

# Bitcoin で CTF

Bitcoin based CTF

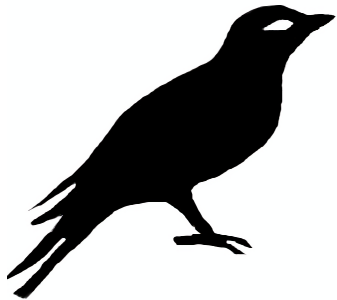
吉村 優

Hikaru YOSHIMURA

株式会社ドワンゴ  
[yyu@mental.poker](mailto:yyu@mental.poker)

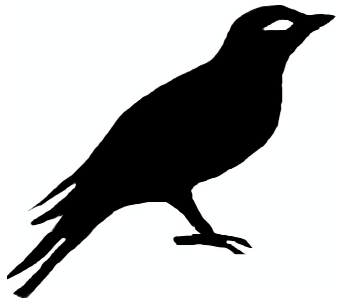
November 20, 2017  
(Git Commit ID: 8dfde9e)

# 自己紹介



Twitter    @\_yyu\_  
Qiita       yyu  
GitHub     y-yu

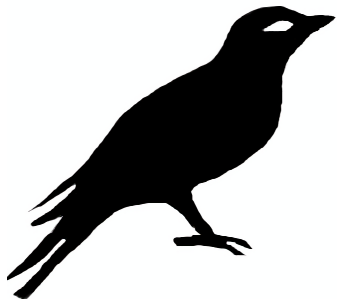
# 自己紹介



- 筑波大学 情報科学類卒（学士）

Twitter    @\_yyu\_  
Qiita       yyu  
GitHub     y-yu

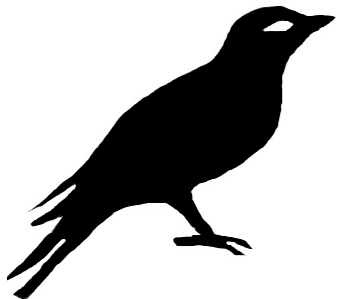
# 自己紹介



- 筑波大学 情報科学類卒（学士）
- 株式会社ドワンゴ 入社

Twitter    @\_yyu\_  
Qiita       yyu  
GitHub     y-yu

# 自己紹介



- 筑波大学 情報科学類卒（学士）
- 株式会社ドワンゴ 入社
- 第二サービス開発本部  
Dwango Cloud Service 部  
認証基盤セクション

Twitter    @\_yyu\_  
Qiita       yyu  
GitHub     y-yu

# CTF とは？

# CTF とは？

- “Capture The Flag” の略でセキュリティ系の競技のこと

# CTF とは？

- “Capture The Flag” の略でセキュリティ系の競技のこと
- いくつかの形式があるがここでは *jeopardy* 形式について説明する



# CTF とは？

- “Capture The Flag” の略でセキュリティ系の競技のこと
- いくつかの形式があるがここでは *jeopardy* 形式について説明する

## CTF (jeopardy)

- 脆弱性を攻撃するなどして**フラッグワード**と呼ばれる文字列を得る
- フラッグワードによって相応のポイントが得られ、最終的に最もポイントを獲得したチームが勝利する

# CTF とは？

- “Capture The Flag” の略でセキュリティ系の競技のこと
- いくつかの形式があるがここでは *jeopardy* 形式について説明する

## CTF (jeopardy)

- 脆弱性を攻撃するなどして**フラッグワード**と呼ばれる文字列を得る
- フラッグワードによって相応のポイントが得られ、最終的に最もポイントを獲得したチームが勝利する

- 国際的な CTF は賞金がもらえる

Google CTF 2017 (Final) 1位に 13,337 USD (約 150 万円)

Codegate CTF Finals 2017 1位に 30,000,000 KRW (約 307 万円)

# 従来の CTF の課題

# 従来の CTF の課題

- 勝利したチームが賞金を本当に得られるのかが不明である
  - ▶ たとえば脆弱性の情報が欲しいので CTF を開催したとか

# 従来の CTF の課題

- 勝利したチームが賞金を本当に得られるのかが不明である
  - ▶ たとえば脆弱性の情報が欲しいので CTF を開催したとか
- 問題を解いたチームが時間内に回答したという証拠がない
  - ▶ Write-up を書くという手もあるが、CTF が終わってから解いたという可能性もある

# 提案する CTF の特徴

# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金が得られる

# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金が得られる
  - ▶ 従来の CTF ではポイントの多い順に賞金が決まるが、提案する CTF では賞金が多い順に順位が決まる



# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金が得られる
  - ▶ 従来の CTF ではポイントの多い順に賞金が決まるが、提案する CTF では賞金が多い順に順位が決まる
- 賞金は全て Bitcoin で支払われる

# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金が得られる
  - ▶ 従来の CTF ではポイントの多い順に賞金が決まるが、提案する CTF では賞金が多い順に順位が決まる
- 賞金は全て Bitcoin で支払われる
- 基本的に問題を最初に解答したチーム以外には賞金が支払われない
  - ▶ ただし、改良することで問題の解答順序に関わらず賞金を与えることができる

# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金を得られる
  - ▶ 従来の CTF ではポイントの多い順に賞金が決まるが、提案する CTF では賞金が多い順に順位が決まる
- 賞金は全て Bitcoin で支払われる
- 基本的に問題を最初に解答したチーム以外には賞金が支払われない
  - ▶ ただし、改良することで問題の解答順序に関わらず賞金を与えることができる
- 誰もが閲覧できる Bitcoin のブロックチェーンを利用するので、どのチームが問題を時間内に解答したかが誰にとっても明らかである

# 提案する CTF の特徴

- 正しいフラッグワードを提出した場合、参加者は直ちにその問題に対応する賞金を得られる
  - ▶ 従来の CTF ではポイントの多い順に賞金が決まるが、提案する CTF では賞金が多い順に順位が決まる
- 賞金は全て Bitcoin で支払われる
- 基本的に問題を最初に解答したチーム以外には賞金が支払われない
  - ▶ ただし、改良することで問題の解答順序に関わらず賞金を与えることができる
- 誰もが閲覧できる Bitcoin のブロックチェーンを利用するので、どのチームが問題を時間内に解答したかが誰にとっても明らかである
- 従来の CTF とは異なり開始時刻や終了時刻が JST などではなく、Bitcoin のブロックチェーンの長さが  $n$  に達した時に開始であり、 $m$  ( $m > n$ ) に達した時に終了である

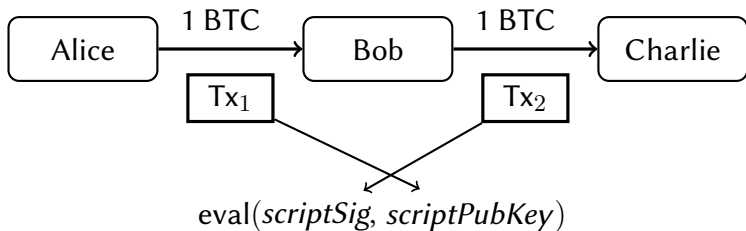
# Bitcoin とスクリプト

# Bitcoin とスクリプト

- Bitcoin のトランザクションには *scriptSig* と *scriptPubKey* という2つの場所に**スクリプト**と呼ばれる非チューリング完全なプログラムを書き込める

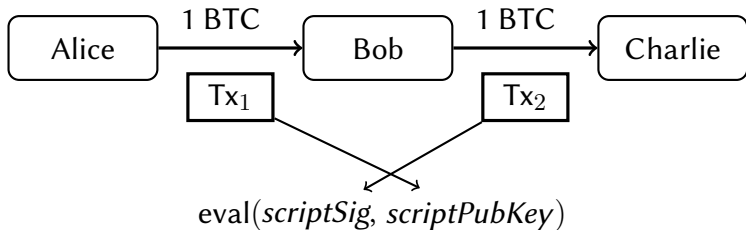
# Bitcoin とスクリプト

- Bitcoin のトランザクションには *scriptSig* と *scriptPubKey* という2つの場所に**スクリプト**と呼ばれる非チューリング完全なプログラムを書き込める
- マイナーは次のようにトランザクションのスクリプトを実行する



# Bitcoin とスクリプト

- Bitcoin のトランザクションには *scriptSig* と *scriptPubKey* という2つの場所に**スクリプト**と呼ばれる非チューリング完全なプログラムを書き込める
- マイナーは次のようにトランザクションのスクリプトを実行する



- マイナーは  $eval$  の結果が  $\overline{0}$  以外ならトランザクション  $Tx_2$  を受理し、 $\overline{0}$  ならば拒否する



# CTF の開催前

# CTF の開催前

- 1 参加チーム  $T_i$  は運営に Bitcoin の公開鍵  $\mathcal{T}_i$  を提出する

# CTF の開催前

- 1 参加チーム  $T_i$  は運営に Bitcoin の公開鍵  $\mathcal{T}_i$  を提出する
- 2 運営は参加登録をブロックチェーンの長さが  $n$  となる前に締め切る

# CTF の開催前

- 1 参加チーム  $T_i$  は運営に Bitcoin の公開鍵  $\mathcal{T}_i$  を提出する
- 2 運営は参加登録をブロックチェーンの長さが  $n$  となる前に締め切る
- 3 問題  $j$  に対応するフラグワードを  $F_j$  として、またこの問題  $j$  を解答した際に得られる賞金を  $B_j$  BTC とし、さらに、  
 $ans_{ij} := H(H(F_j || i))$  とする運営は問題  $j$  に対して、後述のような *scriptPubKey* を持つ  $B_j$  BTC のトランザクション  $Tx_j$  を作成する
  - ▶  $H$  はハッシュ関数 SHA-256 であり、また  $||$  は文字列の結合である

# CTF の開催前

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  OP_DUP
  2
  OP_EQUAL
  ⋮
OP_ENDIF
OP_CHECKSIG
```

# CTF の開催前

- ④ 運営は全てのトランザクション  $Tx_j$  をブロックチェーンへ送信する
  - ▶ ただし、トランザクションをブロックチェーンへ送信する時間はあらかじめ全参加チームに告知する

# CTF の開催前

- ④ 運営は全てのトランザクション  $T_{x_j}$  をブロックチェーンへ送信する
  - ▶ ただし、トランザクションをブロックチェーンへ送信する時間はあらかじめ全参加チームに告知する
- ⑤ 運営は全てのトランザクション  $T_{x_j}$  のトランザクション ID を CTF の問題ページに記載する

# CTF の実施中



# CTF の実施中

- 1 今、チーム  $T_i$  が問題  $j$  のフラッグワード  $F_j$  を得たとする。チーム  $T_i$  はトランザクション  $Tx_j$  を入力に持ち、次のような *scriptSig* を持つトランザクション  $Tx_{ij}$  を作成する。ただし、 $h_{ij} := H(F_j || i)$  であり、 $S_i$  はチーム  $T_i$  の Bitcoin の公開鍵  $\mathcal{T}_i$  に対応する秘密鍵によって作成された署名である

Listing:  $Tx_{ij}$  の *scriptSig*

```
 $S_i$   
 $h_{ij}$   
 $i$ 
```

# CTF の実施中

- 1 今、チーム  $T_i$  が問題  $j$  のフラッグワード  $F_j$  を得たとする。  
チーム  $T_i$  はトランザクション  $Tx_j$  を入力に持ち、次のような *scriptSig* を持つトランザクション  $Tx_{ij}$  を作成する。ただし、 $h_{ij} := H(F_j || i)$  であり、 $S_i$  はチーム  $T_i$  の Bitcoin の公開鍵  $\mathcal{T}_i$  に対応する秘密鍵によって作成された署名である

Listing:  $Tx_{ij}$  の *scriptSig*

```
 $S_i$   
 $h_{ij}$   
 $i$ 
```

- 2 チーム  $T_i$  はトランザクション  $Tx_{ij}$  を Bitcoin のブロックチェーンへ送信する

# CTF の実施中

- 1 今、チーム  $T_i$  が問題  $j$  のフラッグワード  $F_j$  を得たとする。  
チーム  $T_i$  はトランザクション  $Tx_j$  を入力に持ち、次のような *scriptSig* を持つトランザクション  $Tx_{ij}$  を作成する。ただし、 $h_{ij} := H(F_j || i)$  であり、 $S_i$  はチーム  $T_i$  の Bitcoin の公開鍵  $\mathcal{T}_i$  に対応する秘密鍵によって作成された署名である

Listing:  $Tx_{ij}$  の *scriptSig*

```
 $S_i$   
 $h_{ij}$   
 $i$ 
```

- 2 チーム  $T_i$  はトランザクション  $Tx_{ij}$  を Bitcoin のブロックチェーンへ送信する
- 3 問題  $j$  がまだ解かれていないかつフラッグワードが正しい場合、チーム  $T_i$  は  $B_j$  BTC を獲得する

# CTF の終了後

# CTF の終了後

- ① 運営は Bitcoin のブロックチェーンの長さが  $m + 3$  に達したときのブロックチェーンについて、チームに対応する Bitcoin の公開鍵を用いて問題を解答することで獲得した Bitcoin の量を計測する

# CTF の終了後

- ① 運営は Bitcoin のブロックチェーンの長さが  $m + 3$  に達したときのブロックチェーンについて、チームに対応する Bitcoin の公開鍵を用いて問題を解答することで獲得した Bitcoin の量を計測する
- ② Bitcoin を獲得した量でチームの順位付けを行う

# フラッグワードが正しい場合の挙動

# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ① チーム  $T_1$  がトランザクション  $Tx_{1j}$  を送信したとする。その *scriptSig* から、スタックは 

1	$h_{1j}$	$S_1$
---	----------	-------

 となる。ここから  $Tx_j$  の *scriptPubKey* を実行する



# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- 1 チーム  $T_1$  がトランザクション  $Tx_{1j}$  を送信したとする。その *scriptSig* から、スタックは 

1	$h_{1j}$	$S_1$
---	----------	-------

 となる。ここから  $Tx_j$  の *scriptPubKey* を実行する
- 2 スタックの先頭を複製して 1 を載せる 

1	1	1	$h_{1j}$	$S_1$
---	---	---	----------	-------

# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\tau_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ① チーム  $T_1$  がトランザクション  $Tx_{1j}$  を送信したとする。その *scriptSig* から、スタックは 

1	$h_{1j}$	$S_1$
---	----------	-------

 となる。ここから  $Tx_j$  の *scriptPubKey* を実行する
- ② スタックの先頭を複製して 1 を載せる 

1	1	1	$h_{1j}$	$S_1$
---	---	---	----------	-------
- ③ スタックの先頭から 2 つを取り出し、それらを比較する（等しいので 1 が積まれる） 

1	1	$h_{1j}$	$S_1$
---	---	----------	-------

# フラグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\tau_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- 1 チーム  $T_1$  がトランザクション  $Tx_{1j}$  を送信したとする。その *scriptSig* から、スタックは 

1	$h_{1j}$	$S_1$
---	----------	-------

 となる。ここから  $Tx_j$  の *scriptPubKey* を実行する
- 2 スタックの先頭を複製して 1 を載せる 

1	1	1	$h_{1j}$	$S_1$
---	---	---	----------	-------
- 3 スタックの先頭から 2 つを取り出し、それらを比較する（等しいので 1 が積まれる） 

1	1	$h_{1j}$	$S_1$
---	---	----------	-------
- 4 スタックの先頭を取り除き、それが 1 なので OP\_IF から OP\_ELSE を実行する 

1	$h_{1j}$	$S_1$
---	----------	-------

# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

5 スタックの先頭を捨てる 

$h_{1j}$	$S_1$
----------	-------

# フラッグワードが正しい場合の挙動

Listing: Tx<sub>j</sub> の scriptPubKey

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
  ans1j
  OP_EQUALVERIFY
  T1
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ⑤ スタックの先頭を捨てる  $\overline{h_{1j} \mid S_1}$
- ⑥ スタック先頭に SHA-256 を適用し  
結果をスタックの先頭に追加する  
 $\overline{H(h_{1j}) \mid S_1}$

# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- 5 スタックの先頭を捨てる  $\overline{h_{1j} \mid S_1}$
- 6 スタック先頭に SHA-256 を適用し  
結果をスタックの先頭に追加する  
 $\overline{H(h_{1j}) \mid S_1}$
- 7  $ans_{1j}$  をスタックの先頭に追加する  
 $\overline{ans_{1j} \mid H(h_{1j}) \mid S_1}$

# フラッグワードが正しい場合の挙動

Listing: Tx<sub>j</sub> の scriptPubKey

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
  ans1j
  OP_EQUALVERIFY
  T1
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ⑤ スタックの先頭を捨てる  $\overline{h_{1j} \mid S_1}$
- ⑥ スタック先頭に SHA-256 を適用し結果をスタックの先頭に追加する  
 $\overline{H(h_{1j}) \mid S_1}$
- ⑦ ans<sub>1j</sub> をスタックの先頭に追加する  
 $\overline{ans_{1j} \mid H(h_{1j}) \mid S_1}$
- ⑧ スタックの先頭から 2 つを取り出し、それらを比較する。等しくない場合は直ちに失敗となる  $\overline{S_1}$ 
  - ▶  $h_{1j} = H(F_j \parallel 1)$
  - ▶  $ans_{1j} = H(H(F_j \parallel 1))$

# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ⑨ チーム  $T_1$  の公開鍵  $\mathcal{T}_1$  をスタックの先頭に追加する  $\boxed{\mathcal{T}_1 \mid S_1}$



# フラッグワードが正しい場合の挙動

Listing:  $Tx_j$  の *scriptPubKey*

```
OP_DUP
1
OP_EQUAL
OP_IF
  OP_DROP
  OP_SHA256
   $ans_{1j}$ 
  OP_EQUALVERIFY
   $\mathcal{T}_1$ 
OP_ELSE
  ⋮
OP_ENDIF
OP_CHECKSIG
```

- ⑨ チーム  $T_1$  の公開鍵  $\mathcal{T}_1$  をスタックの先頭に追加する  $\boxed{\mathcal{T}_1 \mid S_1}$
- ⑩ スタックの先頭にあるデータを公開鍵として、スタックの先頭から2番目にあるデータとして署名としてそれらを検証する  $\boxed{1}$

# 進んだ話題

# 進んだ話題

- ブロックチェーンの長さが  $m$  になったら運営が賞金を回収するために、トランザクションを次のようにする

Listing: 改良した  $Tx_j$  の *scriptPubKey* の一部

```
⋮  
OP_ELSE  
  OP_DROP  
   $m$   
  OP_CHECKLOCKTIMEVERIFY  
  OP_DROP  
   $\mathcal{T}_{master}$   
OP_ENDIF  
⋮  
OP_CHECKSIG
```

- ▶ ただし  $\mathcal{T}_{master}$  は運営の Bitcoin 公開鍵とする

# 進んだ話題

- 問題とチームのそれぞれごとにトランザクションを用意することで、従来の CTF のように解答順序によらずに賞金を与えることもできる
  - ▶ ただし、そうすると 1 問あたりの賞金を減らさざるを得ない

# まとめ

# まとめ

- Bitcoin を利用した CTF について考えることで、より透明で公平な CTF を構成できた

# まとめ

- Bitcoin を利用した CTF について考えることで、より透明で公平な CTF を構成できた
- Bitcoin のリファレンス実装は、提案する CTF で利用するようなトランザクションを受け付けない。よって現実的には Ethereum など別の暗号通貨を利用しなければならないかもしれない

# まとめ

- Bitcoin を利用した CTF について考えることで、より透明で公平な CTF を構成できた
- Bitcoin のリファレンス実装は、提案する CTF で利用するようなトランザクションを受け付けない。よって現実的には Ethereum など別の暗号通貨を利用しなければならないかもしれない
- 賞金が集まれば、提案する方法で CTF を実施してみたい



# 目次

- 1 自己紹介
- 2 CTF とは？
- 3 従来の CTF の課題
- 4 提案する CTF の特徴
- 5 Bitcoin とスクリプト
- 6 CTF のプロトコル
- 7 進んだ話題
- 8 まとめ

Thank you for your attention!