ChaCha20-Poly1305 および Poly1305 の安全性調査・評価

岩田　哲

名古屋大学大学院工学研究科

2017 年 10 月

エグゼクティブサマリー

メッセージ認証コード Poly1305 [Ber05b] および認証暗号化方式 ChaCha20-Poly1305 [NL15] の安全性調査・評価を行った．

Poly1305 について，次の結果を得た．

- Bernstein による安全性証明 [Ber05b, Ber05a] に問題点は見受けられない．Poly1305 は証明可能安全性を有する MAC である．
- 関連鍵攻撃が可能である．しかし人工的な攻撃であり，現実的な脅威とはならないと考えられる．
- 複数ユーザ安全性について，ユーザ数への依存が限定的な安全性バウンドを導出できる．一般に，証明可能安全性の観点からはユーザ数が増加すると敵の攻撃確率の上界がそれに伴い増加するが，Poly1305 ではこの増加が限定的である．

また文献調査により，次のことを確認した．

- 弱鍵が存在する．
- 再偽造可能性の意味で耐性がある．
- ナンスを再利用すると多項式ハッシュ関数の鍵を導出でき，偽造攻撃ができる．

また，ChaCha20-Poly1305 について次の結果を得た．

- Procter による [Pro14] の証明には一か所軽微な誤りがあるが簡単に修正可能であり，ChaCha20-Poly1305 は証明可能安全性を有する認証暗号化方式である．
- ChaCha20 ブロック関数が関連鍵攻撃に対して安全であるという仮定のもと，ChaCha20-Poly1305 は関連鍵攻撃に対する安全性を有する．
- 再偽造可能性の意味で耐性がある．
- ナンスを再利用すると偽造攻撃ができる．
- 弱鍵が存在する．
- 復号ミスユース耐性について，暗号化の意味での安全性は失われるが，認証の意味での安全性は保たれる．
- 複数ユーザ安全性について，ユーザ数に依存した自明な安全性バウンドを導出できる．また，ユーザ数への依存が限定的な安全性バウンドを導出できると予想される．

Poly1305 と ChaCha20-Poly1305 両方について，ナンスの再利用による安全性への影響は大きく，実装の際にはこれが起こらないよう注意が必要である．弱鍵の影響はどちらも少ないと考えられ，また Poly1305 の関連鍵攻撃が現実的に問題となることはないと考えられる．その他，懸念事項は見受けられない．

なお本報告書は Poly1305 と ChaCha20-Poly1305 のメッセージ認証コードと認証暗号化方式としての理論的な安全性調査・評価を行ったものであり，サイドチャネル攻撃等の実装に関する解析，あるい TLS のプロトコルとしての安全性解析等は扱っていない．

目 次

図 目 次

表 目 次

1　はじめに

メッセージ認証コード Poly1305 は Bernstein によって FSE 2005 にて提案された [Ber05b]．また，認証暗号化方式 ChaCha20-Poly1305 は Nir と Langley により RFC 7539 にて提案され [NL15]，TLS にて利用することが提案されている [LCM+16]．本報告書ではこれらのアルゴリズムの安全性調査・評価を報告する．

本書では下記項目を記載する．

- エグゼクティブサマリー
- Poly1305 の仕様（3 章）
- Poly1305 の安全性評価に関する文献調査（4 章）
- Poly1305 に対する新たな安全性評価の検討（5 章）
- ChaCha20-Poly1305 の仕様（6 章）
- ChaCha20-Poly1305 の安全性評価に関する文献調査（7 章）
- ChaCha20-Poly1305 に対する安全性評価の検討（8 章）

安全性評価を行うにあたり，一般的な暗号利用条件を想定したが，特に TLS での利用が想定されることから，複数ユーザの設定における安全性の検討を行った．予備的な検討段階であるが，敵の攻撃成功確率の上界を導出できるという点で，Poly1305 と ChaCha20-Poly1305 はいずれも複数ユーザの設定における証明可能安全性を有している．

2　準備

整数 $\ell \geq 0$ に対し，$\{0,1\}^\ell$ はすべての ℓ ビットのビット列の集合を表す．また，$\{0,1\}^*$ はすべての有限ビット列の集合を表す．バイトは $\{0,1,\ldots,255\}$ の要素を指し，8 ビットのビット列に対応するとする．バイト列 X のバイト長を $\mathsf{len}(X)$ と表記する．バイト列 X と Y の連結を $X \| Y$ あるいは XY と表記する．整数 x と ℓ に対し，バイト列 X を ℓ バイトごとに分割する操作を $(X[0],\ldots,X[x-1]) \xleftarrow{\ell} X$ と表記する．ただし $X[0],\ldots,X[x-2]$ は ℓ バイト，$1 \leq \mathsf{len}(X[x-1]) \leq \ell$ であり，$X[0],\ldots,X[x-1]$ の連結 $X[0] \| \cdots \| X[x-1]$ が X と一致する．また，空列の分割はこの限りではなく，X が空列の場合は $X[0] \xleftarrow{\ell} X$ とし，$X[0]$ は空列であるとする．有限集合 $\mathcal{S}$ から一様ランダムに要素を選択し，それを S とすることを $S \xleftarrow{\$} \mathcal{S}$ と表記する．16 進数は 0x をつけて 0x03 などと表記する．

3　Poly1305 の仕様

本章ではメッセージ認証コード Poly1305 の仕様を記述する．ブロック暗号として AES-128 を利用し，Poly1305-AES と表記する．Poly1305-AES はタグ生成関数 Poly とタグ検証関数 Ver からなる．タグ生成に素数 $2^{130}-5$ を法とした多項式演算を用いる．タグ生成関数 Poly の入出力は

$$\mathsf{Poly}\colon \mathcal{K}_{\mathsf{Poly}} \times \{0,1\}^{128} \times \{0,1\}^{*} \to \{0,1\}^{128}$$

として表される．ただし，$\mathcal{K}_{\mathsf{Poly}} = \{0,1\}^{128} \times \{0,1\}^{128}$ は鍵空間である．秘密鍵 $(R,K) \in \mathcal{K}_{\mathsf{Poly}}$，ナンス $N \in \{0,1\}^{128}$，任意長のメッセージ M を入力としてとり，128 ビット（16 バイト）のタグ $T \in \{0,1\}^{128}$ を出力する．$T \leftarrow \mathsf{Poly}_{R,K}(N,M)$ と表記する．

タグ検証関数 Ver の入出力は

$$\mathsf{Ver}\colon \mathcal{K}_{\mathsf{Poly}} \times \{0,1\}^{128} \times \{0,1\}^{*} \times \{0,1\}^{128} \to \{\top,\bot\}$$

である．秘密鍵 $(R,K) \in \mathcal{K}_{\mathsf{Poly}}$，ナンス $N \in \{0,1\}^{128}$，任意長のメッセージ M，タグ $T^{*} \in \{0,1\}^{128}$ を入力としてとり，$T \leftarrow \mathsf{Poly}_{R,K}(N,M)$ を計算し，$T = T^{*}$ であれば受理を示す $\top$ を返し，$T \neq T^{*}$ であれば拒否を示す $\bot$ を返す．$\top \leftarrow \mathsf{Ver}_{R,K}(N,M,T^{*})$ や $\bot \leftarrow \mathsf{Ver}_{R,K}(N,M,T^{*})$ と表記する．

Poly1305-AES $= (\mathsf{Poly},\mathsf{Ver})$ の擬似コードを図 1，図 2 に示す．

鍵　K は AES-128 の秘密鍵であり，R は法を $2^{130}-5$ とした多項式演算で用いる．$R \in \{0,1\}^{128}$ であり，次の制約を満たす．

- 下位バイトから数えて 4 バイト目，8 バイト目，12 バイト目，16 バイト目の上位 4 ビットの値を 0 に固定する．
- 下位バイトから数えて 5 バイト目，9 バイト目，13 バイト目の下位 2 ビットの値を 0 に固定する．

言い換えると，バイト列 $(R[0],\ldots,R[15])$ を用いて

$$R = R[0] + 2^{8}R[1] + \cdots + 2^{120}R[15]$$

として表現するとき，$R[3],R[7],R[11],R[15]$ の上位 4 ビットが 0 であり，したがってこれらは $\{0,1,\ldots,15\}$ の値しかとらない．また，$R[1],R[8],R[12]$ の下位 2 ビットが 0 であり，これらは $\{0,4,8,\ldots,252\}$ の値しかとらない．R は 16 バイトの長さであるが，取りうる値は 2^{106} 通りになる．これにより，法 $2^{130}-5$ の多項式演算における計算コストを削減することができる．

Algorithm $\mathsf{Poly}_{R,K}(N, M)$

1. $(M[0], \ldots, M[m-1]) \xleftarrow{16} M$ // $m = \lceil \mathsf{len}(M)/16 \rceil$
2. **for** $i = 1$ **to** $m - 1$ **do**
3. $\quad C[i] \leftarrow M[i] + \mathtt{0x01} \cdot 2^{8\mathsf{len}(M[i])}$
4. $T \leftarrow C[0]$
5. **for** $i = 1$ **to** $m - 1$ **do**
6. $\quad T \leftarrow R \cdot C[i] \bmod (2^{130} - 5)$
7. $T \leftarrow R \cdot T \bmod (2^{130} - 1)$
8. $T \leftarrow T + \mathrm{AES}_K(N) \bmod 2^{128}$
9. **return** T

図 1: $T \leftarrow \mathsf{Poly}_{R,K}(N, M)$ の定義

Algorithm $\mathsf{Ver}_{R,K}(N, M, T^*)$

1. $T \leftarrow \mathsf{Poly}_{R,K}(N, M)$
2. **if** $T = T^*$ **then return** $\top$
3. **else return** $\bot$

図 2: $\top/\bot \leftarrow \mathsf{Ver}_{R,K}(N, M, T^*)$ の定義

メッセージ　メッセージ M を $(M[0], \ldots, M[m-1]) \xleftarrow{16} M$ と 16 バイトごとに分割する．ただし，$m = \lceil \mathsf{len}(M)/16 \rceil$ である（$\mathsf{len}(M)$ は M のバイト長である）．m は M のブロック長を表している．各メッセージブロック $M[i]$（$i = 0, \ldots, m-1$）に，それぞれ最上位にバイト 0x01 を付加し，$C[i]$ とする．すなわち，

$$C[i] = \begin{cases} M[i] + \mathtt{0x01} \cdot 2^{128} & \text{for } i = 0, \ldots, m-2 \\ M[i] + \mathtt{0x01} \cdot 2^{8\mathsf{len}(M[i])} & \text{for } i = m-1 \end{cases}$$

とする．$C[1], \ldots, C[m-2]$ は 17 バイトであり，各 $C[i]$ は $M[i]$ よりも 1 バイト長くなる．

多項式ハッシュ関数　R と $C[1], \ldots, C[m-1]$ を用いて，ハッシュ値 $H_R(M)$ を次のように計算する．

$$H_R(M) = C[0] \cdot R^m + C[1] \cdot R^{m-1} + \cdots + C[m-1] \cdot R \bmod (2^{130} - 5)$$

ただし，R と各 $C[i]$ をリトルエンディアンで表現された符号なし整数とみなして演算を行う．

タグ生成　ハッシュ値 $H_R(M)$ と，鍵 K，ナンス N から次のようにタグ $T \leftarrow \mathsf{Poly}_{R,K}(N, M)$ を生成する．

$$T = H_R(M) + S \bmod 2^{128}$$

だたし $S = \mathrm{AES}_K(N)$ である．最後に T を出力する．

4　Poly1305 に関する文献調査

本章では Poly1305 に関する文献調査の報告をする．4.1 章で提案論文における証明可能安全性を扱い，安全性解析について，関連鍵攻撃，再偽造可能性，ナンス再利用，弱鍵についてそれぞれ 4.2.1 章，4.2.2 章，4.2.3 章，4.2.4 章で扱う．また，改良版，修正版を 4.3 章で扱う．

4.1　証明可能安全性

Poly1305 の証明可能安全性は提案論文 [Ber05b] にて示されている．Poly1305 はナンスを用いる MAC であり，敵はタグ生成オラクルとタグ検証オラクルにアクセスできる．

- タグ生成オラクルは鍵 (R, K) を保持しており，入力 (N, M) に対し，タグ $T \leftarrow \mathsf{Poly}_{R,K}(N, M)$ を返す．
- タグ検証オラクルも同じ鍵 (R, K) を保持しており，入力 (N, M, T^*) に対し，$T \leftarrow \mathsf{Poly}_{R,K}(N, M)$ を計算し，$T = T^*$ であれば受理を示す $\top$ を返し，$T \neq T^*$ であれば拒否を示す $\perp$ を返す．$\top$ は偽造成功に対応し，$\perp$ は偽造失敗に対応する．

敵 $\mathcal{A}$ の目標は一度でも検証オラクルが $\top$ を返すことである．ただし，$\mathcal{A}$ はタグ生成オラクルに対して同じナンスを繰り返し使用してはならず，また鍵 (R, K) は適切な空間から一様ランダムに選択されているとする．また一般性を失うことなく，タグ生成オラクルがクエリ (N, M) に対して T を返したならば，それ以降 (N, M, T) を検証オラクルにクエリせず，さらに，クエリは繰り返さないものとする．

次の定理が示されている．

定理 1 ([Ber05b]) 高々 q 回タグ生成オラクルにアクセスし，高々 q' 回タグ検証オラクルにアクセスする敵 $\mathcal{A}$ を考える．ただし，各クエリの M の長さ

は高々ℓバイトであるとする．また$q+q'$回のクエリをする任意の敵$\mathcal{B}$に対し，$\mathrm{AES}_K(\cdot)$と128ビット上のランダム置換$P(\cdot)$の識別成功確率が高々δであると仮定する．このとき，タグ検証オラクルが$\mathcal{A}$の高々q'回のクエリをすべての拒否し，すべてに対して$\perp$を返す確率は少なくとも

$$1-\delta-\frac{(1-q/2^{128})^{-(q+1)/2}\times 8q'\lceil \ell/16\rceil}{2^{106}} \tag{1}$$

である．

式(1)は$q\le 2^{64}$のとき，敵が一度でも偽造攻撃に成功する確率が高々

$$\delta+\frac{1.649\cdot 8q'\lceil \ell/16\rceil}{2^{106}}<\delta+\frac{14q'\lceil \ell/16\rceil}{2^{106}} \tag{2}$$

であることを示している．式(2)の右辺がqに依存しないことは注目に値する．

定理1は次の補題を用いて示される．

補題 1 ([Ber05b]) MとM'を，長さが高々ℓバイトである互いに異なるメッセージとする．Zを任意の16バイト列とする．このとき，$H_R(M)=H_R(M')+Z$を満たす$R\in\{0,1,\ldots,2^{130}-6\}$は高々$8\lceil \ell/16\rceil$通りである．ただし，加算は法$2^{128}$上で定義される．

補題1はRが要素数2^{106}の空間から一様ランダムに選択されるとき，

$$\Pr_{R\xleftarrow{\$}\mathcal{R}_{\mathsf{Poly}}}[H_R(M)-H_R(M')=Z]\le\frac{8\lceil \ell/16\rceil}{2^{106}}$$

ということを示しており（ただし，演算は$\bmod 2^{128}$で定義される），Poly1305で使用される多項式ハッシュ関数の差分確率の上界を与えている．ここで，$\mathcal{R}_{\mathsf{Poly}}$は$\mathsf{Poly}$の鍵空間$\mathcal{K}_{\mathsf{Poly}}$のうち，$R$の取りうる集合に対応する．

この性質はユニバーサルハッシュ関数として定式化されている[CW79]．一般に，$F:\mathcal{K}_F\times\mathcal{D}\to\mathcal{R}$を鍵付きハッシュ関数とする．$\mathcal{K}_F$は有限の鍵集合，$\mathcal{D}$は定義域，$\mathcal{R}$は値域である．$\mathcal{R}$が2項演算$+$について可換群をなすとし，その逆演算を$-$と表記する．このとき，任意の異なる$M,M'\in\mathcal{D}$と，任意の$Z\in\mathcal{R}$に対し，差分確率について

$$\Pr_{K\xleftarrow{\$}\mathcal{K}_F}[F_K(M)-F_R(M')=Z]\le\epsilon$$

が成り立つとき，Fはϵ-AΔU（ϵ-almost-Δ-ユニバーサル）ハッシュ関数という[Sti95]．とくに，$\mathcal{R}$が$\{0,1\}^n$であり演算として排他的論理和$\oplus$を考えるとき，Fをϵ-AXU（ϵ-almost-XOR-ユニバーサル）ハッシュ関数という[Kra94]．

差分確率の小さなハッシュ関数$H_R(\cdot)$とナンスを用いて$T=H_R(M)+\mathrm{AES}_K(N)$としてタグを生成する方法は，安全なMACの構成法として知られており[Sho96, Ber05a]，これと補題1を組み合わせることにより，定理1が得られる．

4.2 安全性解析

4.2.1 関連鍵攻撃

一般に，$F : \mathcal{K}_F \times \mathcal{D} \to \mathcal{R}$ を鍵付き関数とし，$\mathcal{K}_F$ を有限の鍵集合，$\mathcal{D}$ を定義域，$\mathcal{R}$ を値域とする．F に対する関連鍵攻撃では，関連鍵導出関数（RKD 関数，related-key derivation function）の集合 $\Phi = \{\varphi\}$ を考える．$\varphi : \mathcal{K}_F \to \mathcal{K}_F$ であり，敵はこの集合内の関数を用いて鍵に変更を加えることができると仮定する．例として，$\mathcal{K}_F = \{0,1\}^n$ のとき，

$$\Phi^{\oplus} = \{\mathrm{XOR}_{\Delta} \mid K \mapsto K \oplus \Delta, \Delta \in \mathcal{K}_F\}$$

あるいは

$$\Phi^{+} = \{\mathrm{Add}_{\Delta} \mid K \mapsto K + \Delta \bmod 2^n, \Delta \in \mathcal{K}_F\}$$

が挙げられる．敵は任意に選択した $\Delta \in \mathcal{K}_F$ に対して $F_{K\oplus\Delta}(\cdot)$ や $F_{K+\Delta}(\cdot)$ にアクセスできることを意味する．

ユニバーサルハッシュ関数に基づく MAC に対する関連鍵攻撃は Wang らにより解析されている [WLZZ16]．Poly1305 はこの枠組みの範疇である．

例として，$n = 128$ とし，鍵を $(R, K) \in \{0,1\}^{128} \times \{0,1\}^{128}$，2 ブロックのメッセージ $M = (M[0], M[1])$ に対し，

$$T = (M[0] \cdot R^2 \oplus M[1] \cdot R) \oplus \mathrm{AES}_K(N)$$

で定義される MAC を考える．ただし 1 ブロックは $n = 128$ ビットであり，演算は $\mathrm{GF}(2^n)$ で定義されるとする．$M[0] = M[1]$ とし，RKD 関数の集合として $\Phi^{\oplus}$ を考える．敵が関連鍵 $(R', K') = (R \oplus 0^{n-1}1, K)$ に対する (N, M) をタグ生成オラクルにクエリすると，オラクルは

$$\begin{aligned} T &= (M[0] \cdot (R \oplus 0^{n-1})^2 \oplus M[1] \cdot (R \oplus 0^{n-1})) \oplus \mathrm{AES}_K(N) \\ &= (M[0] \cdot R^2 \oplus M[1] \cdot R) \oplus \mathrm{AES}_K(N) \end{aligned}$$

を敵に返す．これは元の鍵 (R, K) に対する (N, M) のタグと一致し，偽造攻撃に成功する．

Wang らはこれと同様の攻撃が Poly1305 に対して適用可能であると述べている [WLZZ16] が，法 $2^{130} - 5$ 上で定義される多項式ハッシュ関数では上記にある $(R+1)^2 = R^2 + 1$ のような等式は成り立たず，この例によって Poly1305 が攻撃可能であるかどうかは定かではない．

一方で，5.2 章で示すように，入力メッセージを適切に選べば偽造攻撃が可能である．あるいはより複雑な RKD 関数の集合を許すことにより偽造攻撃が可能である．

一般に算術演算に基づく MAC では多くの場合関連鍵攻撃が可能である．一方，ブロック暗号のような要素技術のみに基づく MAC では，ブロック暗号

自体が関連鍵攻撃に対する安全性を有するという仮定により，方式全体の関連鍵攻撃に対する安全性を証明できる場合が多い．したがって，一般に算術演算に基づく MAC の方がブロック暗号のような要素技術のみに基づく MAC よりも関連鍵攻撃に対する耐性が低いと考えられる．

ただし，ここで取り上げたような関連鍵攻撃のシナリオは制約が強く，現実に成立する可能性は低いと考えられ，このことが実際に問題となるようなことはないと考えられる．しかし，関連鍵攻撃のシナリオが成立しないよう使用方法に注意すべきである．

4.2.2 再偽造可能性

MAC に対する再偽造不可能性は Black らにより提案された安全性概念である [BC09]．例として CBC MAC [BKR00] や PMAC [BR02] のような確定的な MAC を考える．仮に敵が偽造に成功，あるいは出力の衝突に成功したと仮定し，その後別の偽造に成功するのにどれだけの計算量が必要かにより安全性を評価する．理想的なランダム関数 F であれば，2 つのメッセージ M と M' の衝突 $F(M) = F(M')$ はそれら以外の衝突ペアを入手するのには役に立たない．しかし CBC MAC では衝突ペア M と M' からさらなる衝突ペアを得るのは容易である．

Poly1305 の再偽造可能性は [BC09] にて解析されている．Poly1305 はナンスを用いる MAC であり，

$$\mathsf{Poly}_{R,K}(N, M) = \mathsf{Poly}_{R,K}(N', M')$$

を満たすナンスとメッセージのペア $((N, M), (N', M'))$ が得られたとしても（ただし $N \neq N'$），それ以外の衝突を得るのには役に立たない．したがって [BC09] の枠組みでは Poly1305 は再偽造可能性の意味で理想的な MAC と同等の安全性を有する．一方で，$N = N'$ が満たされるような場合には次の 4.2.3 章にあるようにハッシュ鍵を導出でき，偽造攻撃が可能となる．

4.2.3 ナンス再利用

Poly1305 はナンスを用いる MAC であり，タグ生成オラクルに対して同じナンスを繰り返してはならない．このことは提案論文 [Ber05b] において強調して述べられており，実際ナンスが繰り返すような攻撃シナリオを考えると，容易にハッシュ鍵を導出できることが [BC09] にて述べられている．

敵が (N, M) と (N, M') をタグ生成オラクルにクエリすると（ただし $M \neq M'$），タグ生成オラクルは

$$T = H_R(M) + \mathrm{AES}_K(N) \bmod 2^{128}$$

および

$$T' = H_R(M') + \mathrm{AES}_K(N) \bmod 2^{128}$$

を敵に返す．したがって

$$H_R(M) - H_R(M') \equiv T - T' \pmod{2^{128}} \tag{3}$$

が成り立ち，これは R を未知変数とする高々$\max\{m, m'\}$ 次の合同方程式とみなせる．ここで $m = \lceil \mathsf{len}(M)/16 \rceil$，$m' = \lceil \mathsf{len}(M')/16 \rceil$ である．式 (3) を直接解けば R（の候補）が求まる．

例えば $M = \mathtt{0x01}$，$M' = \mathtt{0x00}$ とし，ナンス N に対するそれぞれのタグを T と T' とすると，$\lceil \mathsf{len}(M) \rceil = \lceil \mathsf{len}(M') \rceil = 1$ であるから $m = m' = 1$ となり，$H_R(M) = (2^8 + 2^0) \cdot R \bmod (2^{130} - 5)$ と $H_R(M') = 2^8 \cdot R \bmod (2^{130} - 5)$ となる．これらから，

$$\begin{aligned}((2^8 + 2^0) \cdot R \bmod (2^{130} - 5)) - (2^8 \cdot R \bmod (2^{130} - 5)) \\ \equiv T - T' \pmod{2^{128}}\end{aligned}$$

が得られる．高い確率で $(2^8 + 2^0) \cdot R < 2^{128} < 2^{130} - 5$ が成り立ち，この場合

$$R = T - T' \bmod 2^{128}$$

であるから R が求まる．

M や M' を敵が選択せず，より長い場合や，あるいは長さが異なる場合であっても一般に式 (3) を効率的に解くことが可能である．また，式 (3) が複数ある場合には，連立合同方程式を解く代わりに多項式の GCD を計算することによって R を導出できる．

ナンスの再利用による安全性の低下は顕著であり，このことが起こらないよう実装には注意が必要である．

4.2.4 弱鍵

多項式ハッシュ関数の弱鍵は Handschuh と Preneel により [HP08] にて議論されている．一般に，暗号方式に対して鍵空間全体の部分集合 $\mathcal{S}$ を考える．未知の鍵がその集合 $\mathcal{S}$ に属するときに暗号方式が予期しない挙動を示し，なおかつ未知の鍵がその集合 $\mathcal{S}$ に属するかどうかを判定することが容易なとき，その $\mathcal{S}$ を弱鍵集合という．

MAC の場合は予期しない挙動は偽造成功確率が平均よりも大幅に高いことなどを意味する．また，その鍵の集合 $\mathcal{S}$ の要素数が s であるとき，未知の鍵が $\mathcal{S}$ に属するかどうかを

- s 通りの鍵を試す全数探索よりも効率的に
- かつ，s 回の検証オラクルのアクセスよりも効率的に

判定できなければならない．

Poly1305 の場合，$R = 0$ のみからなる集合は弱鍵集合となる [HP08]．この場合任意の M に対して

$$H_R(M) = 0$$

が成立する．したがって，タグ生成オラクルが生成した任意の (N, M, T) に対し，(N, M', T) は検証オラクルに確率 1 で受理される．ここで，M' は M とは異なる任意のメッセージである．ただし $R = 0$ の生起確率が低いため，実際にこのことが問題となることはない [HP08]．

この解析は Saarinen [Saa12] と Procter と Cid [PC13, PC15] により一般化されている．また，[ABBT15] においても一般化されている．Procter と Cid [PC13, PC15] の要点は，多項式ハッシュ関数を用いた MAC においては，鍵空間のあらゆる部分集合が弱鍵集合になる，というものである．R をハッシュ関数 H の未知の鍵，$Q = (Q[0], \ldots, Q[\ell - 1])$ とし，多項式

$$Q(X) = \sum_{0 \le i \le \ell - 1} Q[i] \cdot X^{\ell - i}$$

を考える．$Q(X) = 0$ が R を根として持つとき，$Q(X)$ を偽造多項式という．Poly1305 の文脈では演算はすべて法 $2^{130} - 5$ 上で定義される．

ここで

$$\begin{aligned} H_R(M) &= C[0] \cdot R^m + C[1] \cdot R^{m-1} + \cdots + C[m-1] \cdot R \bmod (2^{130} - 5) \\ &= \sum_{0 \le i \le m-1} C[i] \cdot R^{m-i} \bmod (2^{130} - 5) \end{aligned}$$

に対し $m = \ell$ とすると，

$$H_R(M) + Q(R) = H_R(M)$$

であり，

$$\begin{aligned} H_R(M) + Q(R) &= \sum_{0 \le i \le m-1} (C[i] + Q[i]) \cdot R^{m-i} \bmod (2^{130} - 5) \\ &= H_R(M + Q) \end{aligned}$$

であるから，(N, M) に対する鍵 (R, K) のタグ T は，(N, M') に対する鍵 (R, K) の MAC にもなっている．ただし，

$$M' = M + Q = (M[0] + Q[0], \ldots, M[\ell - 1] + Q[\ell - 1])$$

である．この偽造は確率 1 でタグ検証オラクルに受理される．

また，未知の鍵 R が鍵集合 $\mathcal{S}$ に属するかどうかのテストは，偽造多項式 $Q(X) = \Pi_{S\in\mathcal{S}}(X - S)$ を用いて検証オラクルにクエリをすればよく，これは $\mathcal{S}$ のサイズによらない．Poly1305 におけるメッセージから多項式ハッシュ関数への変換のフォーマットに矛盾しないことが必要であるが，このことは限られた場合にしか問題とはならない．したがって，Poly1305 においては任意のサイズの弱鍵集合が存在する．また，Procter と Cid はこれを鍵回復攻撃に応用している [PC13, PC15].

弱鍵集合のサイズが s のとき，その鍵が実際に使われる確率は $s/2^{106}$ となる．また，これをテストするための偽造多項式の次数は s であるため，タグ検証オラクルへのクエリの長さも s となる．$s/2^{106}$ が実際に問題となるような大きな確率となるためには s を大きくとる必要があり，この場合クエリの長さも s となる．たとえば鍵が使用される確率を 2^{-32} とする場合，s は 2^{74} となり，クエリするメッセージの長さが 2^{74} ブロックになる．

したがって，Poly1305 においては任意のサイズの弱鍵集合が存在するものの，現実的には大きな問題とはならないと考えられる．

なお多項式ハッシュ関数を用いた MAC に対する弱鍵は，一般にブロック暗号において議論される弱鍵，あるいは等価鍵とは位置づけが異なる．鍵長 k ビット，ブロック長 n ビットのブロック暗号 $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ における等価鍵は，任意の $M \in \{0,1\}^n$ に対して等号 $E_K(M) = E_{K'}(M)$ が成立する鍵のペア (K, K') のことをいう．この場合全数探索にかかる計算量が 2^k を下回るため，ブロック暗号における等価鍵の存在は，そのブロック暗号が鍵長相当の安全性を有していないことを直接意味する．

一方 MAC においては一つの鍵が鍵空間から一様ランダムに選択され，その鍵の下で偽造成功確率が低いことを目的に設計されるものであり，弱鍵集合の存在自体が MAC の直接的な安全性に影響するものではない．

4.3 改良版・修正版

Wang らは Poly1305 の鍵を 16 バイトまで削減した IPMAC を提案している [WLW06a]．IPMAC は Poly1305 同様ナンスベースの MAC であり，多項式ハッシュ関数用の鍵 R を $R = E_K(0^n)$ とすることで，Poly1305 から得られる．ただし，ナンス空間から 0^n が除外されなければならない．AES が擬似ランダム置換との仮定のもと，IPMAC が証明可能安全であることが示されており，敵の偽造成功確率の上界が次の式で与えられることが証明されている [WLW06a].

$$\delta + \frac{q(q+1)}{2^{129}} + \frac{8\lceil \ell/16 \rceil}{2^{128}} \tag{4}$$

ただし，q はタグ生成オラクルへのアクセス回数，ℓ はタグ生成オラクルへのクエリにおける M の最大バイト長を表し，敵は 1 度だけタグ検証オラクル

にアクセスする．また，δ は高々$q+1$ 回のクエリをする任意の敵 $\mathcal{B}$ に対し，$\mathrm{AES}_K(\cdot)$ と 128 ビット上のランダム置換 $P(\cdot)$ の識別成功確率の上界を表す．式 (4) の導出においては R の特定のビットを 0 にすることは考慮されておらず，このため成功確率の分母が 2^{128} となっている．

また，Wang らは Poly1305 の鍵を 16 バイトまで削減し，なおかつナンスを必要としない OPMAC（One-Key Poly1305）を提案している [WLW06b]．これは $T \leftarrow H_R(M) + \mathrm{AES}_K(N) \bmod 2^{128}$ とするかわりに

$$T \leftarrow \mathrm{AES}_K(H_R(M) \bmod 2^{128})$$

とし，$R = E_K(0^n)$ とすることで Poly1305 から得られる．OPMAC は確定的 MAC であり，AES が擬似ランダム置換との仮定のもと，証明可能安全であることが示されている [WLW06b]．敵の偽造成功確率の上界は次の式で与えられる．

$$\delta + \frac{(q+2)^2(1+8\lceil \ell/16 \rceil)}{2^{129}} + \frac{1}{2^{128}} \tag{5}$$

記号は式 (4) と同一であり，R の特定ビットを 0 にすることを考慮していない．

一般に式 (5) で与えられる上界は式 (1) や式 (4) よりも劣っている．しかし，Poly1305 と IPMAC はナンスが繰り返さないことに安全性の根拠を置いており，OPMAC はナンスの管理が不要であるという長所がある．

5 Poly1305 の安全性評価

本章では Poly1305 の安全性評価を報告する．5.1 章で証明可能安全について考え，5.2 章では関連鍵攻撃を扱う．5.3 章では複数ユーザ安全性を考える．

5.1 証明可能安全性

Bernstein による一連の証明 [Ber05b, Ber05a] に問題点は見受けられず，Poly1305 は証明可能安全性を有する MAC である．

5.2 関連鍵攻撃

本章では，4.2.1 章で紹介した Wang らの関連鍵攻撃について，適切に入力メッセージを選択すれば Poly1305 に対する偽造攻撃が可能であることを示す．RKD 関数（関連鍵導出関数）の集合として

$$\Phi = \{(a,b) \mid R \mapsto aR + b \bmod (2^{130}-5)\}$$

を考える．敵は定数 a, b に対して $R \mapsto aR + b \bmod (2^{130}-5)$ という操作が可能である．

安全性定義　安全性の定義について，敵はタグ生成オラクルと，タグ検証オラクルにアクセスできる．

- タグ生成オラクルは $((a,b),N,M)$ を入力として受け取り，タグ $T \leftarrow \mathsf{Poly}_{aR+b,K}(N,M)$ を敵に返す．
- タグ検証オラクルは $((a,b),N,M,T^*)$ を入力として受け取り，タグ $T \leftarrow \mathsf{Poly}_{aR+b,K}(N,M)$ を計算し，$T = T^*$ であれば受理を示す $\top$ を返し，$T \neq T^*$ であれば拒否を示す $\bot$ を返す．

敵の攻撃目標はタグ検証オラクルが $\top$ を一度でも出力することである．ただし，敵はタグ生成オラクルに対して同じ (a,b) を用いる場合はナンスを繰り返してはならず，タグ生成オラクルがクエリ $((a,b),N,M)$ に対して T を返したならばそれ以降 $((a,b),N,M,T)$ を検証オラクルにクエリせず，また，クエリは繰り返さないものとする．

まず $(a,b) = (1,1)$ のときの攻撃が可能であることを示す．

関連鍵攻撃（$(a,b) = (1,1)$ の場合）　メッセージ $M = (M[0], M[1])$ と $M' = (M'[0], M'[1])$ を次のように定める．

- $M[0] = 2^{128} - 3$
- $M[1] = 2^{128} - 3$
- $M'[0] = 2^{128} - 3$
- $M'[1] = 2^{128} - 2$

このとき，先頭に 1 バイトの `0x01` を付加した値をそれぞれ $C = (C[0], C[1])$ と $C' = (C'[0], C'[1])$ とすれば，

- $C[0] = 2 \times 2^{128} - 3$
- $C[1] = 2 \times 2^{128} - 3$
- $C'[0] = 2 \times 2^{128} - 3$
- $C'[1] = 2 \times 2^{128} - 2$

となる．M のハッシュ値は $H_R(M) = C[0] \cdot R^2 + C[1] \cdot R \bmod (2^{130} - 5)$ であり，M のハッシュ値は $H_R(M') = C'[0] \cdot R^2 + C'[1] \cdot R \bmod (2^{130} - 5)$ である．

このとき，$H_R(M) = H_{R+1}(M')$，すなわち

$$C[0] \cdot R^2 + C[1] \cdot R \equiv C'[0] \cdot (R+1)^2 + C'[1] \cdot (R+1) \pmod{2^{130} - 5}$$

が成り立つ．実際，右辺は

$$\begin{aligned} &C[0]\cdot R^2 + C[1]\cdot R \\ &\quad \equiv (2\times 2^{128}-3)\cdot R^2 + (2\times 2^{128}-3)\cdot R \pmod{2^{130}-5} \end{aligned}$$

であり，左辺は

$$\begin{aligned} &C'[0]\cdot (R+1)^2 + C'[1]\cdot (R+1) \\ &\quad \equiv (2\times 2^{128}-3)\cdot (R+1)^2 + (2\times 2^{128}-2)\cdot (R+1) \pmod{2^{130}-5} \\ &\quad \equiv \text{右辺} + (2\times 2^{128}-3)(2R+1) + (2\times 2^{128}+R-2) \pmod{2^{130}-5} \\ &\quad \equiv \text{右辺} + (4\times 2^{128}-5)R + (2\times 4^{128}-5) \pmod{2^{130}-5} \\ &\quad \equiv \text{右辺} \end{aligned}$$

である．したがって $H_R(M) = H_{R+1}(M')$ が成り立つ．

(N, M) の鍵 (R, K) に対するタグ T が，(N, M') の鍵 $(R+1, K)$ に対するタグと一致するので，敵は $((1,1), N, M')$ をタグ生成オラクルにクエリし T を得たのち，$((1,0), N, M, T)$ をタグ検証オラクルにクエリすることで偽造攻撃に成功することができる．

次に一般の場合を考える．

関連鍵攻撃（一般の場合） $(a, b) = (1, 1)$ の場合と同様に，メッセージ $M = (M[0], M[1])$ と $M' = (M'[0], M'[1])$ に対し，先頭に 1 バイトの `0x01` を付加した値をそれぞれ $C = (C[0], C[1])$ と $C' = (C'[0], C'[1])$ とする．M のハッシュ値は $H_R(M) = C[0]\cdot R^2 + C[1]\cdot R \bmod (2^{130}-5)$ であり，M のハッシュ値は $H_R(M') = C[0]\cdot R^2 + C[1]\cdot R \bmod (2^{130}-5)$ である．

ここで，次の条件を満たす $a, b, C = (C[0], C[1]), M = (M[0], M[1]), C' = (C'[0], C'[1]), M' = (M'[0], M'[1])$ を考える．

条件 1 $C[0] = \texttt{0x01} \,\|\, M[0]$

条件 2 $C[1] = \texttt{0x01} \,\|\, M[1]$

条件 3 $C'[0] = \texttt{0x01} \,\|\, M'[0]$

条件 4 $C'[1] = \texttt{0x01} \,\|\, M'[1]$

条件 5 $C[0] \equiv a^2 C'[0] \pmod{2^{130}-5}$

条件 6 $C[1] \equiv 2abC'[0] + aC'[1] \pmod{2^{130}-5}$

条件 7 $bC'[0] + C'[1] \equiv 0 \pmod{2^{130}-5}$

このとき，$H_R(M) = H_{aR+b}(M')$，すなわち

$$C[0]\cdot R^2 + C[1]\cdot R \equiv C'[0]\cdot (aR+b)^2 + C'[1]\cdot (aR+b) \pmod{2^{130}-5}$$

が成り立つ．条件 1 から 4 は M と M' から C と C' の変換に関するフォーマットの規則を表しており，条件 5 から 7 はこの式が R に関する恒等式になることを意味している．

実際に，右辺 − 左辺 を整理すると

$$(a^2C'[0] - C[0])R^2 + (2abC'[0] + aC'[1] - C[1])R + (b^2C'[0] + bC'[1]) \equiv 0$$

となる．各条件は係数が 0 であることを保証しており，恒等式になっていることが確かめられる．したがって $H_R(M) = H_{aR+b}(M')$ が成り立つ．

これは (N, M) の鍵 (R, K) に対するタグ T が，(N, M') の鍵 $(aR + b, K)$ に対するタグと一致することを意味する．敵は $((a, b), N, M')$ をタグ生成オラクルにクエリし T を得たのち，$((1, 0), N, M, T)$ をタグ検証オラクルにクエリすることで偽造攻撃に成功することができる．

上記の条件を満たす $a, b, M[0], M[1], M'[0], M'[1]$ の導出方法について，mod $(2^{130} - 5)$ 上で定義される連立合同方程式

$$\begin{cases} 2^{128} + M[0] \equiv a^2(2^{128} + M'[0]) & (6) \\ 2^{128} + M[1] \equiv 2ab(2^{128} + M'[0]) + a(2^{128} + M'[1]) & (7) \\ b(2^{128} + M'[0]) + (2^{128} + M'[1]) \equiv 0 & (8) \end{cases}$$

を考える．

1. a と $M'[0]$ を適当に与えると，式 (6) から $M[0]$ が一意に定まる．
2. この $M'[0]$ に対し b を適当に定めると式 (8) から $M'[1]$ が得られる．
3. これまでに定めた $a, b, M'[0], M'[1]$ から式 (7) より $M[1]$ が定まる．これらの値が mod 2^{128} の要素となっていれば攻撃成功である．

これはいつでも成功するわけではないが，実際に多くのパラメータを導出することができる．

たとえば $a = 55$ と $M'[0] = 2^{128} - 3$ とすると，式 (6) から

$$\begin{aligned} M[0] &= 340282366920938463463374607431768209941 \\ &= 2^{128} - 1515 \end{aligned}$$

となる．ただし，10 進数で表記している．$b = 3$ としてこれらを式 (8) に代入すると

$$\begin{aligned} M'[1] &= 340282366920938463463374607431768211455 \\ &= 2^{128} - 1 \end{aligned}$$

を得る．最後に式 (7) から

$$\begin{aligned} M[1] &= 340282366920938463463374607431768211371 \\ &= 2^{128} - 85 \end{aligned}$$

表 1: 関連鍵攻撃に成功するパラメータの例

a	55
b	1
$M[0]$	$2^{128}-1515$
$M[1]$	$2^{128}-30$
$M'[0]$	$2^{128}-3$
$M'[1]$	$2^{128}-2$
a	61
b	1
$M[0]$	$2^{128}-5584$
$M[1]$	$2^{128}-94$
$M'[0]$	$2^{128}-4$
$M'[1]$	$2^{128}-1$
a	99
b	1
$M[0]$	$2^{128}-14704$
$M[1]$	$2^{128}-151$
$M'[0]$	$2^{128}-4$
$M'[1]$	$2^{128}-1$

を得る．いずれも 2^{128} 未満であり，攻撃に成功する．その他の攻撃に成功するパラメータの例を表 1 に示す．

したがって定数 a,b に対して $R \mapsto aR+b \bmod (2^{130}-5)$ といった操作を行うことができる敵は Poly1305 の偽造が可能である．一方で，関連鍵攻撃の成立可能性については鍵にどのような操作を行うことができるか，すなわち RKD 関数の集合の定義が重要であり，ここで挙げたような算術演算を用いる例は人工的であり，現実的な脅威とはならないと考えられる．

5.3 複数ユーザ安全性

複数ユーザが存在するときの共通鍵暗号の安全性は近年になって盛んに研究されている [BBT16, CMS11, HT16, HT17, FJM14, LMP17, ML15, ST16]．まず Chatterjee ら [CMS11] による MAC の複数ユーザ安全性を定義する．

$H : \mathcal{K}_H \times \mathcal{D} \to \{0,1\}^n$ を MAC とする．$\mathcal{K}_H$ は鍵空間，$\mathcal{D}$ は定義域，タグは $T \in \{0,1\}^n$ であり，確定的な MAC を考えている．シングルユーザの設定においては，敵 $\mathcal{A}$ はタグ生成オラクルに M をクエリしタグ $T = H_K(M)$ を受け取る．敵は偽造文 (M', T') を出力する．$\mathcal{A}$ の目的は検証に受理されるよ

うな，すなわち $T' = H_K(M')$ であるような (M', T') を出力することである．この安全性を MAC1 安全性という．敵 $\mathcal{A}$ は，実行時間 t，確率 ϵ で偽造に成功するとき，H を MAC1 の意味で (t, ϵ)-break するという．H を (t, ϵ)-break するような敵が存在しないとき，H は MAC1 の意味で (t, ϵ)-安全であるという．

複数ユーザ設定においては，まず U 個の独立な鍵 $K_1, \ldots, K_U$ が選ばれ，これらはユーザ 1 から U に対応する．敵 $\mathcal{A}$ は次のようにオラクルにアクセスする．すなわち，クエリ (i, M) に対し（ただし $i \in \{1, \ldots, U\}$ であり，$M \in \mathcal{D}$ である），タグ生成オラクルは $T = H_{K_i}(M)$ を返す．さらに敵はオラクルを corrupt することができ，任意の $i \in \{1, \ldots, U\}$ に対してその鍵 K_i を得ることができる．敵の目標は次を満たす (i, M, T) を出力することである．

- $i \in \{1, \ldots, U\}$ であり，$M \in \mathcal{D}$ である
- オラクル i は corrupt されていない
- H_{K_i} に対して M をクエリしていない
- $T = H_{K_i}(M)$

この安全性を MAC*安全性という．敵 $\mathcal{A}$ は，実行時間 t，確率 ϵ で偽造に成功するとき，H を MAC*の意味で (t, ϵ)-break するという．H を (t, ϵ)-break するような敵が存在しないとき，H は MAC*の意味で (t, ϵ)-安全であるという．

Chatterjee らは MAC が MAC1 の意味で (t, ϵ)-安全であれば，MAC*の意味で $(t, \epsilon U)$-安全であることを示している [CMS11]．

ナンスを用いる MAC についても同様に安全性定義，安全性証明が可能である．この場合，各ユーザについてナンスを再利用することはできないが，異なるユーザ間で同じナンスを利用することは許される．$H : \mathcal{K}_H \times \mathcal{N} \times \mathcal{D} \to \{0,1\}^n$ をナンスを用いる MAC とする．$\mathcal{K}_H$ は鍵空間，$\mathcal{N}$ はナンス空間，$\mathcal{D}$ は定義域，タグは $T \in \{0,1\}^n$ である．

シングルユーザの設定においては，敵 $\mathcal{A}$ はタグ生成オラクルに (N, M) をクエリしタグ $T = H_K(N, M)$ を受け取る．$\mathcal{A}$ の目的は $T' = H_K(N', M')$ であるような (N', M', T') を出力することである（MAC1 安全性）．敵 $\mathcal{A}$ は，実行時間 t，確率 ϵ で偽造に成功するとき，H を MAC1 の意味で (t, ϵ)-break するという．H を (t, ϵ)-break するような敵が存在しないとき，H は MAC1 の意味で (t, ϵ)-安全であるという．

複数ユーザ設定において，U 個の独立な鍵 $K_1, \ldots, K_U$ が選ばれる．敵 $\mathcal{A}$ は，クエリ (i, N, M) に対し（ただし $i \in \{1, \ldots, U\}$，$N \in \mathcal{N}$，$M \in \mathcal{D}$ である），タグ生成オラクルは $T = H_{K_i}(N, M)$ を返す．ただし，同じオラクル i に N を繰り返し用いてはならない．さらに敵はオラクルを corrupt することができ，任意の $i \in \{1, \ldots, U\}$ に対してその鍵 K_i を得ることができる．敵の目標は次を満たす (i, N, M, T) を出力することである．

- $i \in \{1, \ldots, U\}$，$N \in \mathcal{N}$，$M \in \mathcal{D}$ である
- オラクル i は corrupt されていない
- H_{K_i} に対して (N, M) をクエリして T を得ていない
- $T = H_{K_i}(N, M)$

この安全性を MAC*安全性という．敵 $\mathcal{A}$ は，実行時間 t，確率 ϵ で偽造に成功するとき，H を MAC*の意味で (t, ϵ)-break するという．H を (t, ϵ)-break するような敵が存在しないとき，H は MAC*の意味で (t, ϵ)-安全であるという．

[CMS11] と同じ議論により，敵の攻撃成功確率は上界はシングルユーザにおける成功確率の上界の U 倍となる．実際，次にように帰着できる．$\mathcal{A}$ を H を MAC*の意味で (t, ϵ)-break する敵とする．この $\mathcal{A}$ を用いて，H に対して MAC1 の意味で偽造攻撃をする $\mathcal{B}$ を構成する．

- $\mathcal{B}$ は $\mathcal{A}$ が偽造を行うインデックス $j \in \{1, \ldots, U\}$ を一様ランダムに選択する．
- $\mathcal{B}$ は $i \in \{1, \ldots, U\}, i \neq j$ に対応するユーザの鍵 K_i を一様ランダムに選択する．ユーザ j の鍵は $\mathcal{B}$ のタグ生成オラクルが保持している鍵 K である．
- $\mathcal{B}$ は $\mathcal{A}$ を実行する．ユーザ $i \neq j$ に対する corrupt クエリとタグ生成クエリ (i, N, M) に対しては自身が保持する K_i を用いる．
- ユーザ j に対するタグ生成クエリは自身のタグ生成オラクルに問い合わせをする．ユーザ j に対する corrupt クエリは攻撃失敗である．
- $\mathcal{A}$ が (j, N, M, T) を出力したら，$\mathcal{B}$ は (N, M, T) を出力して終了する．そうでなければ $\mathcal{B}$ は攻撃失敗である．

$\mathcal{A}$ の動作は $\mathcal{B}$ がユーザ j を corrupt しない限り j の値とは独立である．したがって，$\mathcal{A}$ が攻撃に成功し，なおかつ $\mathcal{B}$ が j を正しく推測する確率は ϵ/U であるから，$\mathcal{B}$ は H を MAC1 の意味で $(t, \epsilon/U)$-break する．したがって，H が MAC1 の意味で (t, ϵ)-安全であれば，MAC*の意味で $(t, \epsilon U)$-安全である．

Poly1305 においては，式 (2) より，ユーザ数が U であれば敵の攻撃成功確率は高々

$$U\left(\delta + \frac{14\lceil \ell/16 \rceil}{2^{106}}\right) \tag{9}$$

となる．ただし $q \leq 2^{64}$ と仮定した．また，q はタグ生成オラクルへのアクセス総数の最大値（ユーザ i に q_i 回アクセスするようであれば，$\sum_{1 \leq i \leq U} q_i \leq q$），タグ検証オラクルへは [CMS11] の安全定義に従い 1 度だけアクセスすることとし，ℓ はクエリのメッセージの最大バイト長を表す．

式 (9) は Poly1305 の複数ユーザが存在する場合の敵の攻撃成功確率の上界を与えており，この意味で Poly1305 は複数ユーザの設定においても証明可能安全性を有している．

一方で，いくつかの暗号方式においては複数ユーザの安全性がシングルユーザの安全性から低下しない，あるいは限定的な影響しかない場合があることが知られている [BBT16, HT16, HT17, LMP17, ML15, ST16]．

例えば，Luykx らは複数ユーザにおける PRP-PRF switching lemma の解析を示している [LMP17]．シングルユーザにおける PRP-PRF switching lemma は，

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{Perm}(n)}(\mathcal{A}) = \Pr[\mathcal{A}^P \Rightarrow 1] - \Pr[\mathcal{A}^F \Rightarrow 1]$$

として，

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{Perm}(n)}(\mathcal{A}) \leq \frac{0.5q^2}{2^n}$$

を示している [BR06]．ここで，$P \xleftarrow{\$} \mathrm{Perm}(n)$ は $\{0,1\}^n$ 上のランダム置換，$F \xleftarrow{\$} \mathrm{Func}(n)$ は $\{0,1\}^n$ から $\{0,1\}^n$ へのランダム関数，q は $\mathcal{A}$ のクエリ回数を表す．$\mathrm{Perm}(n)$ は $\{0,1\}^n$ 上のすべての置換の集合，$\mathrm{Func}(n)$ は $\{0,1\}^n$ から $\{0,1\}^n$ へのすべての関数の集合である．複数ユーザの PRP-PRF switching lemma は，ユーザ数を U として

$$\mathbf{Adv}^{\text{mu-prf}}_{\mathrm{Perm}(n)}(\mathcal{A}) = \Pr[\mathcal{A}^{P_1,\ldots,P_U} \Rightarrow 1] - \Pr[\mathcal{A}^{F_1,\ldots,F_U} \Rightarrow 1]$$

を考える（上記のように利得関数に mu-をつけて複数ユーザを考えていることを表す）．

自明な証明手法に従えば

$$\mathbf{Adv}^{\text{mu-prf}}_{\mathrm{Perm}(n)}(\mathcal{A}) \leq U\frac{0.5q^2}{2^n}$$

が上界として得られるが，

$$\mathbf{Adv}^{\text{mu-prf}}_{\mathrm{Perm}(n)}(\mathcal{A}) \leq \frac{0.5q^2}{2^n} \tag{10}$$

となることが示せる [LMP17]．ただし，q は $\mathcal{A}$ のクエリの総数の最大値である．また Luykx らは [LMP17] において Wegman-Carter 認証コードを解析しており，Poly1305 はこの枠組みにあてはまる．

$H : \mathcal{K}_H \times \mathcal{D} \to \mathcal{R}$ を鍵付きハッシュ関数，$\mathcal{K}_H$ は有限の鍵集合，$\mathcal{D}$ は定義域，$\mathcal{R}$ は値域であり，2 項演算 $+$ について群であるとし，その逆演算を $-$ とする．また，$F : \mathcal{R} \to \mathcal{R}$ をランダム関数とする．

このとき，Wegman-Carter 認証コード $\mathsf{WC} = (\mathsf{Tag}, \mathsf{Ver})$ は次のように定義される．タグ生成関数 Tag はナンス $N \in \mathcal{R}$ とメッセージ $M \in \mathcal{D}$，鍵 $K \in \mathcal{K}_H, F$ に対し，

$$T = H_K(M) + F(N)$$

を出力する．$T \leftarrow \mathsf{Tag}_{K,F}(N,M)$ と表記する．検証アルゴリズム Ver は鍵 K と F を保持しており，入力 (N,M,T^*) に対し $T \leftarrow \mathsf{Tag}_{K,F}(N,M)$ を計算し，$T = T^*$ であれば $\top$ を，そうでなければ $\bot$ を返す．

このシングルユーザ安全性は

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathsf{WC}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{Tag}_{K,F},\mathsf{Ver}_{K,F}} \text{ forges}]$$

と定義される．$\mathcal{A}$ forges はタグ検証オラクルが一度でも $\top$ を返すイベントである．複数ユーザの安全性は

$$\mathbf{Adv}^{\text{mu-auth}}_{\mathsf{WC}}(\mathcal{B}) = \Pr[\mathcal{B}^{\mathsf{Tag}_{K_1,F_1},\mathsf{Ver}_{K_1,F_1},\ldots,\mathsf{Tag}_{K_U,F_U},\mathsf{Ver}_{K_U,F_U}} \text{ forges}]$$

と定義される．この場合の $\mathcal{B}$ forges はいずれかのタグ検証オラクルが一度でも $\top$ を返すイベントである．

Luykx らの結論は，H が多項式ハッシュ関数であれば，$\mathbf{Adv}^{\text{mu-auth}}_{\mathsf{WC}}(\mathcal{B})$ は U に依存せず，最良の $\mathbf{Adv}^{\mathrm{auth}}_{\mathsf{WC}}(\mathcal{A})$ が $\mathbf{Adv}^{\text{mu-auth}}_{\mathsf{WC}}(\mathcal{B})$ の上界になる，というものである [LMP17]．

これを Poly1305 に当てはめる．AES-128 の代わりに $\{0,1\}^n$ 上のランダム置換を用いた Poly1305 を $\text{Poly1305}^{\mathsf{p}} = (\mathsf{Poly}^{\mathsf{p}},\mathsf{Ver}^{\mathsf{p}})$ とし，$\{0,1\}^n$ から $\{0,1\}^n$ へのランダム関数を用いた Poly1305 を $\text{Poly1305}^{\mathsf{f}} = (\mathsf{Poly}^{\mathsf{f}},\mathsf{Ver}^{\mathsf{f}})$ とする．

式 (2) は，

$$\mathbf{Adv}^{\mathrm{auth}}_{\text{Poly1305}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{auth}}_{\text{Poly1305}^{\mathsf{p}}}(\mathcal{A})$$

および

$$\mathbf{Adv}^{\mathrm{auth}}_{\text{Poly1305}^{\mathsf{p}}}(\mathcal{A}) \le \frac{14q'\lceil \ell/16 \rceil}{2^{106}} \tag{11}$$

を示している．$\mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{B})$ は $E = \text{AES-128}$ の擬似ランダム置換としての安全性を表す．

ここで $\mathbf{Adv}^{\text{mu-auth}}_{\text{Poly1305}}(\mathcal{A})$ について考える．まず

$$\begin{aligned}\mathbf{Adv}^{\text{mu-auth}}_{\text{Poly1305}}(\mathcal{A}) \le\ & \mathbf{Adv}^{\text{mu-prp}}_{E}(\mathcal{C}) && (12)\\ & + \mathbf{Adv}^{\text{mu-prf}}_{\text{Perm}(n)}(\mathcal{B}) && (13)\\ & + \mathbf{Adv}^{\text{mu-auth}}_{\text{Poly1305}^{\mathsf{f}}}(\mathcal{A}) && (14)\end{aligned}$$

となる．式 (12) は $E = \text{AES-128}$ の複数ユーザにおける擬似ランダム置換としての安全性を表している．式 (13) は複数ユーザにおける PRP-PRF switch であり，式 (10) より $0.5(q+q')^2/2^{128}$ が上界となる（q は $\mathcal{A}$ のタグ生成オラクルへのアクセス総数の最大値，q' はタグ検証オラクルへのアクセス総数の最大値である）．式 (14) は，[LMP17] の結果から，シングルユーザの安全性

$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{Poly1305^f}}(\mathcal{A})$ が上界となる．式 (11) より

$$\begin{aligned}\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{Poly1305^f}}(\mathcal{A}) &\leq \frac{0.5(q+q')^2}{2^{128}} + \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{Poly1305^p}}(\mathcal{A}) \\ &\leq \frac{0.5(q+q')^2}{2^{128}} + \frac{14q'\lceil \ell/16 \rceil}{2^{106}}\end{aligned}$$

であるから，最終的に，

$$\mathbf{Adv}^{\text{mu-auth}}_{\mathrm{Poly1305}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{mu-prp}}_{E}(\mathcal{C}) + \frac{(q+q')^2}{2^{128}} + \frac{14q'\lceil \ell/16 \rceil}{2^{106}} \qquad (15)$$

を得る．ただし，q はタグ生成オラクルへのアクセス総数の最大値，q' はタグ検証オラクルへのアクセス総数の最大値，ℓ はクエリの M の最大バイト長である．$\mathbf{Adv}^{\text{mu-prp}}_{E}(\mathcal{C})$ は高々$q+q'$ 回のクエリにより，U 個のブロック暗号のオラクル $(E_{K_1}, \ldots, E_{K_U})$ と U 個のランダム置換のオラクル $(P_1, \ldots, P_U)$ を識別する利得である．

式 (15) で与えられる上界の第一項の $\mathbf{Adv}^{\text{mu-prp}}_{E}(\mathcal{C})$ はユーザ数に依存する [Bih02, BMS05] ものの，第二項，第三項がユーザ数 U と独立である．したがって，式 (15) はユーザ数の影響が限定的な安全性バウンドを与えている．

6 ChaCha20-Poly1305 の仕様

本章では ChaCha20-Poly1305 の仕様を示す [NL15]．構成要素としてストリーム暗号 ChaCha20 で利用されているブロック関数 [Ber08] と，Poly1305 に基づく多項式ハッシュ関数を用いる．ナンスごとに Poly1305 の鍵が変わることに大きな特徴がある．

ChaCha20-Poly1305 を CC-Poly と略記する．CC-Poly は 2 つのアルゴリズム $(\mathsf{Enc}, \mathsf{Dec})$ からなる．

- 鍵は 256 ビット（32 バイト）の鍵 K であり，暗号化アルゴリズム Enc_K は入力として 96 ビット（12 バイト）のナンス N，任意長の AD（associated data）A，および任意長の平文 M をとり，平文と同じ長さの暗号文 C と 128 ビット（16 バイト）のタグ T を出力する．
- 復号アルゴリズム Dec_K は入力として 96 ビット（12 バイト）のナンス N，任意長の AD A，任意長の暗号文 C，および 128 ビット（16 バイト）のタグ T をとり，暗号文と同じ長さの平文 M か，あるいは拒否を示す $\perp$ を返す．

CC-Poly はその内部で ChaCha20 のブロック関数 CC を用いる．CC の入出力は

$$\mathsf{CC} : \mathcal{K}_{\mathsf{CC}} \times \{0,1\}^{32} \times \{0,1\}^{96} \to \{0,1\}^{512}$$

で与えられる．ここで $\mathcal{K}_{\mathsf{CC}} = \{0,1\}^{256}$ は鍵空間である．入力として 256 ビット（32 バイト）の秘密鍵 K，32 ビット（4 バイト）のカウンタ ctr，および 96 ビット（12 バイト）のナンス N をとり，512 ビット（64 バイト）の鍵ストリーム Z を出力する．$Z \leftarrow \mathsf{CC}_K(\mathsf{ctr}, N)$ と表記する．

CC-Poly では，Poly1305 の多項式ハッシュ関数の部分のみを考える．すなわち，Poly の入出力を

$$\mathsf{Poly} : \mathcal{K}_{\mathsf{Poly}} \times \{0,1\}^* \to \{0,1\}^{128}$$

とする．ここで $\mathcal{K}_{\mathsf{Poly}} = \{0,1\}^{128}$ は鍵空間である．入力として 128 ビット（16 バイト）の秘密鍵 R と任意長のメッセージ Y をとり，ハッシュ値 $T \leftarrow \mathsf{Poly}_K(Y)$ を出力する．ただし，R は特定のビットが 0 になるようにマスクされる．

暗号化アルゴリズム Enc_K は次のように動作する．

- $\mathsf{CC}_K(0 \,\|\, N)$ を計算し，その上位 128 ビットを R，次の 128 ビットを S として，Poly のワンタイム鍵 R と擬似ワンタイムパッド S を生成する．
- 鍵ストリーム $Z = \mathsf{CC}_K(1, N) \,\|\, \mathsf{CC}_K(2, N) \,\|\, \cdots$ を生成し，これと平文 M との排他的論理和から暗号文 $C = Z \oplus M$ を生成する．
- まず AD A と暗号文 C からメッセージ Y を次のように構成する．

  $$Y \leftarrow A \,\|\, \mathsf{pad}(A) \,\|\, C \,\|\, \mathsf{pad}(C) \,\|\, \mathsf{pad}(A, C)$$

  ここで $\mathsf{pad}(A)$ は $A \,\|\, \mathsf{pad}(A)$ の長さが 16 バイトの整数倍になるような最も短いゼロバイト列 $\mathtt{0x0}\ldots\mathtt{0}$ であり，$\mathsf{pad}(C)$ は $C \,\|\, \mathsf{pad}(C)$ の長さが 16 バイトの整数倍になるような最も短いゼロバイト列 $\mathtt{0x0}\ldots\mathtt{0}$ である．$\mathsf{pad}(A)$ と $\mathsf{pad}(C)$ は A ないしは C が初めから 16 バイトの整数倍であれば空列となる．$\mathsf{pad}(A, C)$ は $\mathsf{len}(A)$ と $\mathsf{len}(C)$ をそれぞれ 8 バイトで表現したバイト列の連結である．メッセージ Y の長さはつねに 16 バイトの整数倍となる．

  最後に，$T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$ としてタグ T を生成する．ただし，R の特定ビットは 0 とする．

復号アルゴリズム Dec_K は次のように動作する．

- $\mathsf{CC}_K(0 \,\|\, N)$ を計算し，その上位 128 ビットを R，次の 128 ビットを S として，Poly のワンタイム鍵 R と擬似ワンタイムパッド S を生成する．
- 暗号化アルゴリズムと同様の手順で AD A と暗号文 C からメッセージ Y を構成する．次に $T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$ としてタグ T を生成する．ただし，R の特定ビットは 0 とする．入力されたタグとこれが一致すれば次に進み，そうでなければ拒否する．

Algorithm $\mathsf{Enc}_K(N, A, M)$

1. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(M))$
2. $C \leftarrow M \oplus Z$
3. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
4. **return** (C, T)

Algorithm $\mathsf{Dec}_K(N, A, C, T^*)$

1. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
2. **if** $T \neq T^*$ **then return** $\perp$
3. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(C))$
4. $M \leftarrow C \oplus Z$
5. **return** M

図 3: $(C, T) \leftarrow \mathsf{Enc}_K(N, A, M)$ と $M/\perp \leftarrow \mathsf{Dec}_K(N, A, C, T)$ の定義

- 鍵ストリーム $Z = \mathsf{CC}_K(1, N) \,\|\, \mathsf{CC}_K(2, N) \,\|\, \cdots$ を生成し，これと暗号文 C との排他的論理和から平文 $M = Z \oplus C$ を生成する．

CC-Poly $= (\mathsf{Enc}, \mathsf{Dec})$ の擬似コードを図 3 に示す．また，内部で用いるサブルーチンの定義を図 4，図 5 に示す．参考として，ChaCha20 ブロック関数 CC の仕様を付録 Aに示す．図 6 に暗号化アルゴリズムの流れを示す．

7 ChaCha20-Poly1305 に関する文献調査

本章では ChaCha20-Poly1305 に関する文献調査の報告をする．7.1 章で Procter による証明可能安全性を扱い，その他を 7.2 章で扱う．

7.1 証明可能安全性

CC-Poly の安全性は Procter により示されている [Pro14]．安全性定義は [BN08, RBB03] によるものを考えており，暗号化オラクルと復号オラクルにアクセスする敵を考える．暗号化オラクルはクエリ (N, A, M) に対し $(C, T) \leftarrow \mathsf{Enc}_K(N, A, M)$ を返す．復号オラクルはクエリ (N, A, C, T) に対し M か $\perp$ を返す．このとき，CC-Poly の AE 安全性を次のように定義する．

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathsf{CC\text{-}Poly}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{Enc}_K, \mathsf{Dec}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$, \perp} \Rightarrow 1]$$

Algorithm $\mathsf{KGen}_K(N, \ell)$

1. $m \leftarrow \lceil \ell/64 \rceil$
2. **for** $i = 1$ **to** m **do**
3. $\quad Z[i] \leftarrow \mathsf{CC}_K(i, N)$
4. $Z \leftarrow$ first ℓ bytes of $Z[1] \,\|\, \cdots \,\|\, Z[m]$
5. **return** Z

Algorithm $\mathsf{Tag}_K(N, A, C)$

1. $(R, S) \xleftarrow{16}$ first 32 bytes of $\mathsf{CC}_K(0, N)$
2. $R \leftarrow R \wedge \texttt{0x0ffffffc0ffffffc0ffffffc0fffffff}$
3. $Y \leftarrow A \,\|\, \mathsf{pad}(A) \,\|\, C \,\|\, \mathsf{pad}(C) \,\|\, \mathsf{pad}(A, C)$
4. $T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$
5. **return** T

図 4: $Z \leftarrow \mathsf{KGen}_K(N, \ell)$ と $T \leftarrow \mathsf{Tag}_K(N, A, C)$ の定義

Algorithm $\mathsf{Poly}_R(Y)$

1. $(Y[0], \ldots, Y[y-1]) \xleftarrow{16} Y$ // $y = \lceil \mathsf{len}(Y)/16 \rceil$
2. **for** $i = 1$ **to** $y - 1$ **do**
3. $\quad C[i] \leftarrow Y[i] + \texttt{0x01} \cdot 2^{8\mathsf{len}(Y[i])}$
4. $T \leftarrow C[0]$
5. **for** $i = 1$ **to** $y - 1$ **do**
6. $\quad T \leftarrow R \cdot C[i] \bmod (2^{130} - 5)$
7. $T \leftarrow R \cdot T \bmod (2^{130} - 5)$
8. $T \leftarrow T \bmod 2^{128}$
9. **return** T

図 5: $T \leftarrow \mathsf{Poly}_R(Y)$ の定義

だたし，\$ は Enc_K と同じ長さの乱数を返し，$\perp$ はつねに $\perp$ を返すオラクルである．また，敵は暗号化オラクルに同じナンスを用いてクエリしてはならず，暗号化クエリ (N, A, M) に対し (C, T) を得たのちに (N, A, C, T) を復号オラクルにクエリしてはならず，クエリは繰り返さないとする．

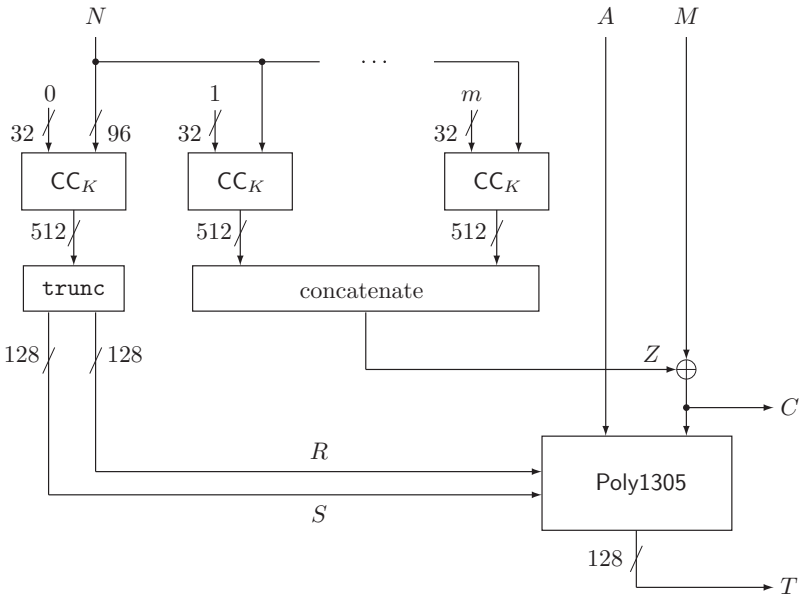

図 6: ChaCha20-Poly1305 の暗号化アルゴリズムの全体像

このとき Procter は次の式を示した [Pro14].

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathsf{CC\text{-}Poly}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CC}}(\mathcal{B}) + q'' \frac{8\lceil \ell_{\max}/16 \rceil}{2^{106}} \tag{16}$$

ただし q'' は復号オラクルへのアクセス回数を表し，$\ell_{\max}$ は暗号化クエリと復号クエリに対する Poly への入力 Y の最大バイト長を表す．また，$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CC}}(\mathcal{B})$ は CC の擬似ランダム関数として安全性を表し，これは小さいと考えられている．

7.2 その他の文献

Bhargavan らは TLS 1.3 record layer の安全性を解析しており [DFK+17]，その中で AES-GCM と ChaCha20-Poly1305 を含む認証暗号化方式の一般的構成法を示し，TLS 1.3 での使用に特化した証明可能安全性を示している．それに基づき，TLS 1.3 record layer 全体の安全性を議論している．

Nir と Langley による RFC 7539 においても ChaCha20-Poly1305 の安全性が議論されている [NL15]．使用にあたってはナンスが繰り返さないことが重要である点，繰り返した場合は平文の排他的論理和が漏えいすることが述べられている．またサイドチャネル攻撃ついての言及があり，タイミングによるサイドチャネル攻撃を避けるために実行時間が一定となるよう実装するべきである，ということが述べられている．

Ashur らはナンスの再利用についての解析を行っている [ADL17]．通常ナンスを再利用すると，平文の排他的論理和が漏えいし暗号化としての安全性が失われる．あるいはハッシュ関数の鍵が漏えいし，偽造攻撃が可能となる．Ashur らは，いくつかのナンスが再利用されたとしても，ある時点以降ナンスの再利用がなければ，CC-Poly は暗号化についても認証についてもその時点以降は安全であることを証明した．これは再利用されたナンスに対する安全性は失われるが，それ以外のナンスについては影響が波及しないことを示している．

8 ChaCha20-Poly1305 の安全性評価

本章では ChaCha20-Poly1305 の安全性評価を報告する．8.1 章で証明可能安全について考える．8.2 章では関連鍵攻撃，8.3 章では再偽造可能性，8.4 章ではナンス再利用，8.5 章では弱鍵，8.6 章では復号ミスユース耐性，8.7 章では複数ユーザ安全性について考える．

8.1 証明可能安全性

Procter による [Pro14] の証明には一か所軽微な誤りがあるが簡単に修正可能であり，最終的に得られるバウンドに影響しない．CC-Poly は証明可能安全性を有する認証暗号化方式である．

[Pro14] の証明では，鍵ストリームを真のランダム関数の出力から構成するゲーム (E^1, D^1) と，鍵ストリームを真のランダムビット列として生成するゲーム (E^2, D^2) を考え，これらのゲームが等価であるとされている．暗号化クエリにおいては等価であるが，復号クエリにおいては以前に暗号化クエリで使用したナンスを再利用することができ，この場合，鍵ストリームはすでに決まっている．したがって，ナンスの再利用をチェックするルーチンを加えるといった修正が必要であるが，これは容易であり，最終的な安全性のバウンドに影響しない．

8.2 関連鍵攻撃

CC-Poly では多項式ハッシュ関数を利用しているものの，その鍵はナンスごとに更新される．CC が関連鍵攻撃に対しても安全な擬似ランダム関数であれば，鍵に操作が加えられたとしてもすべての鍵が更新されるため，多項式ハッシュ関数にあったような関連鍵攻撃を行うことはできない．

実際に CC が関連鍵攻撃に対して安全な擬似ランダム関数であるとの仮定の下で CC-Poly 全体が関連鍵攻撃に対して安全な認証暗号化方式であることの証明が可能である．

たとえば，鍵導出関数の集合として

$$\Phi^{\oplus} = \{\mathrm{XOR}_{\Delta} \mid K \mapsto K \oplus \Delta, \Delta \in \mathcal{K}_F\}$$

を考える．敵が $\Delta_1, \Delta_2, \ldots$ を使用した場合，

$$\mathsf{Enc}_{K\oplus\Delta_1}, \mathsf{Dec}_{K\oplus\Delta_1}, \mathsf{Enc}_{K\oplus\Delta_2}, \mathsf{Dec}_{K\oplus\Delta_2}, \ldots$$

をオラクルとして有することになる．CC が関連鍵攻撃に対して安全な擬似ランダム関数であれば，これらの内部で使用されている

$$\mathsf{CC}_{K\oplus\Delta_1}, \mathsf{CC}_{K\oplus\Delta_2}, \ldots$$

はそれぞれ独立なランダム関数

$$F_i : \{0,1\}^{32} \times \{0,1\}^{96} \to \{0,1\}^{512}$$

に置き換わる．$(\mathsf{Enc}_{K\oplus\Delta_i}, \mathsf{Dec}_{K\oplus\Delta_i})$ をそれぞれ $(\mathsf{Enc}_{F_i}, \mathsf{Dec}_{F_i})$ と書くことにすれば，敵の観点からは独立な鍵 F_i を有する複数の ChaCha20-Poly1305 を攻撃することになる．これ以降は情報理論的な安全性証明となり，8.7 章にある複数ユーザ安全性と同じ議論から，敵の攻撃成功確率の上界を導出できる．

8.3 再偽造可能性

認証暗号化方式に対する再偽造不可能性は Forler らにより解析されている [FLLW17]．仮に敵が偽造に成功，あるいは出力の衝突に成功したと仮定し，その後別の偽造に成功するのにどれだけの計算量が必要かにより安全性を評価する．理想的な認証暗号化方式であれば，たとえば 2 つの入力 (N, A, M) と (N', A', M') のタグが衝突したとして（ただし $N \neq N'$），これら以外の衝突ペアを入手するのには役に立たない．

CC-Poly でも同様に，繰り返さないナンスを用いている限り，タグが衝突する 2 つの入力 (N, A, M) と (N', A', M') が得られたとしてもこれら以外の衝突を得るのには役に立たず，CC-Poly は再偽造可能性の意味で理想的な認証暗号化方式と同等の安全性を有する．一方で Poly1305 と同様に，$N = N'$ が満たされるような場合にはハッシュ鍵を導出でき，汎用的偽造が可能となる．

8.4 ナンス再利用

CC-Poly で使用する Poly では特定のフォーマットに従って入力メッセージ Y を生成するため，これに対処する必要があるが，ナンスを再利用すると R と S を復元することができる．また，得られた R と S を用いることにより，

ナンス N に対する偽造を行うことができる [II16]．ただし Ashur らにより示されているように，他のナンスについては影響がない [ADL17]．

本章では [II16, Ima17] に従い，偽造ができることを確認する．任意のナンス N を固定し，AD $A = \mathsf{0x00}$，$A' = \mathsf{0x0000}$ を考える．平文 M を空列とする．以下，整数 $a \leq b$ に対して

$$\begin{cases} [a,b] = \{x \mid a \leq x \leq b\} \\ [a,b) = \{x \mid a \leq x < b\} \\ (a,b] = \{x \mid a < x \leq b\} \\ (a,b) = \{x \mid a < x < b\} \end{cases}$$

とする．

敵は暗号化オラクル Enc_K に (N, A, M) と (N, A', M) をクエリし，(C, T) と (C', T') を得る．C と C' は空列である．$\mathsf{len}(A) = 1$ と $\mathsf{len}(A') = 2$ より，それぞれの Poly への入力メッセージ Y と Y' は

$$\begin{cases} Y = A + \mathsf{len}(A) \cdot 2^{128\lceil \mathsf{len}(A)/16 \rceil} = 1 \cdot 2^{128} \\ Y' = A' + \mathsf{len}(A') \cdot 2^{128\lceil \mathsf{len}(A')/16 \rceil} = 2 \cdot 2^{128} \end{cases}$$

となる．それぞれ 16 バイトごとに分割し $(Y[0], Y[1]) \xleftarrow{16} Y$ と $(Y'[0], Y'[1]) \xleftarrow{16} Y'$ とすれば，$Y[0] = 0$，$Y[1] = 1$，$Y'[0] = 0$，$Y'[1] = 2$ となる．特定のビットを 0 に固定したあとの R に対して，

$$\begin{cases} \mathbb{Y}(R) = 2^{128}R + (1 + 2^{128})R & (17) \\ \mathbb{Y}'(R) = 2^{128}R + (2 + 2^{128})R & (18) \end{cases}$$

とおくと，

$$\begin{cases} T = (\mathbb{Y}(R) \bmod (2^{130} - 5) + S) \bmod 2^{128} & (19) \\ T' = (\mathbb{Y}'(R) \bmod (2^{130} - 5) + S) \bmod 2^{128} & (20) \end{cases}$$

となる．ある整数 κ，κ' に対して

$$\begin{cases} T + 2^{128}\kappa = \mathbb{Y}(R) \bmod (2^{130} - 5) + S & (21) \\ T' + 2^{128}\kappa' = \mathbb{Y}'(R) \bmod (2^{130} - 5) + S & (22) \end{cases}$$

が成り立つ．T と T' を移項すれば，

$$\begin{cases} 2^{128}\kappa = (\mathbb{Y}(R) \bmod (2^{130} - 5) + S) - T & (23) \\ 2^{128}\kappa' = (\mathbb{Y}'(R) \bmod (2^{130} - 5) + S) - T' & (24) \end{cases}$$

を得る．$S \in [0, 2^{128})$ であり $T \in [0, 2^{128})$ であるから $S - T \in (-2^{128}, 2^{128})$ となり，式 (23) の右辺の範囲は

$$\begin{aligned} (\mathbb{Y}(R) \bmod (2^{130} - 5) + S) - T &\in (-2^{128}, 2^{130} + 2^{128} - 5) \\ &\subseteq (-2^{128}, 5 \cdot 2^{128}) \end{aligned}$$

となる．$\kappa \in (-2^{128}, 5\cdot 2^{128})$ であり，κ が整数であることより $\kappa \in \{0,1,2,3,4\}$ である．同様にして式 (24) より $\kappa' \in \{0,1,2,3,4\}$ が成り立つ．

次に，ある整数 λ，λ' に対して式 (21) と式 (22) は

$$\begin{cases} T - S + 2^{128}\kappa + (2^{130}-5)\lambda = \mathbb{Y}(R) & (25) \\ T' - S + 2^{128}\kappa' + (2^{130}-5)\lambda' = \mathbb{Y}'(R) & (26) \end{cases}$$

と書き直せる．式 (26) から式 (25) を引き，式 (17) と式 (18) を用いると

$$\begin{aligned} T' - T + 2^{128}(\kappa' - \kappa) + (2^{130}-5)(\lambda' - \lambda) &= \mathbb{Y}'(R) - \mathbb{Y}(R) \\ &= R \end{aligned}$$

が成り立つ．$\kappa^* = \kappa' - \kappa$ とし，$\lambda^* = \lambda' - \lambda$ とすると

$$T' - T + 2^{128}\kappa^* + (2^{130}-5)\lambda^* = R$$

を得る．これより

$$(2^{130}-5)\lambda^* = R - (T' - T) - 2^{128}\kappa^* \tag{27}$$

が成り立つ．$\kappa^* \in \{0, \pm 1, \pm 2, \pm 3, \pm 4\}$ である．

次に，式 (27) の $T' - T$ の符号により，次のように場合分けして考える．

- $T' - T \geq 0$
- $T' - T < 0$

$T' - T \geq 0$ のとき　$R \in [0, 2^{124})$ であり，$T' - T + 2^{128}\kappa^* \in [2^{128}\kappa^*, 2^{128}(1+\kappa^*))$ なので，式 (27) の範囲は

$$R - (T' - T) - 2^{128}\kappa^* \in (-2^{128}(1+\kappa^*), 2^{124} - 2^{128}\kappa^*)$$

となる．次に，9 通りの κ^* の値に対し，これに対応する λ^* が存在するかを確かめる．

$\kappa^* = -4$ のとき　$R - (T' - T) - 2^{128}\kappa^* = R - (T' - T) + 4 \cdot 2^{128} \in (3\cdot 2^{128}, 2^{124} + 4\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (3\cdot 2^{128}, 2^{124} + 4\cdot 2^{128})$ を満たす λ^* は $\lambda^* = 1$ である．

$\kappa^* = -3$ のとき　$R - (T' - T) - 2^{128}\kappa^* = R - (T' - T) + 3 \cdot 2^{128} \in (2\cdot 2^{128}, 2^{124} + 3\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (2\cdot 2^{128}, 2^{124} + 3\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = -2$ のとき　$R - (T' - T) - 2^{128}\kappa^* = R - (T' - T) + 2 \cdot 2^{128} \in (2^{128}, 2^{124} + 2\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (2^{128}, 2^{124} + 2\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = -1$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)+2^{128} \in (0, 2^{124}+2^{128})$ より，$(2^{130}-5)\lambda^* \in (0, 2^{124}+2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 0$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T) \in (-2^{128}, 2^{124})$ より，$(2^{130}-5)\lambda^* \in (-2^{128}, 2^{124})$ を満たす λ^* は $\lambda^* = 0$ である．

$\kappa^* = 1$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-2^{128} \in (-2\cdot 2^{128}, 2^{124}-2^{128})$ より，$(2^{130}-5)\lambda^* \in (-2\cdot 2^{128}, 2^{124}-2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 2$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-2\cdot 2^{128} \in (-3\cdot 2^{128}, 2^{124}-2\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (-3\cdot 2^{128}, 2^{124}-2\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 3$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-3\cdot 2^{128} \in (-4\cdot 2^{128}, 2^{124}-3\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (-4\cdot 2^{128}, 2^{124}-3\cdot 2^{128})$ を満たす λ^* は $\lambda^* = -1$ である．

$\kappa^* = 4$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-4\cdot 2^{128} \in (-5\cdot 2^{128}, 2^{124}-4\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (-5\cdot 2^{128}, 2^{124}-4\cdot 2^{128})$ を満たす λ^* は $\lambda^* = -1$ である．

以上より，$T'-T \geq 0$ の場合に (κ^*, λ^*) に対応する R の値の候補は

$$R = \begin{cases} T'-T-5 & \text{if } (\kappa^*, \lambda^*) = (-4, 1) \\ T'-T & \text{if } (\kappa^*, \lambda^*) = (0, 0) \\ T'-T-2^{128}+5 & \text{if } (\kappa^*, \lambda^*) = (3, -1) \\ T'-T+5 & \text{if } (\kappa^*, \lambda^*) = (4, -1) \end{cases}$$

となる．

$T'-T < 0$ のとき　$R \in [0, 2^{124})$ と $T'-T \in (-2^{128}, 0)$ より $R-(T'-T) \in (0, 2^{124}+2^{128})$ なので，

$$R-(T'-T)-2^{128}\kappa^* \in (-2^{128}\kappa^*, 2^{124}+2^{128}(1-\kappa^*))$$

である．$T'-T \geq 0$ の場合と同様に，9 通りの κ^* の値に対し，対応する λ^* が存在するかを確かめる．

$\kappa^* = -4$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)+4\cdot 2^{128} \in (4\cdot 2^{128}, 2^{124}+5\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (4\cdot 2^{128}, 2^{124}+5\cdot 2^{130})$ を満たす λ^* は存在しない．

$\kappa^* = -3$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)+3\cdot 2^{128} \in (3\cdot 2^{128}, 2^{124}+4\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (3\cdot 2^{128}, 2^{124}+4\cdot 2^{128})$ を満たす λ^* は $\lambda^* = 1$ である．

$\kappa^* = -2$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)+2\cdot 2^{128} \in (2\cdot 2^{128}, 2^{124}+3\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (2\cdot 2^{128}, 2^{124}+3\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = -1$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)+2^{128} \in (2^{128}, 2^{124}+2\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (2^{128}, 2^{124}+2\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 0$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T) \in (0, 2^{124}+2^{128})$ より，$(2^{130}-5)\lambda^* \in (0, 2^{124}+2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 1$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-2^{128} \in (-2^{128}, 2^{124})$ より，$(2^{130}-5)\lambda^* \in (-2^{128}, 2^{124})$ を満たす λ^* は $\lambda^* = 0$ である．

$\kappa^* = 2$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-2\cdot 2^{128} \in (-2\cdot 2^{128}, 2^{124}-2^{128})$ より，$(2^{130}-5)\lambda^* \in (-2\cdot 2^{128}, 2^{124}-2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 3$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-3\cdot 2^{128} \in (-3\cdot 2^{128}, 2^{124}-2\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (-3\cdot 2^{128}, 2^{124}-2\cdot 2^{128})$ を満たす λ^* は存在しない．

$\kappa^* = 4$ のとき　$R-(T'-T)-2^{128}\kappa^* = R-(T'-T)-4\cdot 2^{128} \in (-4\cdot 2^{128}, 2^{124}-3\cdot 2^{128})$ より，$(2^{130}-5)\lambda^* \in (-4\cdot 2^{128}, 2^{124}-3\cdot 2^{128})$ を満たす λ^* は $\lambda^* = -1$ である．

対応する R の値の候補は

$$R = \begin{cases} T'-T+2^{128}-5 & \text{if } (\kappa^*, \lambda^*) = (-3, 1) \\ T'-T+2^{128} & \text{if } (\kappa^*, \lambda^*) = (1, 0) \\ T'-T+5 & \text{if } (\kappa^*, \lambda^*) = (4, -1) \end{cases}$$

となる．

したがって $T'-T \geq 0$ のときに R の候補を 4 通り，$T'-T < 0$ のとき 3 通りに絞ることができる．これらを式 (19) あるいは (20) に代入すればこれらに対応する S の値を導出できる．

上記の手順を利用して次のようにして偽造攻撃を行うことができる．

1. 任意のナンス N を固定し，AD $A = \texttt{0x00}$，$A' = \texttt{0x0000}$ とする．平文 M を空列とする．
2. 敵は暗号化オラクル Enc_K に (N, A, M) と (N, A', M) をクエリし，(C, T) と (C', T') を得る（C と C' は空列である）．
3. 上記の手順に従い，(R, S) の候補を得てから，ランダムに (R, S) を決める．
4. 任意の (A'', C'') を選び（ただし $(A'', C'') \neq (A, C), (A', C')$），それに対応するタグ T'' を R と S を用いて計算する．
5. (N, A'', C'', T'') を検証オラクルにクエリする．

正しく (R, S) が選ばれればこの偽造攻撃は成功し，$T' - T \geq 0$ であれば確率 $1/4$，そうでなければ $1/3$ で成功する．

8.5 弱鍵

CC-Poly の秘密鍵は ChaCha20 ブロック関数で用いる鍵 K であり，Poly で使用する R はナンスと K に依存する．

Poly1305 にあったような弱鍵は R を鍵として考えているものであり，その意味では CC-Poly の直接的な弱鍵とはなっていない．たとえば $\mathcal{S} = \{R_1, \ldots, R_s\}$ として偽造多項式を $R_1, \ldots, R_s$ から $Q(X) = \Pi_{1 \leq i \leq s}(X - R_i)$ と生成したとして，未知の鍵 R がこの中に含まれていれば $Q(X)$ を用いて確率 1 で偽造に成功する．また，未知の鍵が集合 $\mathcal{S}$ に属しているかのテストは，$Q(X)$ を用いて復号オラクルにクエリをすることにより行うことができる．

したがってこの $\mathcal{S}$ は弱鍵の定義を満たすように考えられるが，これらに対応する CC の鍵の集合を列挙することは容易ではない．すなわち，trunc_{16} を入力の先頭 16 バイトを返す関数として，

$$R_1 = \mathsf{trunc}_{16}(\mathsf{CC}_{K_1}(0, N)), \ldots, R_s = \mathsf{trunc}_{16}(\mathsf{CC}_{K_s}(0, N)) \qquad (28)$$

を満たす鍵 $K_1, \ldots, K_s$ が存在する（各 K_i は一意とは限らない）．弱鍵集合としては $\{K_1, \ldots, K_s\}$ となるが，N と $R_1, \ldots, R_s$ からこれらを求めるのは困難である．

一方で，固定された N と K に対して R が一意に定まる．したがって，$K_1, \ldots, K_s$ を任意に固定し，$\mathcal{S} = \{K_1, \ldots, K_s\}$ として，式 (28) に従って $R_1, \ldots, R_s$ を定めると，この $\mathcal{S}$ は弱鍵集合としての要件を満たす．未知の鍵 K が $\mathcal{S}$ に含まれていれば $R_1, \ldots, R_s$ から生成される偽造多項式 $Q(X)$ を用いれば偽造攻撃に成功する．また，$Q(X)$ を用いて復号オラクルにクエリをすることにより，未知の鍵 K が集合 $\mathcal{S}$ に属しているかのテストを行うことができる．

したがって ChaCha20-Poly1305 には弱鍵が存在する．しかし，Poly1305 と同様に弱鍵の存在が現実的な問題とはならないと考えられるとともに，全体の鍵空間のサイズは 2^{256} であるからその影響は Poly1305 よりも小さいものと考えられる．

8.6 復号ミスユース耐性

復号ミスユース耐性は Andreeva らにより提案された安全性定義であり，検証で受理されないような暗号文に対する平文を入手できる状況を考える [ABL+14]．暗号化に関する安全性 PA（plaintext awareness）と認証に関する安全性 INT-RUP（integrity under releasing unverified plaintext）がある．

CC-Poly は暗号化に関する安全性を満たさないが，認証に関する安全性を満たすことが証明できる．本章では [II16, IMI16, Ima17, IMI17] に基づき，これらの概要を示す．

まずこの設定では，ChaCha20-Poly1305 を暗号化アルゴリズム，復号アルゴリズムだけではなく，暗号化，復号，検証の 3 つのアルゴリズムからなると捉える．それぞれ次のように定義される．

- 暗号化アルゴリズム Enc_K は 6 章と同様に定義され，鍵 K を用いて入力 (N, A, M) から暗号文とタグのペア (C, T) を出力する．
- 復号アルゴリズム Ver_K は (N, A, C, T^*) を入力とし，鍵ストリーム $Z = \mathsf{CC}_K(1, N) \,\|\, \mathsf{CC}_K(2, N) \,\|\, \cdots$ を生成し，これと暗号文 C との排他的論理和から平文 $M = Z \oplus C$ を生成する．検証に相当するチェックは行わず，T^* は用いない．また，つねに平文を返す．
- 検証アルゴリズム Ver_K は (N, A, C, T^*) を入力とし，次のように動作する．
  - $\mathsf{CC}_K(0 \,\|\, N)$ を計算し，その上位 128 ビットを R，次の 128 ビットを S として，Poly のワンタイム鍵 R と擬似ワンタイムパッド S を生成する．
  - 暗号化アルゴリズムと同様の手順で AD A と暗号文 C からメッセージ Y を構成する．次に $T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$ としてタグ T を生成する．ただし，R の特定ビットは 0 とする．入力されたタグとこれが一致すれば $\top$ を出力し，そうでなければ $\bot$ を出力する．

  検証アルゴリズムはチェックのみを行い，平文の計算は行わない．

CC-Poly $= (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ の擬似コードを図 7 に示す．また，内部で用いるサブルーチンのは図 4，図 5 にある．ChaCha20 ブロック関数 CC の仕様は付録 Aにある．

Algorithm $\mathsf{Enc}_K(N, A, M)$

1. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(M))$
2. $C \leftarrow M \oplus Z$
3. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
4. **return** (C, T)

Algorithm $\mathsf{Dec}_K(N, A, C, T^*)$

1. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(C))$
2. $M \leftarrow C \oplus Z$
3. **return** M

Algorithm $\mathsf{Ver}_K(N, A, C, T^*)$

1. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
2. **if** $T \neq T^*$ **then return** $\perp$
3. **return** $\top$

図 7: Enc_K，Dec_K，Ver_K の定義

8.6.1 PA 安全性

まず ChaCha20-Poly1305 に対する PA を定義する．PA は直感的には平文を知らずに暗号文を作成できないという性質であり，復号クエリを使っても暗号化クエリで得られる以上の平文の情報を手に入れることができないことを指す．

Ext を extractor とする．これは状態を保持できるアルゴリズムであり，鍵 K を入力とせず，暗号化オラクル Enc_K や復号オラクル Dec_K にも直接アクセスしない．$\mathcal{A}$ を 2 つのオラクル $\mathcal{O}_1$ と $\mathcal{O}_2$ にアクセスする敵とする．PA1 においては，Ext は $\mathcal{A}$ が $\mathcal{O}_1$ にクエリした履歴にアクセスできるが，PA2 ではアクセスできない．また PA2 においは，$\mathcal{A}$ が $\mathcal{O}_1$ に対して (N, A, M) をクエリし (C, T) を受け取ったならば，その後 (N, A, C, T) を $\mathcal{O}_2$ にクエリしてはならない．このとき，上記の制限を満たして Ext が効率よく復号オラクル Dec_K をシミュレートできるとき，それぞれ PA1 安全，PA2 安全といい，利得を

$$\begin{cases} \mathbf{Adv}^{\mathrm{pa1},\mathsf{Ext}}_{\mathsf{CC\text{-}Poly}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{Enc}_K,\mathsf{Dec}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{Enc}_K,\mathsf{Ext}} \Rightarrow 1] \\ \mathbf{Adv}^{\mathrm{pa2},\mathsf{Ext}}_{\mathsf{CC\text{-}Poly}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{Enc}_K,\mathsf{Dec}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{Enc}_K,\mathsf{Ext}} \Rightarrow 1] \end{cases}$$

と定義する．ただし，$K \xleftarrow{\$} \mathcal{K}_{\mathsf{CC}}$ である．また，$\mathcal{O}_1$ に対してナンスを再利用してはならないが，$\mathcal{O}_2$ に対しては再利用してもよく，$\mathcal{O}_1$ と $\mathcal{O}_2$ の間でナン

スを繰り返し利用しても構わない．

Andreeva らはナンスを用いたカウンタモードが PA1 安全ではないことを示しており [ABL+14]，同じ議論で ChaCha20-Poly1305 が PA1 を満たさないことを示せる．

敵は $\mathcal{A}$ はナンス N と AD A と A'，タグ T を任意に選び，1 ブロックの任意の暗号文 C に対し，復号オラクルに対して (N, A, C, T) をクエリする．復号オラクルが Dec_K のとき，復号の式は $M = \mathsf{CC}_K(1, N) \oplus C$ と表され，$M \oplus C$ から鍵ストリーム Z（$= \mathsf{CC}_K(1, N)$）の値を得る．1 ブロックの $M' \neq M$ に対し，暗号化オラクルに対して (N, A', M) をクエリすると，暗号文が $C' = \mathsf{CC}_K(1, N) \oplus M' = Z \oplus M'$ ということがわかる．しかし，復号オラクルが extractor のとき，extractor は鍵 K を知らず，暗号化クエリや復号クエリを行うことはできない．1 回目の復号クエリにおいてナンス N に対する暗号化クエリの履歴がないので，どの extractor も鍵ストリーム Z を計算することができない．したがって，ChaCha20-Poly1305 は PA1 安全ではない．

以上から，次の定理が成り立つ．

定理 2 ChaCha20-Poly1305 は PA1 安全ではない．

PA1 安全でなければ PA2 安全でもないので，次の系を得る．

系 1 ChaCha20-Poly1305 は PA2 安全ではない．

8.6.2 INT-RUP 安全性

ChaCha20-Poly1305 に対する INT-RUP 安全性は次のように定義される．敵 $\mathcal{A}$ は，ChaCha20-Poly1305 の暗号化オラクル Enc_K，復号オラクル Dec_K，および検証オラクル Ver_K が与えられる．このとき，$\mathcal{A}$ の利得を

$$\mathbf{Adv}^{\text{int-rup}}_{\mathsf{CC\text{-}Poly}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{Enc}_K, \mathsf{Dec}_K, \mathsf{Ver}_K} \text{ forges}]$$

と定義する．ただし，$K \xleftarrow{\$} \mathcal{K}_{\mathsf{CC}}$ であり，$\mathcal{A}$ forges は Ver_K が $\mathcal{A}$ の検証クエリに対して一度でも $\top$ を返すイベントである．また，$\mathcal{A}$ が暗号化オラクルに (N, A, M) をクエリし (C, T) を受け取った後で，Ver_K に対して (N, A, C, T) をクエリしてはならない．さらに，$\mathcal{A}$ は暗号化オラクルに対してナンスを繰り返してはならない．Dec_K と Ver_K に対してはナンスを再利用してもよく，それぞれのオラクルの間でナンスを再利用してもよい．

$\mathcal{A}$ を ChaCha20-Poly1305 に対する敵であるとする．$\mathcal{A}$ は暗号化クエリを q 回，復号クエリを q' 回，検証クエリを q'' 回行うとする．暗号化クエリが

$$(N_1, A_1, M_1), \ldots, (N_q, A_q, M_q)$$

であり，復号クエリが

$$(N'_1, A'_1, C'_1, T'_1), \ldots, (N'_{q'}, A'_{q'}, C'_{q'}, T'_{q'})$$

であり，検証クエリが

$$(N''_1, A''_1, C''_1, T''_1), \ldots, (N''_{q''}, A''_{q''}, C''_{q''}, T''_{q''})$$

であったとする．このときの暗号化クエリにおける入力の総バイト長を

$$\sum_{1 \le i \le q} (\mathsf{len}(N_i) + \mathsf{len}(A_i) + \mathsf{len}(M_i))$$

とし，復号クエリにおける入力の総バイト長を

$$\sum_{1 \le i' \le q'} (\mathsf{len}(N'_{i'}) + \mathsf{len}(A'_{i'}) + \mathsf{len}(C'_{i'}) + \mathsf{len}(T'_{i'}))$$

とする．検証クエリにおける入力の総バイト長を

$$\sum_{1 \le i'' \le q''} (\mathsf{len}(N''_{i''}) + \mathsf{len}(A''_{i''}) + \mathsf{len}(C''_{i''}) + \mathsf{len}(T''_{i''}))$$

と定義する．

また，暗号化クエリにおける平文の総ブロック数を $\sum_{1 \le i \le q} \lceil \mathsf{len}(M_i)/64 \rceil$，復号クエリにおける暗号文の総ブロック数を $\sum_{1 \le i' \le q'} \lceil \mathsf{len}(C'_{i'})/64 \rceil$ と定義する．さらに，暗号化クエリと検証クエリに対するメッセージの最大バイト長を $\max\{\mathsf{len}(Y_1), \ldots, \mathsf{len}(Y_q), \mathsf{len}(Y''_1), \ldots, \mathsf{len}(Y''_{q''})\}$ と定義する．ただし，Y_i は暗号化クエリにおける Poly1305 への入力メッセージを表し，$Y''_{i''}$ は検証クエリにおける Poly1305 への入力メッセージを表す．

このとき，以下の定理が成り立つ．

定理 3 $\mathcal{A}$ を暗号化オラクルに高々q 回，復号オラクルに高々q' 回，検証オラクルに高々q'' 回アクセスする敵とする．それぞれのクエリの入力の総バイト長が高々σ バイト，高々σ' バイト，高々σ'' バイトであり，暗号化クエリの平文の総ブロック数が高々μ_M ブロック，復号クエリの暗号文の総ブロック数が高々$\mu'_{C'}$ ブロックであり，さらに暗号化クエリと検証クエリに対するメッセージの最大バイト長が高々$\ell_{\max}$ バイトであるとする．このとき，高々$\mu_M + q + \mu'_{C'} + q''$ 回オラクルにクエリする CC に対する敵 $\mathcal{B}$ が存在し，

$$\mathbf{Adv}^{\text{int-rup}}_{\text{CC-Poly}}(\mathcal{A}) \le \mathbf{Adv}^{\text{prf}}_{\mathsf{CC}}(\mathcal{B}) + q'' \frac{8\lceil \ell_{\max}/16 \rceil}{2^{106}} \tag{29}$$

が成り立つ．また，$\mathcal{B}$ の実行時間は $\mathcal{A}$ の実行時間と $O(\sigma + \sigma' + \sigma'')$ である．

以下，証明の概略を示す．一般性を失うことなく，$\mathcal{A}$ は確定的アルゴリズムであり，それぞれのオラクルに q 回，q' 回，q'' 回アクセスするとする．証

明は game-playing technique [BR06] に従う．図 8 にゲーム 0 とゲーム 1 の定義を示す．ゲーム 0 は枠内の処理を含み，ゲーム 1 は含まない．ゲーム 0 とゲーム 1 はフラグ forge が true にならない限り同一であるから，fundamental lemma of game playing [BR06] より，

$$\mathbf{Adv}^{\text{int-rup}}_{\text{CC-Poly}}(\mathcal{A}) \leq \Pr[\mathcal{A}^{\text{ゲーム } 1} \text{ sets } \mathsf{forge}] \tag{30}$$

が成り立つ．

次に，CC_K をランダム関数 $F \xleftarrow{\$} \{f \mid f : \{0,1\}^{32} \times \{0,1\}^{96} \to \{0,1\}^{512}\}$ に置き換えたゲームをゲーム 2 とする．ゲーム 2 を図 9 に示す．このとき，定理にある CC に対する敵 $\mathcal{B}$ が存在し，

$$\Pr[\mathcal{A}^{\text{ゲーム } 1} \text{ sets } \mathsf{forge}] - \Pr[\mathcal{A}^{\text{ゲーム } 2} \text{ sets } \mathsf{forge}] \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{CC}}(\mathcal{B}) \tag{31}$$

が成り立つ．

ここでゲーム 2 において，F の入力 $(0, N)$ はタグ生成のみに用いられ KGen_F では使用されないため，Enc_F が暗号文 C の生成に利用する乱数と Dec_F が使用する乱数は，Enc_F がタグ T の生成に使用する乱数と Ver_F が使用する乱数とは独立である．とくに，Dec_F はフラグ forge を true とすることと独立である．

KGen_F を $\mathcal{A}$ がシミュレートすると捉えれば，forge を true することは

$$(N, A, C) \mapsto T$$

によって定まるメッセージ認証コードに対して，q 回のタグ生成オラクルと q'' 回のタグ検証オラクルへのアクセスにより偽造をすることに他ならない．このときの偽装成功確率の上界は Procter の結果 [Pro14] より

$$\Pr[\mathcal{A}^{\text{ゲーム } 2} \text{ sets } \mathsf{forge}] \leq q'' \frac{8\lceil \ell_{\max}/16 \rceil}{2^{106}} \tag{32}$$

となる．

式 (30)，式 (31)，式 (32) より，式 (29) を得る．

8.7 複数ユーザ安全性

CC-Poly の複数ユーザ安全性を解析した文献はないが，GCM についての解析が知られている [LMP17]．

ChaCha20-Poly1305 ではユーザ数が U であるとき，式 (16) より敵の攻撃成功確率に関する自明な安全性バウンド

$$\mathbf{Adv}^{\text{mu-AE}}_{\text{CC-Poly}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{mu-prf}}_{\mathsf{CC}}(\mathcal{B}) + Uq'' \frac{8\lceil \ell_{\max}/16 \rceil}{2^{106}} \tag{33}$$

Algorithm Initialize

1. $K \xleftarrow{\$} \{0,1\}^{256}$
2. $\mathsf{forge} \leftarrow \mathsf{false}$

Algorithm $\mathsf{Enc}_K(N, A, M)$

1. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(M))$
2. $C \leftarrow M \oplus Z$
3. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
4. **return** (C, T)

Algorithm $\mathsf{Dec}_K(N, A, C, T^*)$

1. $Z \leftarrow \mathsf{KGen}_K(N, \mathsf{len}(C))$
2. $M \leftarrow C \oplus Z$
3. **return** M

Algorithm $\mathsf{Ver}_K(N, A, C, T^*)$

1. $T \leftarrow \mathsf{Tag}_K(N, A, C)$
2. **if** $T = T^*$ **then** $\mathsf{forge} \leftarrow \mathsf{true}$; $\boxed{\textbf{return } \top}$
3. **return** $\bot$

Algorithm $\mathsf{KGen}_K(N, \ell)$

1. $m \leftarrow \lceil \ell/64 \rceil$
2. **for** $i = 1$ **to** m **do**
3. $\quad Z[i] \leftarrow \mathsf{CC}_K(i, N)$
4. $Z \leftarrow$ first ℓ bytes of $Z[1] \,\|\, \cdots \,\|\, Z[m]$
5. **return** Z

Algorithm $\mathsf{Tag}_K(N, A, C)$

1. $(R, S) \xleftarrow{16}$ first 32 bytes of $\mathsf{CC}_K(0, N)$
2. $R \leftarrow R \wedge \texttt{0x0fffffffc0fffffffc0fffffffc0ffffffff}$
3. $Y \leftarrow A \,\|\, \mathsf{pad}(A) \,\|\, C \,\|\, \mathsf{pad}(C) \,\|\, \mathsf{pad}(A, C)$
4. $T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$
5. **return** T

Algorithm Finalize

1. **return** forge

図 8: ゲーム 0 とゲーム 1 の定義

Algorithm Initialize

1. $F \xleftarrow{\$} \{f \mid f : \{0,1\}^{32} \times \{0,1\}^{96} \to \{0,1\}^{512}\}$
2. $\mathsf{forge} \leftarrow \mathsf{false}$

Algorithm $\mathsf{Enc}_F(N, A, M)$

1. $Z \leftarrow \mathsf{KGen}_F(N, \mathsf{len}(M))$
2. $C \leftarrow M \oplus Z$
3. $T \leftarrow \mathsf{Tag}_F(N, A, C)$
4. **return** (C, T)

Algorithm $\mathsf{Dec}_F(N, A, C, T^*)$

1. $Z \leftarrow \mathsf{KGen}_F(N, \mathsf{len}(C))$
2. $M \leftarrow C \oplus Z$
3. **return** M

Algorithm $\mathsf{Ver}_F(N, A, C, T^*)$

1. $T \leftarrow \mathsf{Tag}_F(N, A, C)$
2. **if** $T = T^*$ **then** $\mathsf{forge} \leftarrow \mathsf{true}$
3. **return** $\perp$

Algorithm $\mathsf{KGen}_F(N, \ell)$

1. $m \leftarrow \lceil \ell/64 \rceil$
2. **for** $i = 1$ **to** m **do**
3. $\quad Z[i] \leftarrow F(i, N)$
4. $Z \leftarrow$ first ℓ bytes of $Z[1] \,\|\, \cdots \,\|\, Z[m]$
5. **return** Z

Algorithm $\mathsf{Tag}_F(N, A, C)$

1. $(R, S) \xleftarrow{16}$ first 32 bytes of $F(0, N)$
2. $R \leftarrow R \wedge \texttt{0x0fffffffc0fffffffc0fffffffc0ffffffff}$
3. $Y \leftarrow A \,\|\, \mathsf{pad}(A) \,\|\, C \,\|\, \mathsf{pad}(C) \,\|\, \mathsf{pad}(A, C)$
4. $T \leftarrow \mathsf{Poly}_R(Y) + S \bmod 2^{128}$
5. **return** T

Algorithm Finalize

1. **return** forge

図 9: ゲーム 2 の定義

を得る．式 (33) は CC-Poly の複数ユーザにおける安全性を示しており，敵の攻撃成功確率の上界を導出できるという点で，ChaCha20-Poly1305 は複数ユーザ安全性の意味で証明可能安全性を有している．

式 (33) において第一項の $\mathbf{Adv}^{\text{mu-prf}}_{\mathsf{CC}}(\mathcal{B})$ は計算量的な安全性を示しており，[Bih02, BMS05] にある結果からこの項はこれ以上改善できず厳密である．一方で，情報理論的な項

$$Uq''\frac{8\lceil \ell_{\max}/16\rceil}{2^{106}}$$

は，[LMP17] と同様の解析を行うことにより，ユーザ数に依存せず，

$$q''\frac{8\lceil \ell_{\max}/16\rceil}{2^{106}}$$

まで改善できると予想される．全体の安全性バウントとしては $\mathbf{Adv}^{\text{prf}}_{\mathsf{CC}}(\mathcal{B})$ のみにユーザ数 U がかかり，それ以外への影響がなく，

$$U\mathbf{Adv}^{\text{prf}}_{\mathsf{CC}}(\mathcal{B}) + q''\frac{8\lceil \ell_{\max}/16\rceil}{2^{106}}$$

を安全性バウンドとして示せるものと予想される．

9 まとめ

認証暗号化方式 ChaCha20-Poly1305 およびメッセージ認証コード Poly1305 の安全性調査・評価を報告した．

Poly1305 について，Bernstein による安全性証明 [Ber05b, Ber05a] に問題点がないことを確認した．また文献調査により，弱鍵が存在すること，再偽造可能性の意味で耐性があること，ナンスを再利用すると多項式ハッシュ関数の鍵を導出でき，偽造攻撃ができることを確認した．

安全性評価として，関連鍵攻撃が可能であることと，複数ユーザ安全性についてユーザ数への依存が限定的な安全性バウンドを導出できることを示した．

また ChaCha20-Poly1305 について，Procter による証明 [Pro14] を修正することにより，ChaCha20-Poly1305 の証明可能安全性バウンドを導出できることを確認した．

安全性評価として，ChaCha20 ブロック関数が関連鍵攻撃に対して安全であれば ChaCha20-Poly1305 が関連鍵攻撃に対する安全性を有すること，再偽造可能性の意味で耐性があること，ナンスを再利用すると偽造攻撃ができること，弱鍵が存在することを示した．また，復号ミスユース耐性について，暗号化の意味での安全性は失われるものの，認証の意味での安全性は保たれること，複数ユーザ安全性について，ユーザ数に依存した自明な安全性バウンドを導出できることを示した．

上記のうち，ナンスの再利用による安全性への影響は大きいことは懸念事項であり，注意して実装されなくてはならない．弱鍵の影響は少ないと考え

られ，また Poly1305 の関連鍵攻撃が現実的に問題となることはないと考えられる．その他，懸念事項は見受けられない．

謝辞

本報告書は名古屋大学大学院工学研究科計算理工学専攻，今村和弥君の協力を得て作成した．

参考文献

[ABBT15] Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted Polynomials and Forgery Attacks on GCM. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 762–786. Springer, 2015.

[ABL+14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting Authenticated Encryption Robustness with Minimal Modifications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[BBT16] Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-Function Based PRFs: AMAC and Its Multi-User Security. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances*

in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I, volume 9665 of *Lecture Notes in Computer Science*, pages 566–595. Springer, 2016.

[BC09] John Black and Martin Cochran. MAC Reforgeability. In Orr Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*, pages 345–362. Springer, 2009.

[Ber05a] Daniel J. Bernstein. Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2005.

[Ber05b] Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.

[Ber08] Daniel. J. Bernstein. ChaCha, a variant of Salsa20. Workshop Record of SASC 2008: The State of the Art of Stream Ciphers, 2008. `http://cr.yp.to/papers.html#chacha`.

[Bih02] Eli Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

[BMS05] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved Time-Memory Trade-Offs with Multiple Data. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston,*

ON, Canada, August 11-12, 2005, Revised Selected Papers, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer, 2005.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[BR02] John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.

[BR06] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[CMS11] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another Look at Tightness. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 293–319. Springer, 2011.

[CW79] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[DFK+17] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella Béguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. Implementing and Proving the TLS 1.3 Record Layer. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 463–482. IEEE Computer Society, 2017.

[FJM14] Pierre-Alain Fouque, Antoine Joux, and Chrysanthi Mavromati. Multi-user Collisions: Applications to Discrete Logarithm, Even-Mansour and PRINCE. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 420–438. Springer, 2014.

[FLLW17] Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Reforgeability of Authenticated Encryption Schemes. In Josef Pieprzyk and Suriadi Suriadi, editors, *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, volume 10343 of *Lecture Notes in Computer Science*, pages 19–37. Springer, 2017.

[HP08] Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008.

[HT16] Viet Tung Hoang and Stefano Tessaro. Key-Alternating Ciphers and Key-Length Extension: Exact Bounds and Multi-user Security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2016.

[HT17] Viet Tung Hoang and Stefano Tessaro. The Multi-user Security of Double Encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 381–411, 2017.

[II16] 今村 和弥, 岩田 哲. ChaCha20-Poly1305 の Nonce-Misuse および Decryption-Misuse 耐性. 2016 年暗号と情報セキュリティシンポジウム，SCIS 2016, 2016.

[Ima17] 今村 和弥. ストリーム暗号とユニバーサルハッシュ関数に基づく認証暗号化方式の安全性に関する研究. 名古屋大学工学研究科計算理工学専攻修士論文, 2017.

[IMI16] Kazuya Imamura, Kazuhiko Minematsu, and Tetsu Iwata. Integrity Analysis of Authenticated Encryption Based on Stream Ciphers. In Liqun Chen and Jinguang Han, editors, *Provable Security - 10th International Conference, ProvSec 2016, Nanjing, China, November 10-11, 2016, Proceedings*, volume 10005 of *Lecture Notes in Computer Science*, pages 257–276, 2016.

[IMI17] Kazuya Imamura, Kazuhiko Minematsu, and Tetsu Iwata. Integrity analysis of authenticated encryption based on stream ciphers. *International Journal of Information Security*, Jun 2017.

[Kra94] Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994.

[LCM+16] Adam Langley, Wan-Teh Chang, Nikos Mavrogiannopoulos, Joachim Strombergson, and Simon Josefsson. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). RFC 7905, 2016. `https://tools.ietf.org/html/rfc7905`.

[LMP17] Atul Luykx, Bart Mennink, and Kenneth G. Paterson. Analyzing Multi-key Security Degradation. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 575–605. Springer, 2017.

[ML15] Nicky Mouha and Atul Luykx. Multi-key Security: The Even-Mansour Construction Revisited. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO*

2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I, volume 9215 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2015.

[NL15] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539, 2015. `https://tools.ietf.org/html/rfc7539`.

[PC13] Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 287–304. Springer, 2013.

[PC15] Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes. *J. Cryptology*, 28(4):769–795, 2015.

[Pro14] Gordon Procter. A Security Analysis of the Composition of ChaCha20 and Poly1305. *IACR Cryptology ePrint Archive*, 2014:613, 2014.

[RBB03] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[Saa12] Markku-Juhani Olavi Saarinen. Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.

[Sho96] Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328. Springer, 1996.

[ST16] Thomas Shrimpton and R. Seth Terashima. Salvaging Weak Security Bounds for Blockcipher-Based Constructions. In

Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 429–454, 2016.

[Sti95] Douglas R. Stinson. On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(52), 1995.

[WLW06a] Dayin Wang, Dongdai Lin, and Wenling Wu. An Improved Poly1305 MAC. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 284–292, 2006.

[WLW06b] Dayin Wang, Dongdai Lin, and Wenling Wu. OPMAC: One-Key Poly1305 MAC. In Helger Lipmaa, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology, Second SKLOIS Conference, Inscrypt 2006, Beijing, China, November 29 - December 1, 2006, Proceedings*, volume 4318 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 2006.

[WLZZ16] Peng Wang, Yuling Li, Liting Zhang, and Kaiyan Zheng. Related-Key Almost Universal Hash Functions: Definitions, Constructions and Applications. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2016.

A ChaCha20 ブロック関数の仕様

CC-Poly では内部で ChaCha20 のブロック関数 CC を用いる [Ber08]．CC の入出力は

$$\mathsf{CC} : \mathcal{K}_{\mathsf{CC}} \times \{0,1\}^{32} \times \{0,1\}^{96} \to \{0,1\}^{512}$$

で与えられる．$\mathcal{K}_{\mathsf{CC}} = \{0,1\}^{256}$ は鍵空間である．入力として 256 ビット（32 バイト）の秘密鍵 K，32 ビット（4 バイト）のカウンタ ctr，および 96 ビッ

ト（12 バイト）のナンス N をとり，512 ビット（64 バイト）の鍵ストリーム Z を出力する．

状態配列　状態配列は 512 ビット（64 バイト）からなり，これを 32 ビット（4 バイト）ごとに区切り，行列

$$X = \begin{bmatrix} X[0] & X[1] & X[2] & X[3] \\ X[4] & X[5] & X[6] & X[7] \\ X[8] & X[9] & X[10] & X[11] \\ X[12] & X[13] & X[14] & X[15] \end{bmatrix}$$

として表現する．入力である鍵 K とナンス N を $(K[0], \ldots, K[7]) \xleftarrow{4} K$ および $(N[0], N[1], N[2]) \xleftarrow{4} N$ と 4 バイトごとに分割し，状態配列を

$$X = \begin{bmatrix} \mathsf{Con}[0] & \mathsf{Con}[0] & \mathsf{Con}[2] & \mathsf{Con}[3] \\ K[0] & K[1] & K[2] & K[3] \\ K[4] & K[5] & K[6] & K[7] \\ \mathsf{ctr} & N[0] & N[1] & N[2] \end{bmatrix} \tag{34}$$

と初期化する．ただし，$\mathsf{Con}[i]$ は定数であり，$\mathsf{Con}[0] = \mathtt{0x61707865}$，$\mathsf{Con}[1] = \mathtt{0x3320646e}$，$\mathsf{Con}[2] = \mathtt{0x79622d32}$，$\mathsf{Con}[3] = \mathtt{0x6b206574}$ である．

ラウンド処理　ラウンド処理は 1/4 ラウンド関数 QuaterRound を基本とする，全部で 20 ラウンドからなる処理である．QuaterRound の定義を図 10 に示す．また，図 11 に動作の流れを示す．ただし，$X \lll n$ はバイト列 X を n ビット左に巡回シフトしたバイト列を表す．

各ラウンドでは，奇数ラウンドにおいて列成分に QuaterRound を適用し，偶数ラウンドでは対角成分に QuaterRound を適用する．$\mathsf{CC}_K(\mathsf{ctr}, N)$ の定義を図 12 に示す．

Algorithm QuaterRound(a, b, c, d)

1. $a \leftarrow a + b \bmod 2^{32}$
2. $d \leftarrow d \oplus a$
3. $d \leftarrow d \lll 16$
4. $c \leftarrow c + d \bmod 2^{32}$
5. $b \leftarrow b \oplus c$
6. $b \leftarrow b \lll 12$
7. $a \leftarrow a + b \bmod 2^{32}$
8. $d \leftarrow d \oplus a$
9. $d \leftarrow d \lll 8$
10. $c \leftarrow c + d \bmod 2^{32}$
11. $b \leftarrow b \oplus c$
12. $b \leftarrow b \lll 7$

図 10: QuaterRound(a, b, c, d) の定義

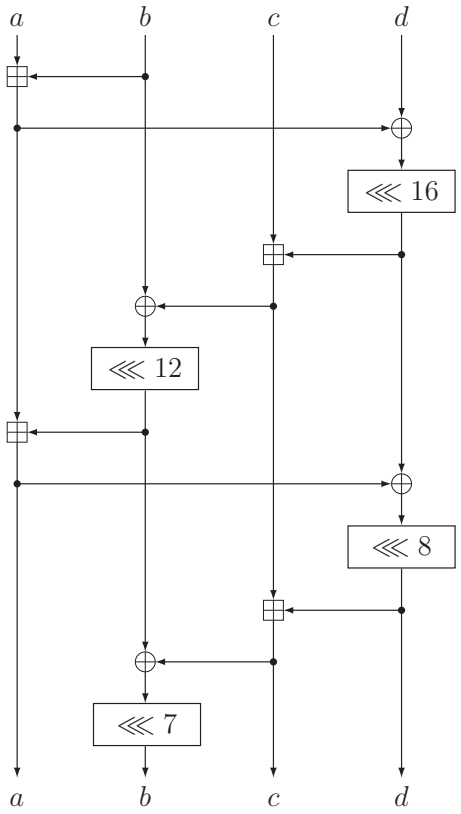

図 11: QuaterRound(a, b, c, d) の動作

Algorithm $\mathsf{CC}_K(\mathsf{ctr}, N)$

1. $X_{\text{init}} \leftarrow$ 式 (34) の行列
2. $X \leftarrow X_{\text{init}}$
3. **for** $i = 1$ **to** 10 **do**
4. $\quad \mathsf{QuaterRound}(X[0], X[4], X[8], X[12])$
5. $\quad \mathsf{QuaterRound}(X[1], X[5], X[9], X[13]))$
6. $\quad \mathsf{QuaterRound}(X[2], X[6], X[10], X[14]))$
7. $\quad \mathsf{QuaterRound}(X[3], X[7], X[11], X[15])$
8. $\quad \mathsf{QuaterRound}(X[0], X[5], X[10], X[15])$
9. $\quad \mathsf{QuaterRound}(X[1], X[6], X[11], X[12])$
10. $\quad \mathsf{QuaterRound}(X[2], X[7], X[8], X[13])$
11. $\quad \mathsf{QuaterRound}(X[3], X[4], X[9], X[14])$
12. $X \leftarrow X + X_{\text{init}} \bmod 2^{32}$ // 行列の成分ごとの和
13. $Z \leftarrow X[0] \| \cdots \| X[15]$
14. **return** Z

図 12: $Z \leftarrow \mathsf{CC}_K(\mathsf{ctr}, N)$ の定義