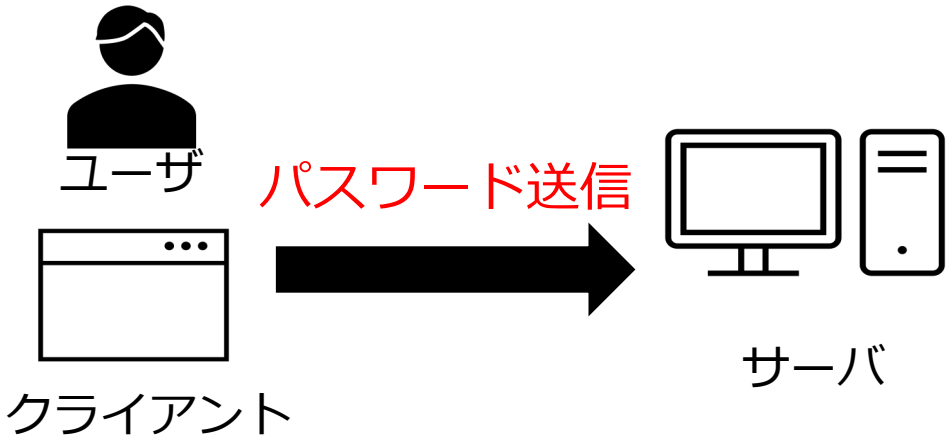
FIDO2の形式化の再考と複数モードの検証への拡張

佐藤瑞己(茨城大学)　　米山一樹(茨城大学)

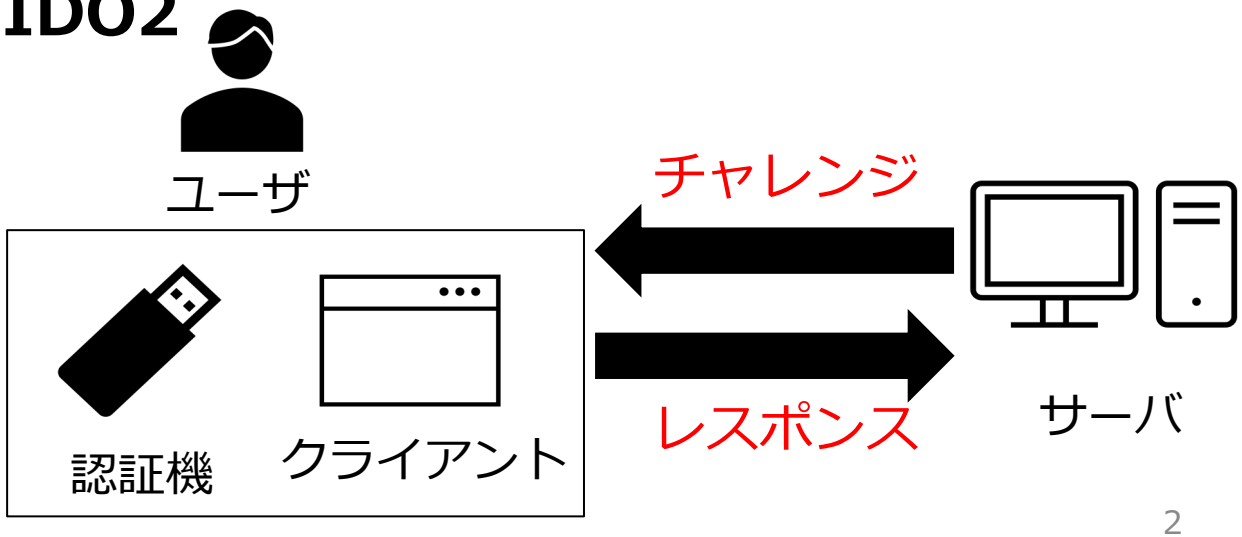
FIDO2

- FIDO Alliance, W3Cが共同で標準化を進めている認証プロトコル
- Challenge & Response方式を利用したパスワードレス認証
- Windows, iOS, Androidなどで既に実装され利用可能
- サブプロトコルのWebAuthnとCTAPからなる

パスワード認証



FIDO2



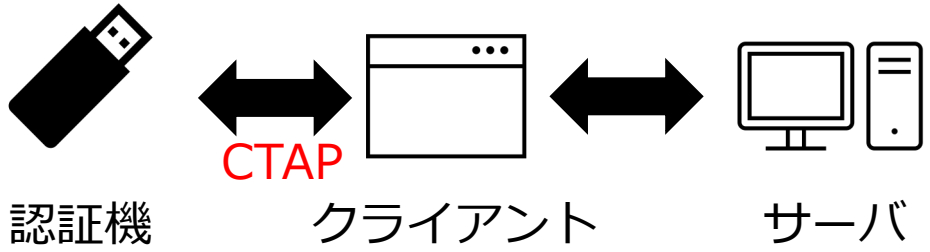
WebAuthnとCTAP

- WebAuthn：サーバによるユーザの認証
  - 登録プロトコルと認証プロトコルからなる

- CTAP：認証機-クライアント間の通信の保護
  - CTAP2が一般的に利用されているが, CTAP2.1など新しい仕様も存在

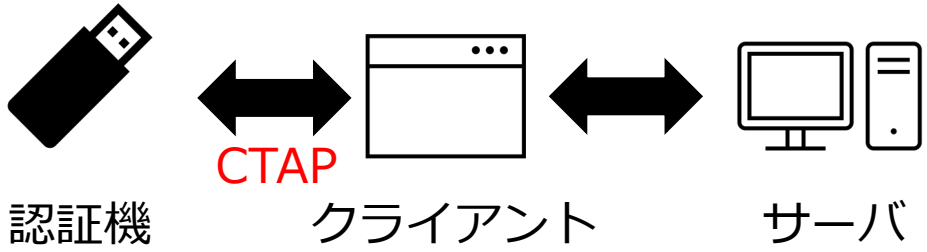
登録プロトコル

認証機をサーバに登録



認証プロトコル

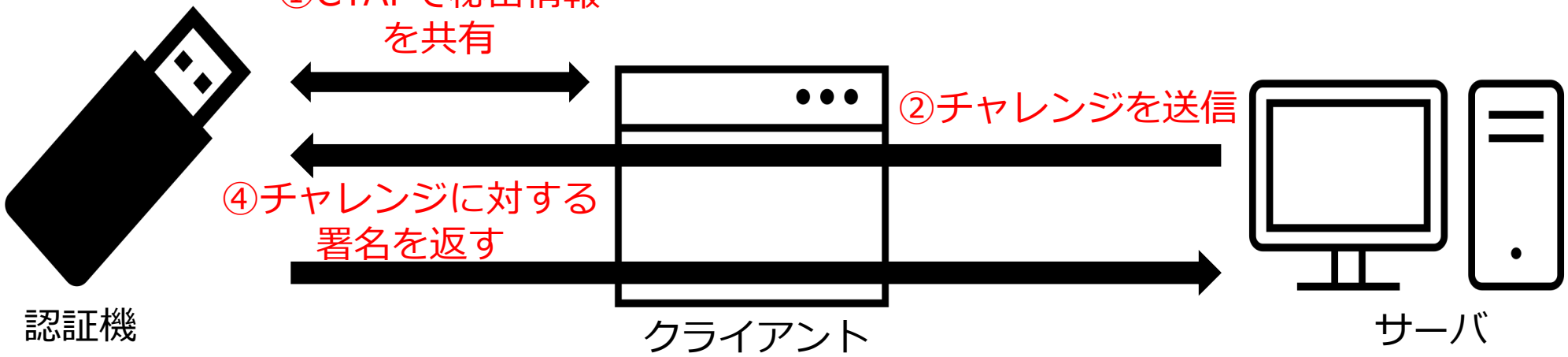
認証機を用いてユーザ認証



登録プロトコル(簡略版)

アテステーション秘密鍵 sk_{at}

アテステーション公開鍵 pk_{at}



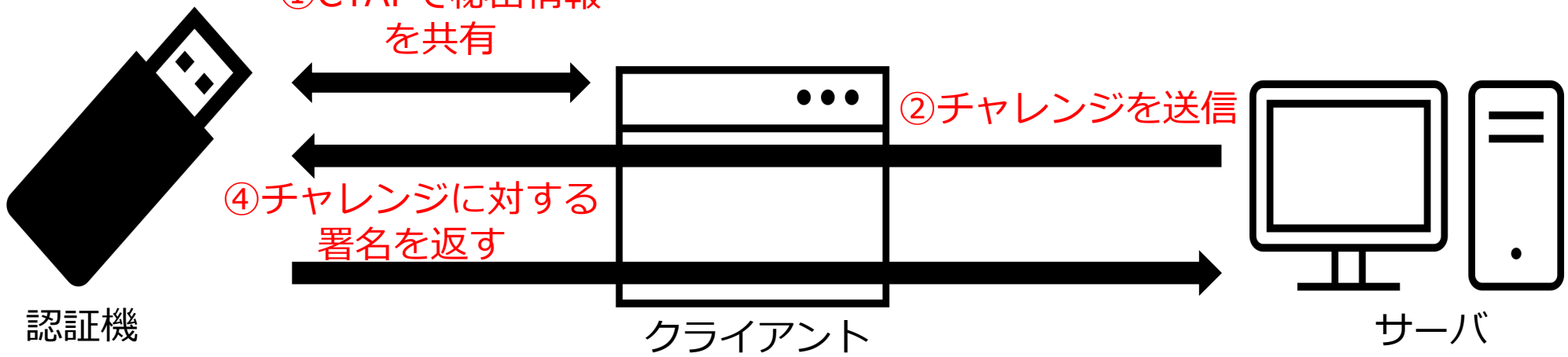
③クレデンシャル鍵ペアを作成、チャレンジに署名(sk_{at})

⑤署名を検証する

認証プロトコル(簡略版)

クレデンシャル秘密鍵 sk_{cre}

クレデンシャル公開鍵 pk_{cre}



③署名を作成(sk_{cre})

⑤署名を検証する

FIDO2の形式検証の既存研究

- GuanらによるProVerifを用いた検証 [GLYZ22]
  - WebAuthn2とCTAP2の仕様を形式化
  - 秘密情報の漏えいなどを考慮した安全性を検証
  - 検証の結果, FIDO2に対していくつかの攻撃が存在することを示した.

- [GLYZ22]の問題点
  - 形式化において, 証明書の取り扱いにミスが存在
  - FIDO2における検証していないモードや仕様が存在

⇒ プロトコル仕様に忠実かつ複数のモードにおける網羅的な検証が望ましい

[GLYZ22] Jingjing Guan, Hui Li, Haisong Ye, and Ziming Zhao. A formal analysis of the fido2 protocols. ESORICS 2022.

貢献

貢献1：形式化のミスの修正
[GLYZ22]の形式化における証明書の扱いに関するミスを修正

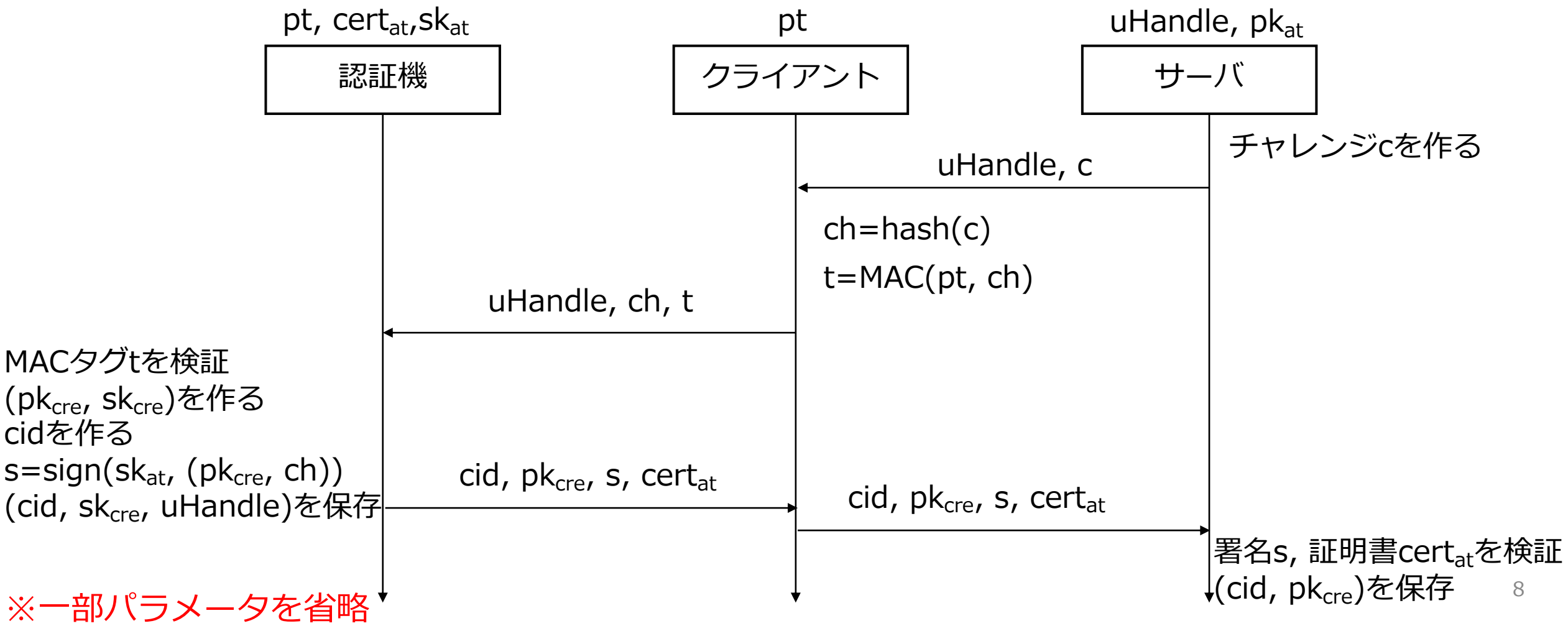
貢献2：複数モードの検証(70通り)
[GLYZ22]で形式化されていないWebAuthnの複数のモードについて形式化
CTAPについては，改良プロトコルであるsPACA [BBCW21] も形式化

結果
既存の形式化では見つからなかった攻撃を発見
攻撃に対する対策を提案

[BBCW21] Manuel Barbosa, Alexandra Boldyreva, Shan Chen, Bogdan Warinschi. Provable security analysis of fido2. CRYPTO 2021.

登録プロトコルの流れ

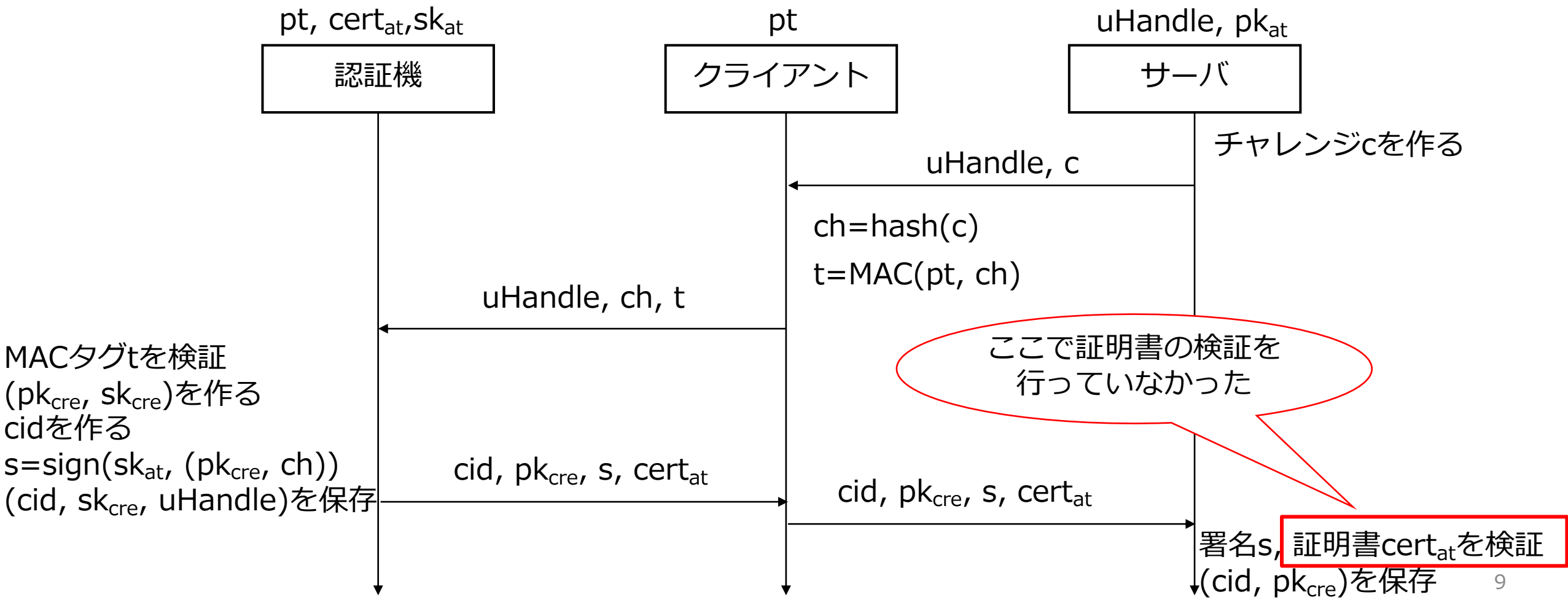
CTAPを用いて、pinToken ptを事前に認証機、クライアント間で共有しておく



※一部パラメータを省略

貢献1 - 証明書の扱いに関するミス

登録プロトコルにおいて, 証明書を検証していなかった



貢献2 - 複数モードの検証

・WebAuthnの複数モードとCTAPの異なる仕様を組み合わせた検証

	WebAuthn	CTAP
[GLYZ22]	トランザクション認証モード, ストレージモード アテステーションモード(Basic)	CTAP2
本研究	トランザクション認証モード, ストレージモード, アテステーションモード(Basic, None)	CTAP2, CTAP2.1, sPACA

- アテステーションモード(Basic)の検証において[GLYZ22]では攻撃者にアテステーション秘密鍵の漏えいを許していたが, 我々の形式化では許さない（より現実的な攻撃者を仮定）

アテステーションモード

- WebAuthnでは, 認証機が信頼できるかをサーバ側で確認できるように登録プロトコルにおいてアテステーションが可能

- しばしば用いられるモードとして, Basic, Noneモードが存在

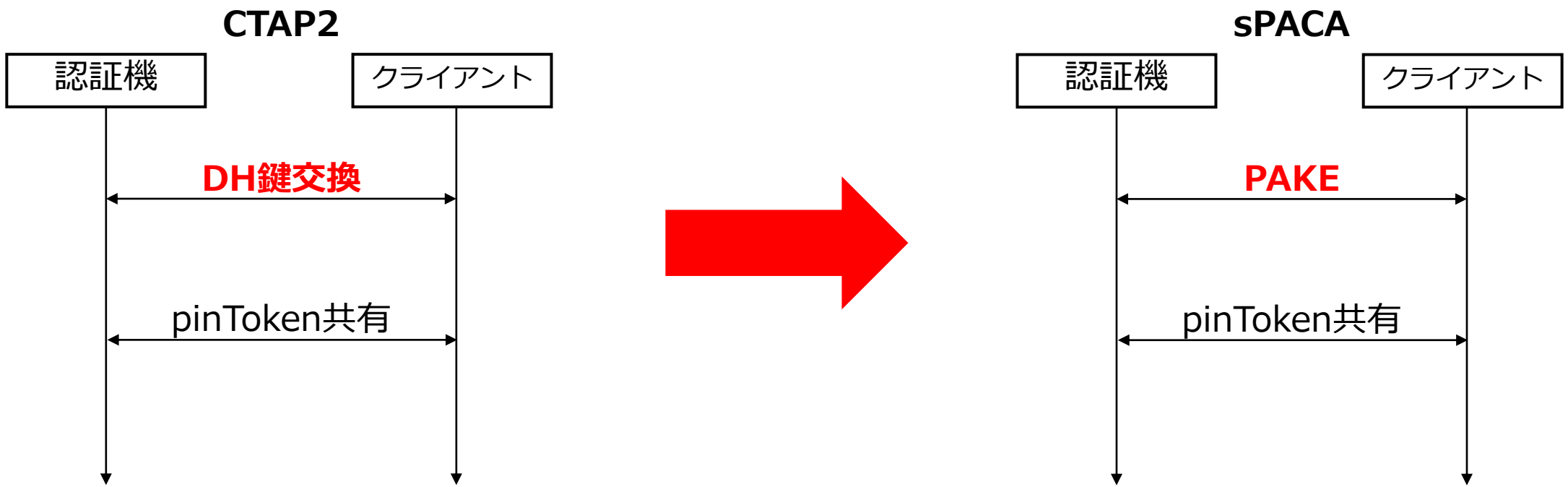
Basic	None
認証機の製造時にアテステーション鍵ペアを埋め込み, それを用いてアテステーションを行う.	アテステーションを行わない.

CTAP2.1

- 現在標準化中のプロトコル
- プロトコルの大まかな流れはCTAP2を継承
- 鍵交換の部分をpuvProtocolとして抽象化
- $\mathrm{puvProtocol}_1$と$\mathrm{puvProtocol}_2$が存在し, $\mathrm{puvProtocol}_1$はCTAP2と同等のプロトコルとなっている

sPACA [BBCW21]

- Barbosaらによって提案された, CTAP2の改良プロトコル.
- CTAP2に対する中間者攻撃を対策.
- DH鍵交換の代わりにPassword Authenticated Key Exchange（PAKE）を用いる.



プロトコルの形式化の概要

- 全部の組み合わせ(70通り)を別々に形式化するのは大変
- ProVerifのライブラリ機能を用いてWebAuthnのモードやCTAPの仕様をモジュール的に扱えるようにして効率化

プロトコルの合成部分(登録プロトコルの場合)

```
!Authenticator(…, at_type, au_type, ctap_type) |
!Client(…, at_type, ctap_type) |
!Server(…, at_type)
```

- ctap_typeでCTAPのバージョンを指定する
- au_typeでストレージモードを指定する
- at_typeでアテステーションモードを指定する
- authenticateでは更にトランザクション認証モードの指定も行う
- Authenticator, Client, Serverの動作はpvlファイルに記述

検証したい安全性

- 秘匿性
  - アテステーション秘密鍵 sk_{at}
  - クレデンシャル秘密鍵 sk_{cre}
  - クレデンシャル鍵ID cid
  - pinToken

- 認証性
  - 登録プロトコル
    1. クライアント-サーバ間
    2. 認証機-サーバ間
  - 認証プロトコル
    1. 認証機-サーバ間
    2. 漏えいを考慮した認証機-サーバ間

秘匿性の検証

- 以下のパラメータに関して, 秘匿性を検証する.
  - アテステーション秘密鍵 sk_{at}
  - クレデンシャル秘密鍵 sk_{cre}
  - クレデンシャル鍵ID cid
  - pinToken

秘匿性検証のクエリ

```
query secret skat.
query secret skcre.
query secret cid.
query secret pinToken.
```

検証結果 - 秘匿性

CTAP2, CTAP2.1を利用した場合

	検証結果
sk_{at}	✓
sk_{cre}	✓
cid	×
pinToken	×

sPACAを利用した場合

	検証結果
sk_{at}	✓
sk_{cre}	✓
cid	×
pinToken	✓

- CTAP2.1を利用した場合は, Guanらの結果と同様になった
- sPACAについて, CTAP2に存在した中間者攻撃が修正されていることを確認した
  - sPACAを用いた場合, pinTokenの秘匿性が保たれている

登録プロトコルの認証性

1.クライアント-サーバ間での認証性

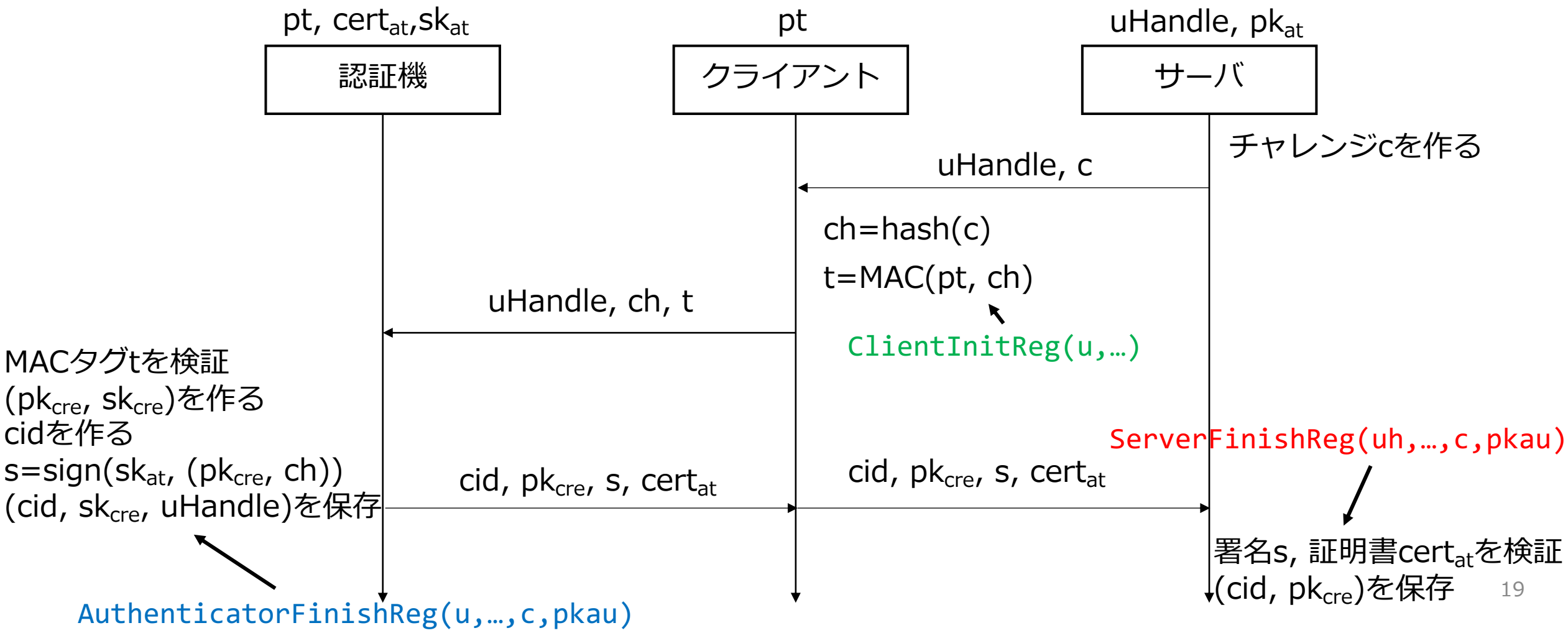
```
query uh:UserHandle, …, cid:bitstring, pkau:spkey;
inj-event(ServerFinishReg(uh,…,c,pkau)) ==> inj-event(ClientInitReg(u,…)).
```

2.認証機-サーバ間での認証性

```
query uh: UserHandle, …, cid: bitstring, pkau: spkey;
inj-event(ServerFinishReg(u,…,c,pkau)) ==>
          inj-event(AuthenticatorFinishReg(u,…,c,pkau)).
```

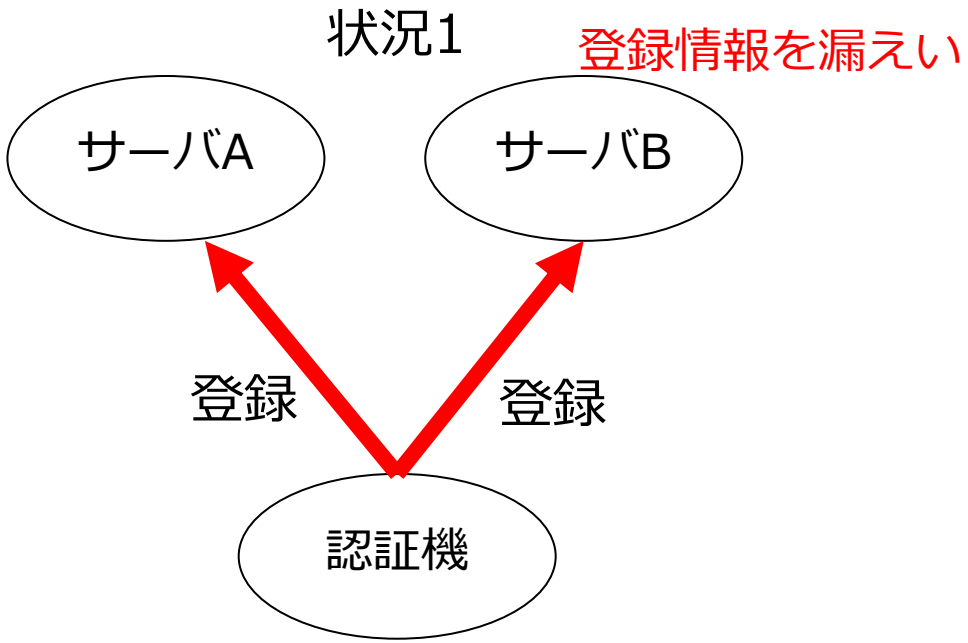
登録プロトコルの認証性

CTAPを用いて、pinToken ptを事前に認証機、クライアント間で共有しておく

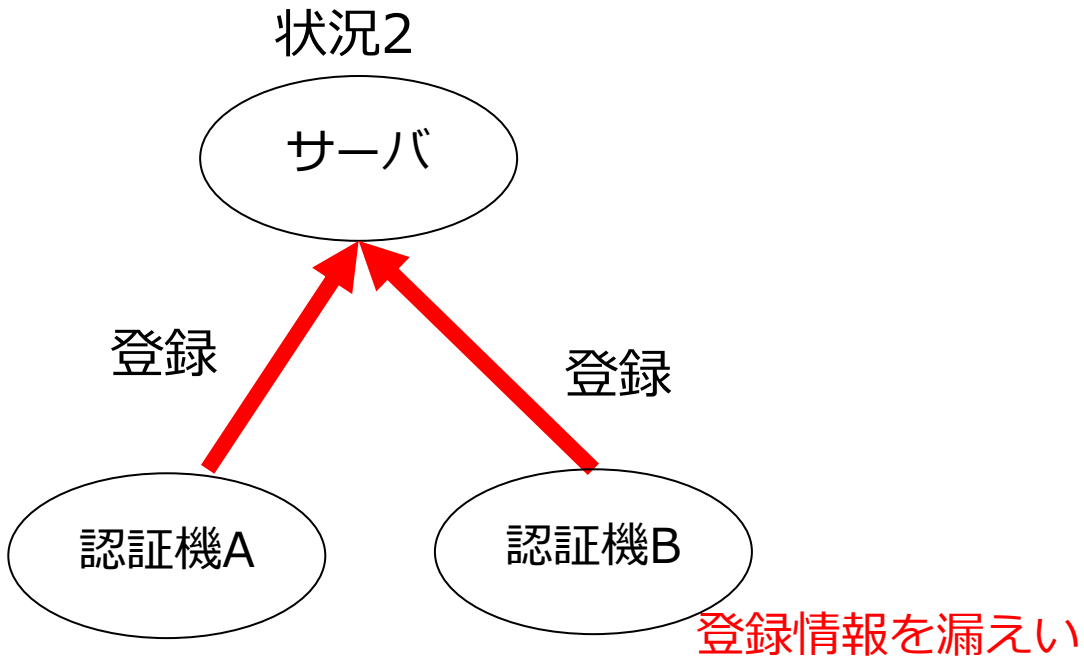


認証プロトコルの認証性

1. 認証機-サーバ間での認証性
2. 漏えいを考慮した認証機-サーバ間の認証性



サーバBが登録情報を漏洩する



認証機Bが登録情報を漏洩する

認証プロトコルの認証性

1. 認証機-サーバ間での認証性
2. 漏えいを考慮した認証機-サーバ間の認証性

認証機-サーバ間での認証性

```
query uh:UserHandle, …, cid:bitstring;
inj-event(ServerFinishAuth(u,…,c)) ==> inj-event(AuthenticatorFinishAuth(u,…,c)).
```

- 2について, クエリは1と同様だが, 情報の漏えいを形式化している

検証結果 - 認証性(sPACAの場合)

登録プロトコル

	検証結果
認証性①	✓
認証性②	×

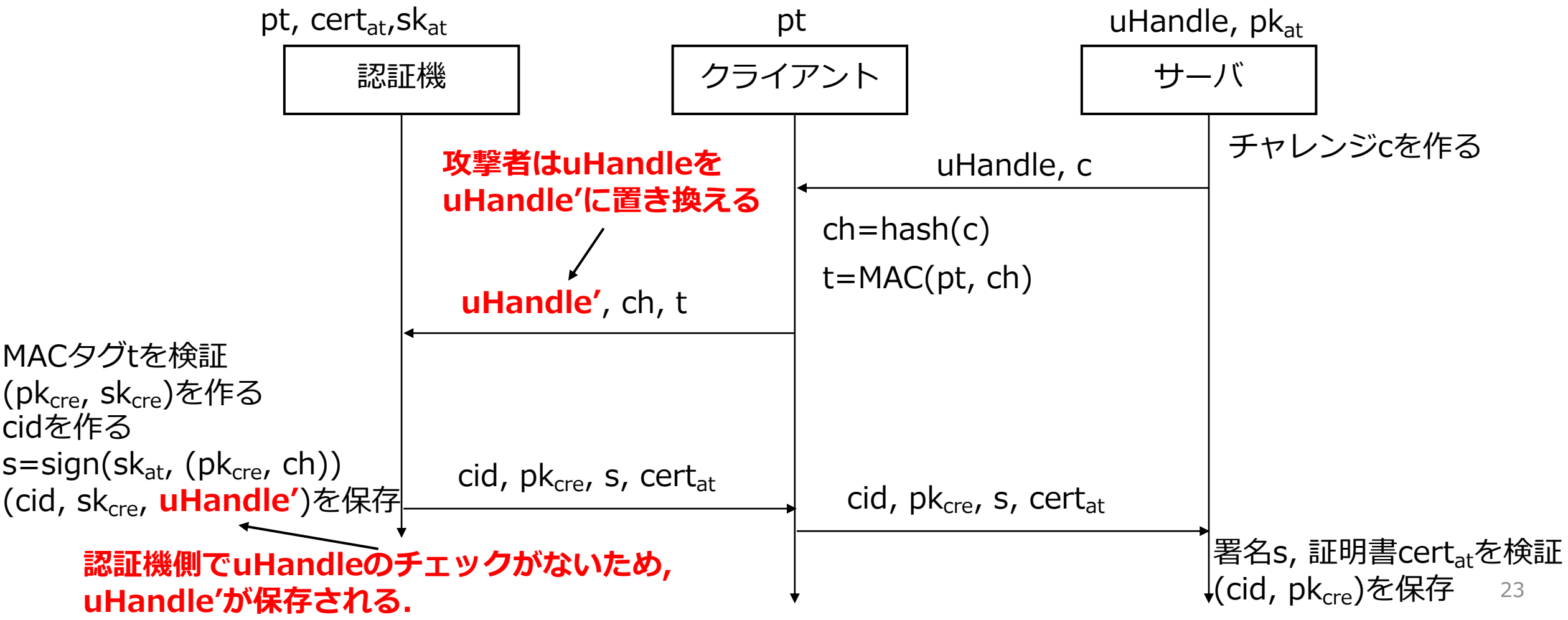
認証プロトコル

	検証結果
認証性①	✓
認証性②-状況1	✓
認証性②-状況2	✓

- アテステーションモードがnoneの場合には自明に×(攻撃者が認証機-クライアント間を自由に改ざんできるため)
- アテステーションモードがbasicの場合は自明ではない
  - 登録プロトコルにおいて, 攻撃が見つかった

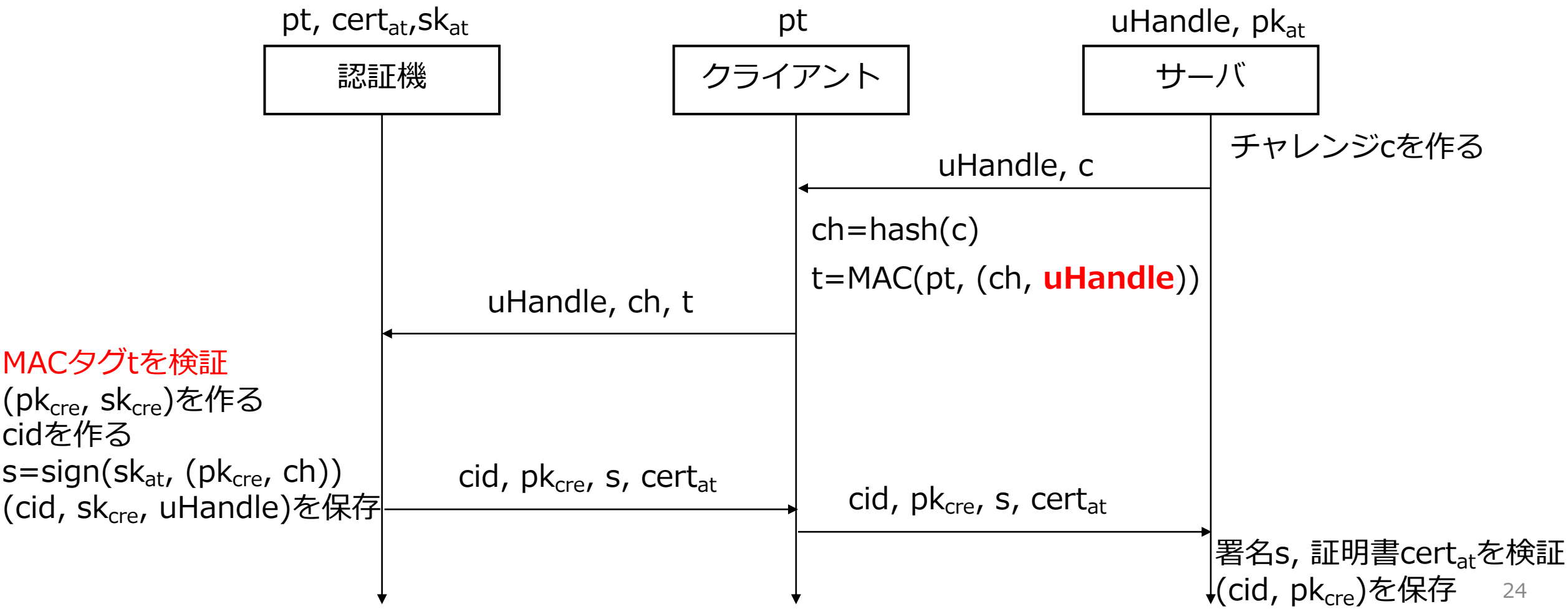
見つかった攻撃

CTAPを用いて、pinToken ptを事前に認証機、クライアント間で共有しておく



攻撃への対策

MACにuHandleを入れる



まとめ

- 既存の形式化における証明書の扱いに関するミスを修正
- 形式化されていないFIDO2のモードや仕様について形式化
- 検証の結果, 既存の検証では見つからなかった攻撃を発見