L(A_1, C_1) \oplus E_K(N_1 \| \| 0^{32}), T_2 = \text{GHASH}_L(A_2, C_2) \oplus E_K(N_2 \| \| 0^{32})$
    - $N_1 = N_2 \rightarrow T_1 \oplus T_2 = \text{GHASH}_L(A_1, C_1) \oplus \text{GHASH}_L(A_2, C_2) = \text{GHASH}_L(A_1 \oplus A_2, C_1 \oplus C_2)$.
    - Lを知ると任意の(N, A, C)に対する偽造が作れるようになる.

- 実プロトコルでのGCM Nonce重複
  - TLS通信（Böck et al. WOOT 2016）
  - WPA2 “Krack”（Vanhoef and Piessens CCS 2017）


Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS

Hanno Böck[1], Aaron Zauner[2], Sean Devlin[3], Juraj Somorovsky[4] and Philipp Jovanovic[5]

[1]https://hboeck.de, hanno@hboeck.de
[2]SBA Research gGmbH, azauner@sba-research.org, lambda: resilient.systems, azet@azet.org
[3]Independent, seanpatrickdevlin@gmail.com
[4]Horst Görtz Institute for IT Security, Ruhr University Bochum, juraj.somorovsky@rub.de
[5]École Polytechnique Fédérale de Lausanne (EPFL), philipp.jovanovic@epfl.ch


Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2

Mathy Vanhoef
imec-DistriNet, KU Leuven
Mathy.Vanhoef@cs.kuleuven.be

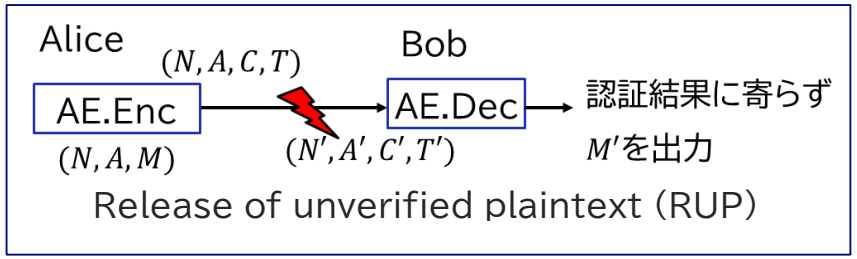
Frank Piessens
imec-DistriNet, KU Leuven
Frank.Piessens@cs.kuleuven.be

https://www.usenix.org/system/files/conference/woot16/woot16-paper-bock.pdf
https://papers.mathyvanhoef.com/ccs2017.pdf

その他の誤用

◆ 復号検証失敗時の情報漏洩(戻り値が複数存在)

- 未検証平文の漏洩(RUP), パディングオラクルなど
- 改ざん失敗しても何等か情報を得られる. これを攻撃に利用



◆ 暗号文と誤った鍵との紐付け(key committing) ホットトピック

- 所与の(暗号文,タグ)ペアを作れる鍵Kは正規の一つだけ… という誤解
  - GCMなどでは容易に偽の鍵K'を作れる
- 例: Facebookのメッセージングアプリにおける誤用(Grubbs et al. Crypto 2017)
  - ハラスメントなど問題のあるメッセージの通報機能(Msg franking)に問題. 嘘の通報が可能に

$$(N, A, M) \xrightarrow{K} (C, T)$$

$$M' \xleftarrow{K'} (N', A', C, T)$$

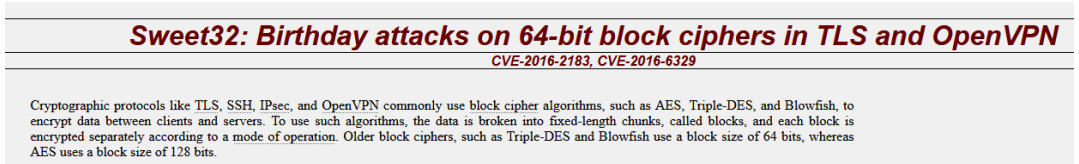
◆ 誤用はありえるものとして認証暗号の設計に反映されつつある

- 常に必要とは限らない.基本はアプリ依存でリスク判断する.
- 誤用耐性を持つ認証暗号は計算量が増える. このコスト増も考慮すべき

マッチングアタック

- $\sigma^2/2^n$ は安全性バウンドの典型的な項(σ: 一つの鍵での暗号化データ量, n: ブロックサイズ)
  - 内部変数の衝突確率に関するバースデーパラドックスから導出.
  - $\sigma = 2^{\frac{n}{2}}$ 程度クエリしたときに何が起きるか
    - カウンターモードのBreak(同じ平文を送り続けていれば暗号文ブロックは衝突しない. 真の乱数なら衝突が期待される)
    - CBC-MACなどでの出力衝突→さらなる偽造作成

- 実プロトコル例: Sweet32 (Bhargavan and Leurent, CCS 2016)
  - TLS, VPNのAEにおける64-bitブロック暗号(TDESなど)の利用: 2^{32}ブロック≒32Gb の処理でバウンドが1になる
    - 適切に鍵更新していれば回避できるが、現実は異なる...
  - モードで対処することも可能
    - $\sigma = 2^{\frac{n}{2}}$ でもバウンドが 1 より十分小さい方式. Beyond-birthday-bound (BBB) AE


Sweet32: Birthday attacks on 64-bit block ciphers in TLS and OpenVPN

CVE-2016-2183, CVE-2016-6329

Cryptographic protocols like TLS, SSH, IPsec, and OpenVPN commonly use block cipher algorithms, such as AES, Triple-DES, and Blowfish, to encrypt data between clients and servers. To use such algorithms, the data is broken into fixed-length chunks, called blocks, and each block is encrypted separately according to a mode of operation. Older block ciphers, such as Triple-DES and Blowfish use a block size of 64 bits, whereas AES uses a block size of 128 bits.

https://sweet32.info/

証明の誤り

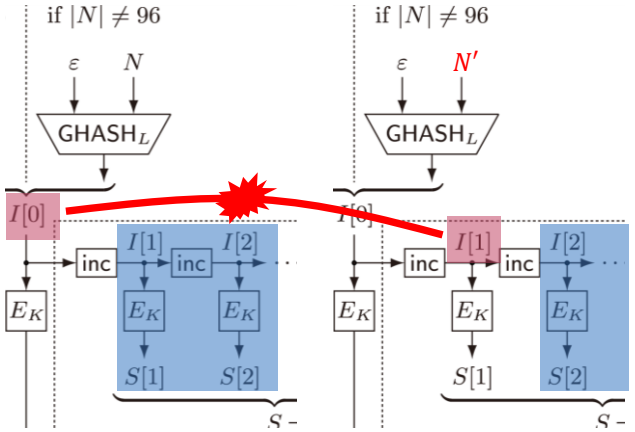
- 安全性証明＝ある離散確率の上界計算
  - Game-playing, H-Coefficient, Chi-Square法など固有の計算技法が発展
  - 先端的な方式の証明は複雑になることが多い
    - 複雑な場合分け、数え上げなど
    - 学術論文で証明付きで提案されたのち、誤りが見つかるケースがある

- 証明の誤りの影響は様々
  - 最悪ケースはアタックにつながる

（次スライドからいくつかのケースを見ていきます）

例1: GCM（Iwata et al. Crypto 2012）

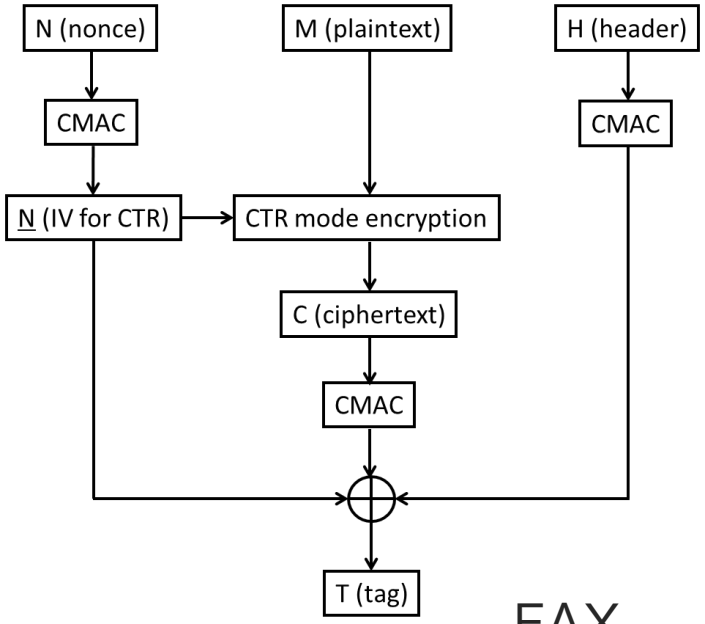
- NIST推奨方式GCMの設計者による証明に誤りを発見
- 非96-bitナンスでのカウンター値の衝突確率評価の誤り
  - $\Pr[\mathrm{GHASH}_L(N) = \mathrm{inc}_r(\mathrm{GHASH}_L(N'))]$, $r = 1,2,\ldots$
    - incは下32-bitだけの算術加算. GHASHは128-bit(体)演算. GHASHの既知の性質だけでは衝突確率は計算不可能
- 新たな安全性バウンドを提示: **攻撃成功確率2^{22}倍**
  - 確率の上界(最悪値)なので実際にこの確率で攻撃できるとは限らない
- 実際に後続研究（Niwa et al. FSE 2015）により、攻撃成功確率はオリジナル証明とほぼ同じレベル(2^5倍)に改善された



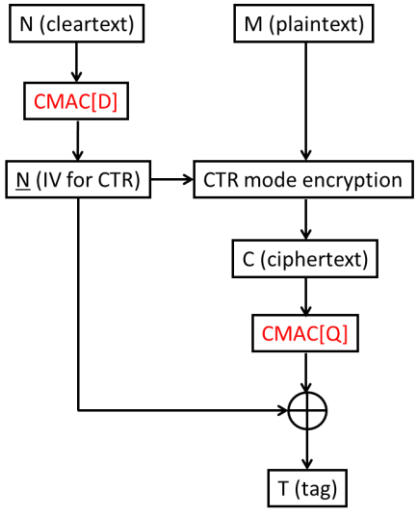
- モード自体は変えず，十分な安全性を回復できた

例2: EAX-prime

- ANSI C12.22. スマートグリッド向けプロトコル
- EAXは安全性証明済み（Bellare et al. FSE 2004）
- EAX-primeはEAXのアルゴリズムを簡素化
  - AESを使うところは変えず. 簡素化によりメモリ量削減などが実現（組み込み用途を考慮）
- ANSIがNISTへ標準化を提案（2011）



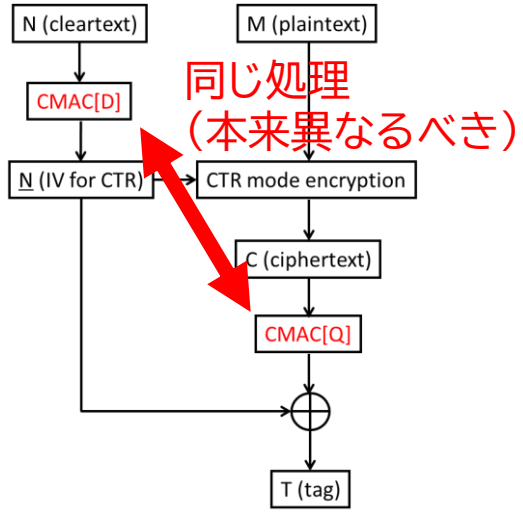
EAX



EAX-prime

EAX-primeの問題点(Minematsu et al. FSE 2013)

- EAX, EAX-primeともCMAC (NIST推奨MACの一つ. CBC-MACがベース)をわずかに改変して利用
  - MAC部分、(長い)Nonceの圧縮など
- EAX-primeのCMAC改変方法に問題があった
  - 2つのCMACバリエーションを作りたいのに、1ブロック平文については同一の処理となる. 1ブロック入力での改ざんが確率1で可能
- 論文を受けてNISTは標準化検討をストップ



EAX-prime

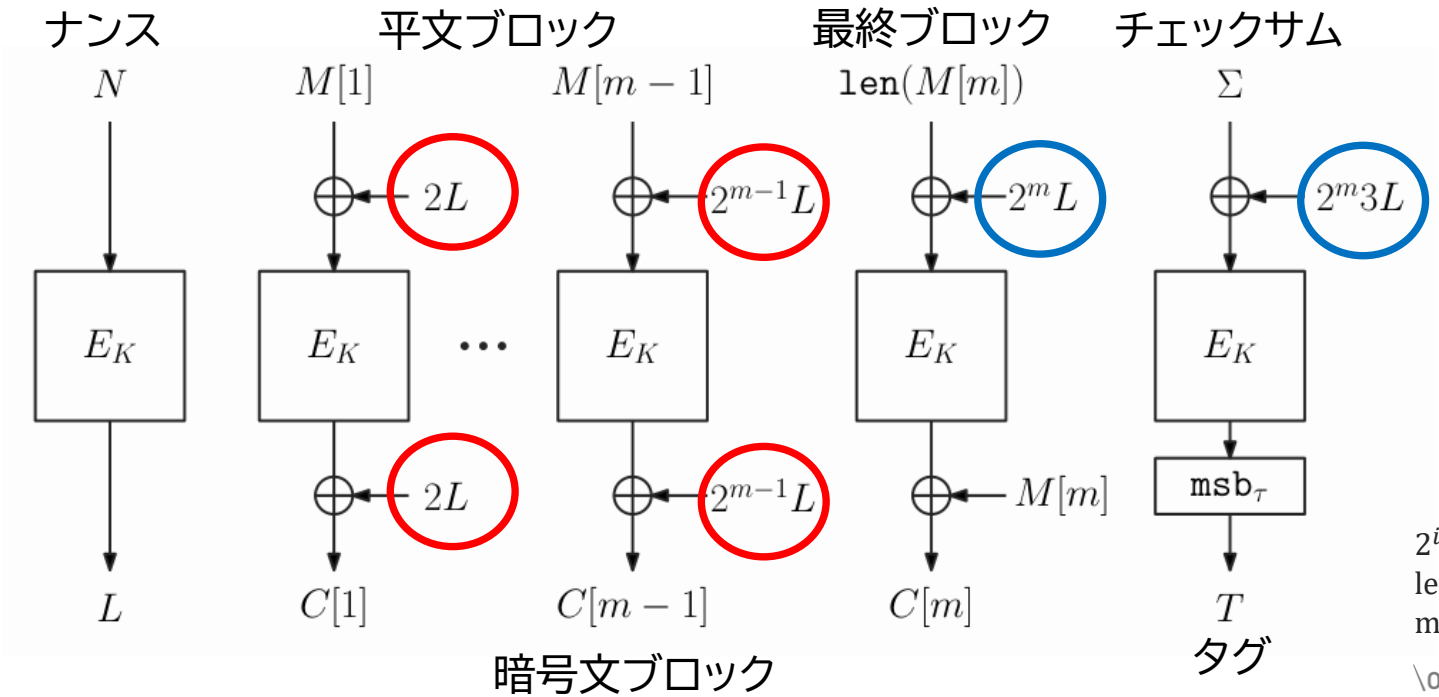
- 一方、2ブロック以上の入力を保証するプロトコルならば安全という証明がつく(Minematsu et al. FSE 2013)
- C12.22はこのケースに該当する模様

例3: OCB2（Rogaway Asiacrypt 2004）

- 著名な認証暗号方式. 提案論文はモードの設計・証明技術に多大な影響を与えた
- 計算効率がよい
  - 汎用結合とは本質的に異なるアプローチ: 1ブロック処理に1ブロック暗号コール
    - 汎用結合方式では1ブロック処理に2コール
  - ブロックごとのパラレル処理が可能
  - 証明可能安全
- ISO標準. 複数のOSSで利用（Javascript暗号ライブラリ SJCL等）
- 複数の派生方式が存在
  - OCB3: RFC, ISO など広く標準化
  - XTS: ストレージ暗号化のIEEE標準. デファクト
  - XEX: PCのメインメモリ暗号化（AMD SEV, Intel TDX）

OCB2の構造(AD省略)

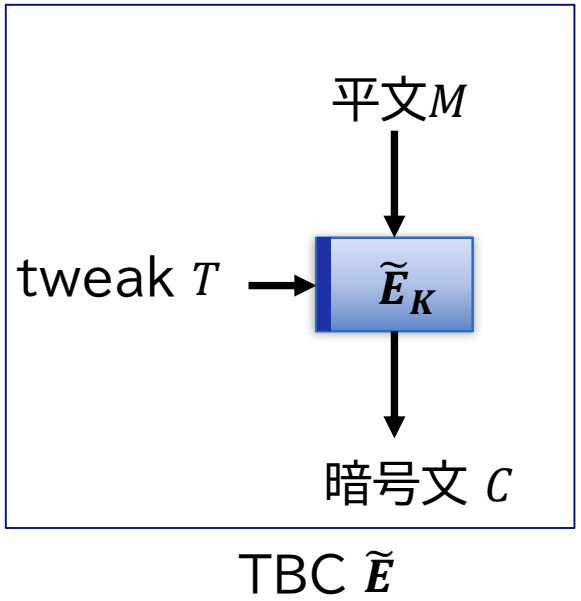
- 暗号化: ECBモードに類似するがブロック暗号Eの上下にナンスと鍵依存のマスクを適用
- 最終平文ブロック暗号化はCTRモードのような処理
- 平文チェックサム $\Sigma = M[1] \oplus M[2] \oplus \cdots$ を暗号化しタグとする.(独立したMACはなし!)
  - 最終ブロックとチェックサムの暗号化処理には上だけマスク

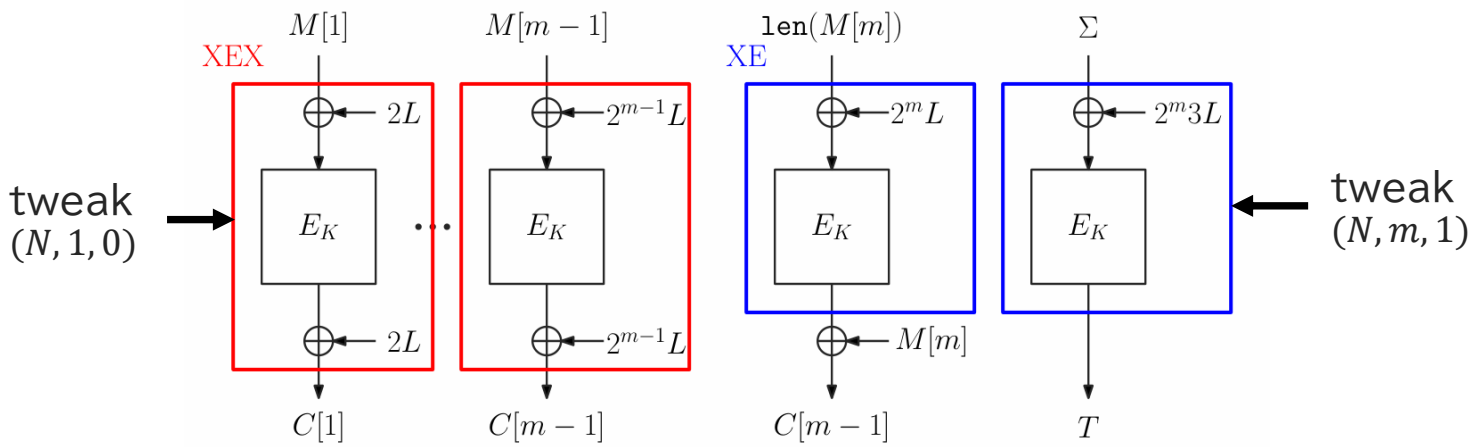


$2^i 3^j L$: 有限体$GF(2^n)$上の定数乗算
$\mathrm{len}(x)$: xの長さのn-bit表現
$\mathrm{msb}_\tau(x)$: xの上位 τ bits

オリジナル論文の安全性証明

- 証明ステップ
  - 1.(ブロック暗号+I/O マスク)をTweakableブロック暗号(TBC) XEX*で抽象化
    - TBC: 暗号化の際にTweakと呼ばれるパラメータを付与(tweakが異なれば同じ平文も異なる暗号文になる. 鍵を変えるのと同様の効果)
  - 2. XEX*のTBC安全性を証明
  - 3. TBCベースのOCBの安全性を証明
  - 2と3の結果を組み合わせると全体の安全性証明となる

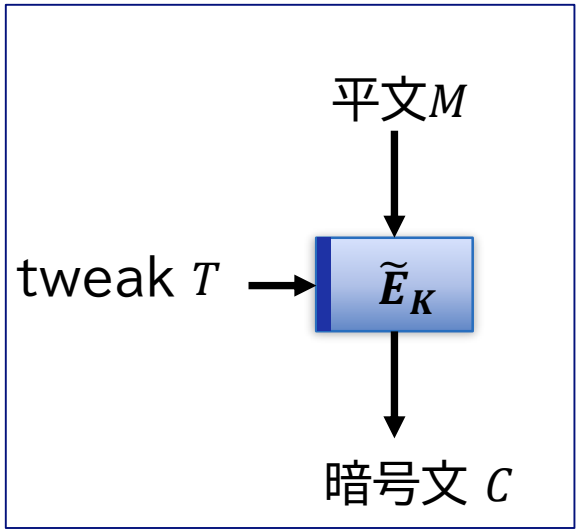




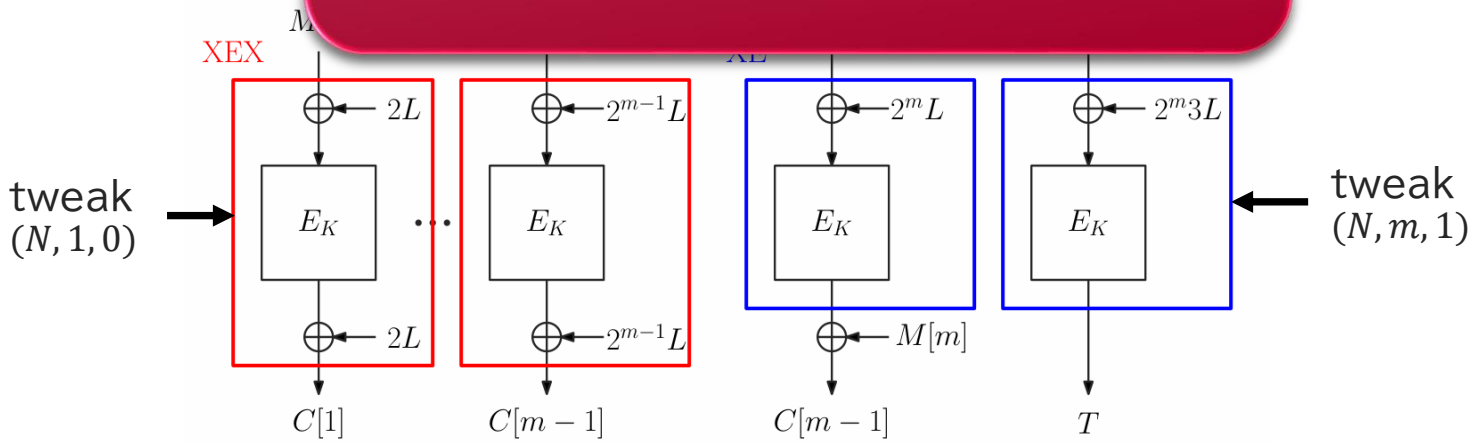
XEX:上下にマスク　XE:上にマスク　XEX* = XEX+XE

オリジナル論文の安全性証明

- 証明ステップ
  - 1.(ブロック暗号+I/O マスク)をTweakableブロック暗号(TBC) XEX*で抽象化
    - TBC: 暗号化の際にTweakと呼ばれるパラメータを付与(tweakが異なれば同じ平文も異なる暗号文になる. 鍵を変えるのと同様の効果)
  - 2. XEX*のTBC安全性を
  - 3. TBCベースのOCBの
  - 2と3の結果を組み合わせ



モジュラーでエレガントな証明技法.
しかし問題が...



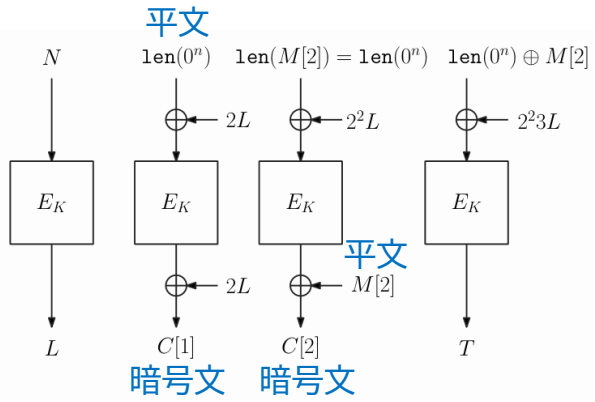
XEX:上下にマスク　XE:上にマスク　XEX* = XEX+XE

OCB2への攻撃 (Inoue et al. Crypto 2019)(15年後!)

- 最小ケースでは2クエリで改ざんが可能
  - ある2ブロック平文$M = (M[1] = \mathrm{len}(0^n), M[2])$を適当なナンス$N$と共に暗号化クエリする. 得られた$C = (C[1], C[2])$ から偽の1ブロック暗号文C'と、それに対応した正しいタグT'の両方が計算できてしまう. 確率ほぼ1で改ざん成功
- ここから派生して平文回復などのより強力な攻撃も実行可能

最小ケースの改ざん攻撃

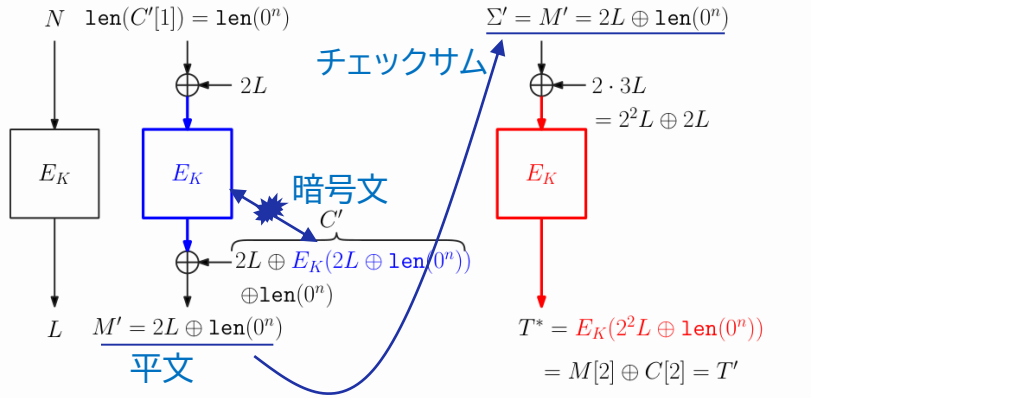
暗号化クエリ



$$C[1] = 2L \oplus E_K(2L \oplus \mathtt{len}(0^n))$$
$$C[2] = M[2] \oplus E_K(2^2L \oplus \mathtt{len}(0^n))$$

$(N' = N,\ C' = C[1] \oplus \mathrm{len}(0^n),\ T' = C[2] \oplus M[2])$

復号クエリ (改ざん)

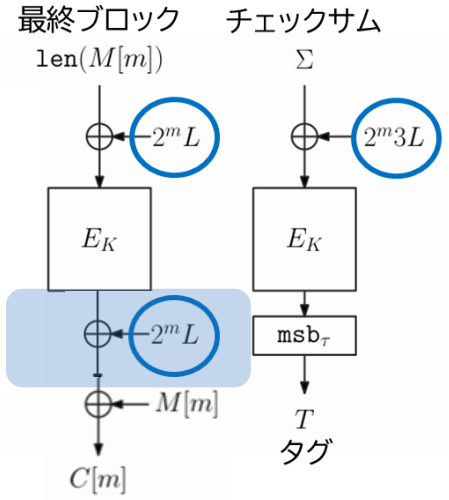


$$C[1] = 2L \oplus E_K(2L \oplus \mathtt{len}(0^n))$$
$$M[2] \oplus C[2] = E_K(2^2L \oplus \mathtt{len}(0^n))$$

攻撃の原因

- 問題の根は証明ステップ 1,2
  - ステップ1と2でTBCアクセス要件(tweakごとに定めたクエリのルール)に齟齬
  - 今の攻撃はステップ2の証明から逸脱: 本来許されるクエリなのにステップ2の証明では禁じられた(証明が考慮しない)クエリになっている
- 修正は容易: 平文最終ブロック暗号化において出力マスクを追加するのみ. 証明可能安全性を得る
- 本攻撃を受けてOCB2はISOから除外
- OCB3は該当処理が異なるため影響なし(安全)

- 1.(ブロック暗号+I/O マスク)をTweakableブロック暗号(TBC) XEX*で抽象化
- 2. XEX*のTBC安全性を証明
- 3. TBCベースのOCBを安全性証明
- 2と3の結果を足すと全体の安全性証明となる



まとめ

- 認証暗号はメッセージの秘匿と真正性を同時に担保する暗号化
  - 秘匿のみの暗号化は改ざんに気づけない
  - 将来はあらゆるものが通信する世界(IoT). 改ざんリスクを物理的に排除するのは困難. 認証暗号の適用をデフォルトと考えるべき
- 認証暗号は典型的には(秘匿のみ)暗号化とメッセージ認証コードの組み合わせで実現可能

- 認証暗号の安全性が内部の暗号コア(ブロック暗号)の安全性へ帰着されるとき、証明可能安全と呼ぶ
  - 内部のブロック暗号が破られない限り、認証暗号は安全
- “安全な認証暗号”とは、安全なブロック暗号を用いた証明可能安全な認証暗号といえる

まとめ(続)

- しかし… 現実には“安全な認証暗号”にも攻撃はありえる
  - 誤用: ナンスの重複利用など
  - マッチングアタック: 安全性証明が保証する以上の大量のデータを処理させる攻撃
    - 典型的には暗号化処理のランダムな内部変数の衝突を起こすことが目的.
  - 安全性証明自体の誤り

ユーザ/開発者としての留意点

- 誤用: プロトコルの中で想定外の利用方法をされていないか
- マッチングアタック: バウンドが示す安全性保証のなかで利用されているか
  - 一つの鍵で処理するデータの量や長さに注意
- 証明の誤り: 安全性証明がきちんと検証されているか. 関連学術論文のサーベイ
  - ユーザサイドでは限界はある. 例え著名な方式であろうと誤りは起こりえる. 専門家を頼ってほしい.

学術的には証明技法の機械化、形式検証の導入など技術革新も望まれる

\Orchestrating a brighter world

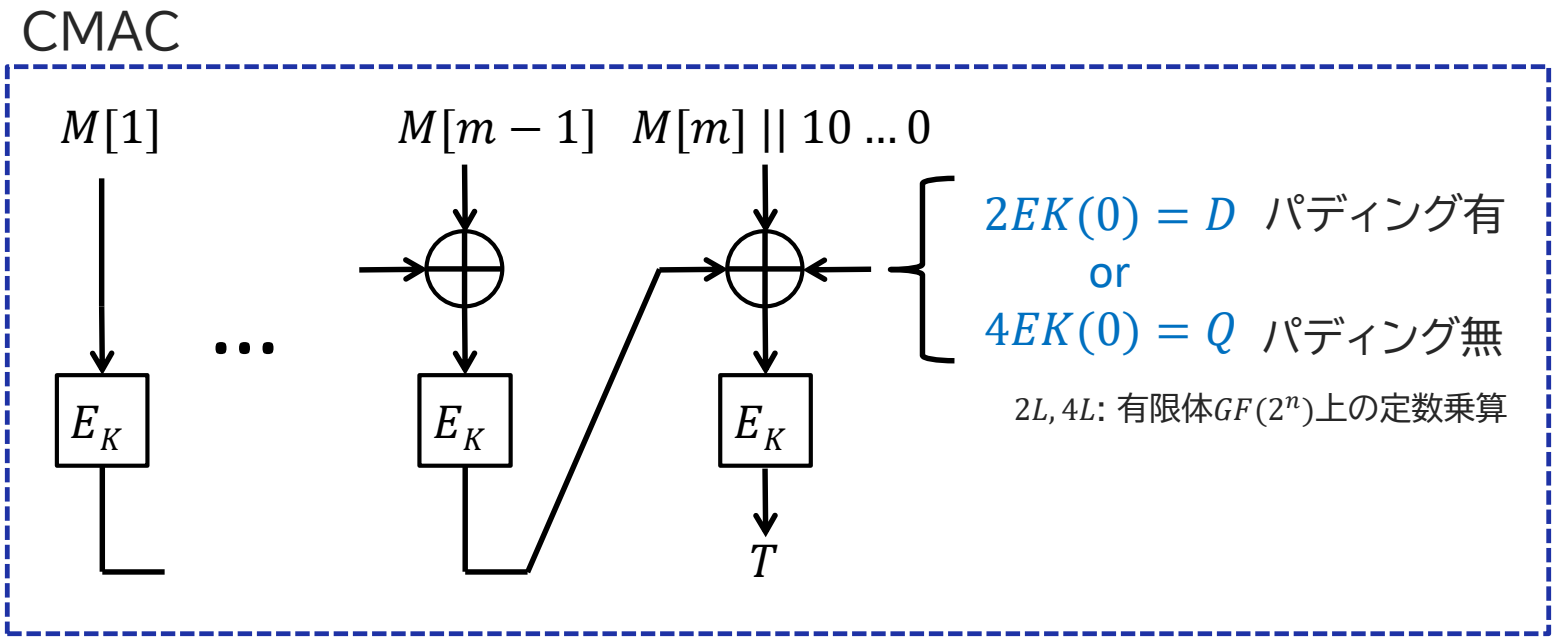
NECは、安全・安心・公平・効率という社会価値を創造し、
誰もが人間性を十分に発揮できる持続可能な社会の実現を目指します。


Orchestrating a brighter world
NEC

補遺

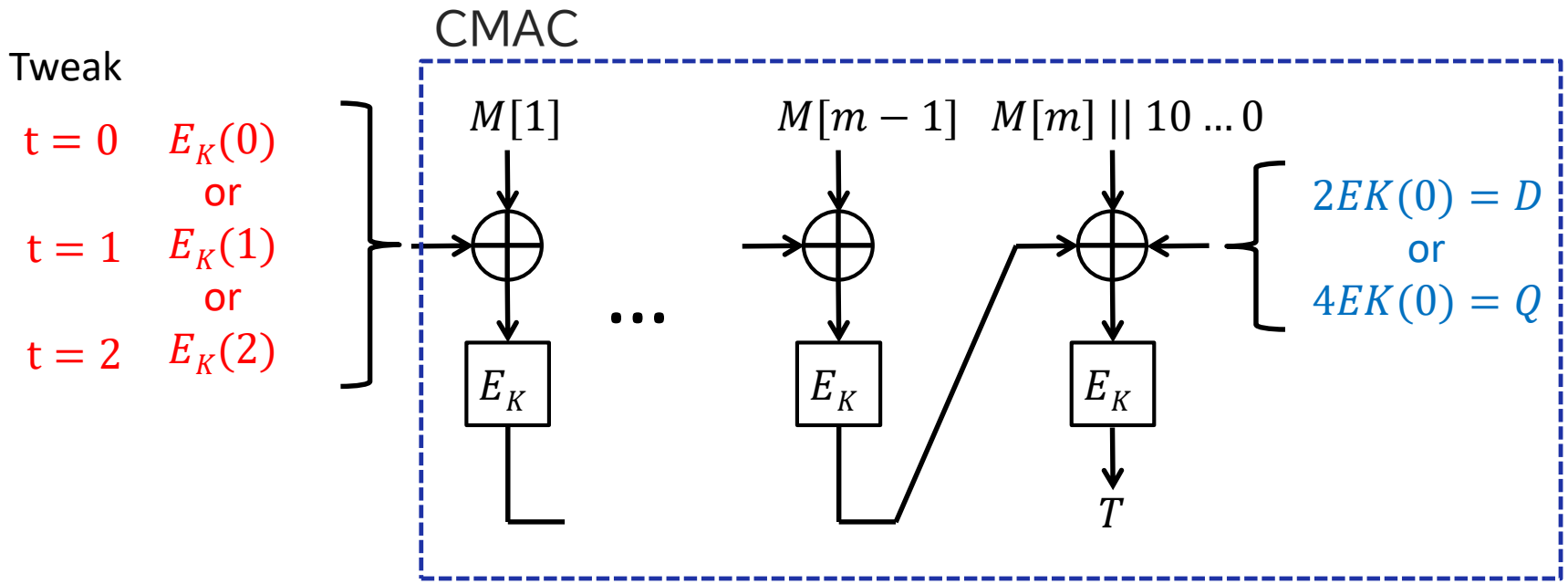
CMAC（NIST推奨MACの一つ）

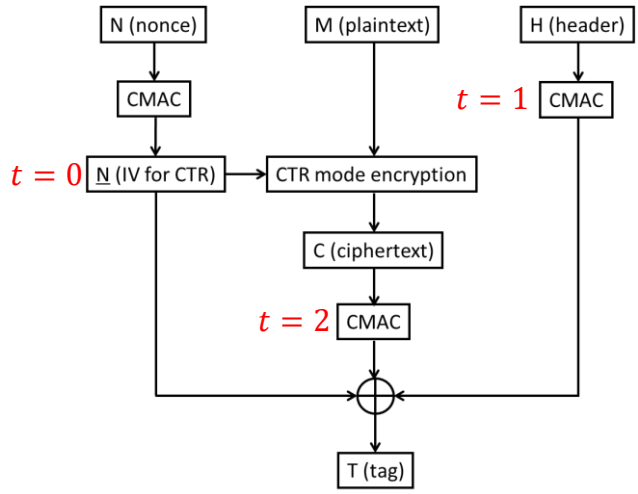
- CMACはCBC-MACの最後に鍵依存のマスク値をXORする
  - 最終ブロックにマスク. メッセージ長がnの倍数か否か（パディングの有無）で2通りのマスク値



EAXの中の改変版 CMAC

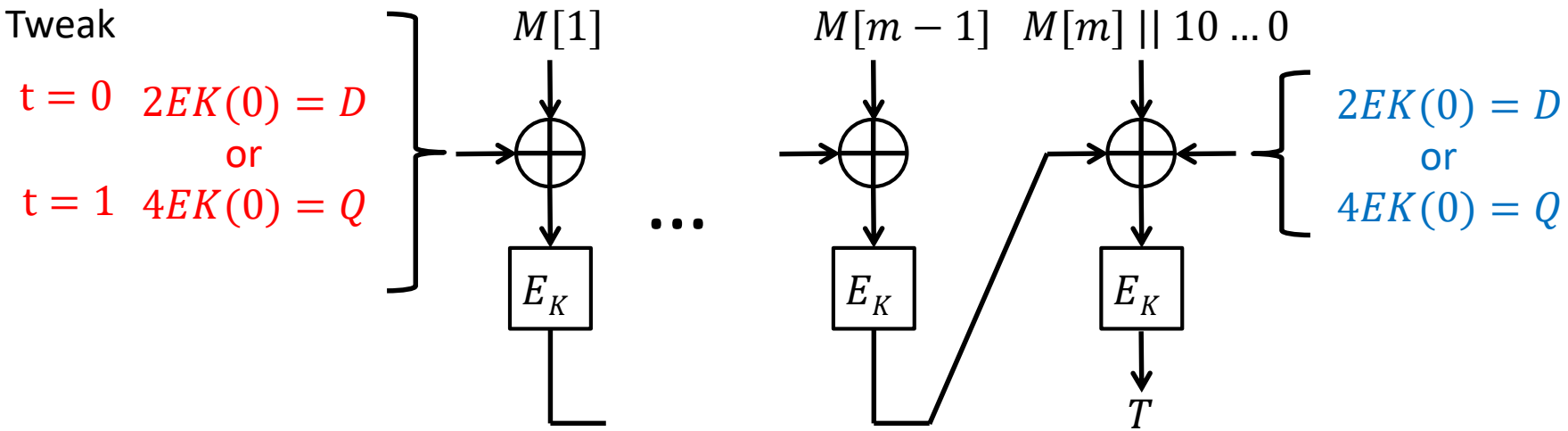
- EAXはCMAC（NIST標準のMAC）を僅かに改変して利用
- EAXでの改変版は最初にも3通りのマスク. tweakと呼ばれるパラメータで指定. Tweak $t \in \{0,1,2\}$.

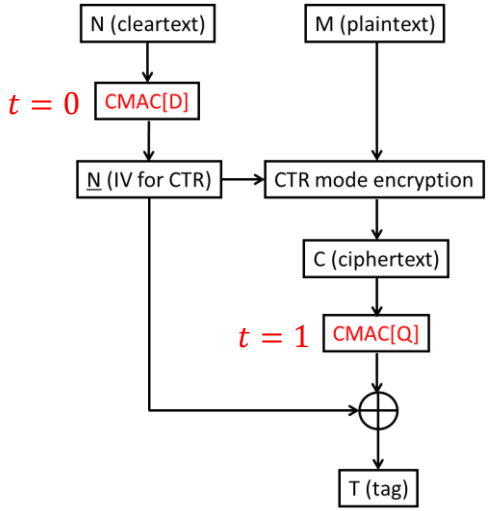




EAX-primeの中の改変版 CMAC

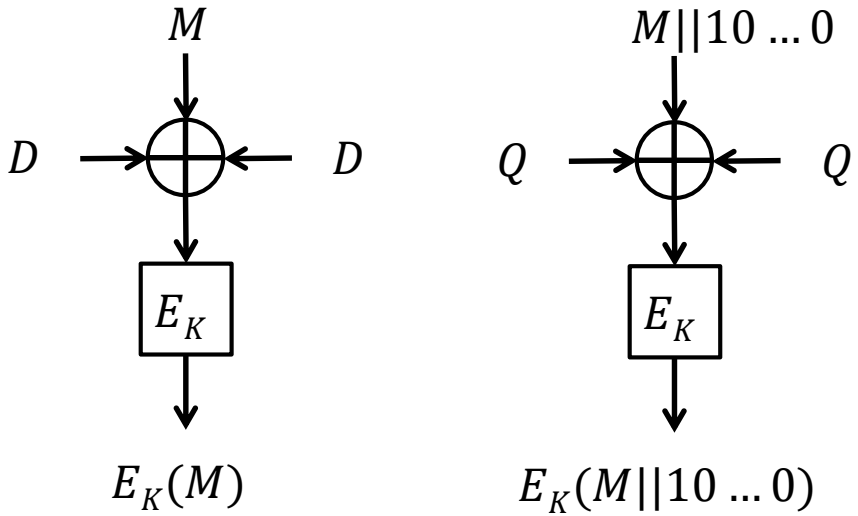
- EAX-primeもCMACを改変. ただしTweakは2通り, $t \in \{0,1\}$.
- 前後でマスクを使いまわす. 場合分けの数とメモリが減る.





EAX-primeへのアタック（Minematsu et al. FSE 2013）

- CMAC入力がn bit以下の場合，tweakとの組みでマスクのキャンセルが発生
- 1ブロック処理により生のE_K入出力が得られる．CBC-MAC系では大抵致命的な攻撃に繋がる．EAX-Primeの場合もこれを用いた改ざんが確率1で可能



- NISTは標準化検討をストップ
- 一方、CMAC入力> n bitsを保証するプロトコルならば安全という証明がつく
  (Minematsu et al. FSE 2013)
- C12.22はこのケースに該当する模様