Anonymous Tokens

Michele Orrù
Anonymous Tokens

Michele Orrù

joint work with Ben Kreuter, Tancrède Lepoint, Mariana Raykova
Anonymous tokens are lightweight, single-use anonymous credentials.
Anonymous tokens are lightweight, single-use anonymous credentials.

... we focus on secret-key tokens with a private metadata bit.
The Problem

“On the Internet, nobody knows you’re a dog.”
CloudFlare's story

Website protection.

[Tor User] \( u \) \( \xrightarrow{\text{request}} \) [CloudFlare] \( i \) \( \xrightarrow{\text{request}} \) [CDN] \( w \)

\( \xleftarrow{\text{response / no}} \)

Privacy Pass: Bypassing Internet Challenges Anonymously. [PETS'18]
CloudFlare's story

Website protection.

[Privacy Pass: Bypassing Internet Challenges Anonymously. [PETS'18]]
CloudFlare's story

Website protection.

Privacy Pass: Bypassing Internet Challenges Anonymously. [PETS'18]
CloudFlare's story

Website protection.

Privacy Pass: Bypassing Internet Challenges Anonymously. [PETS'18]
CloudFlare's story

Website protection.

Privacy Pass: Bypassing Internet Challenges Anonymously. [PETS’18]
Micro payments.
Other stories

Fraud prevention.

[Fighting fraud using partially blind signatures. Facebook Engineering Blog]
Deprecating 3rd party cookies.

Chromium Blog
Private metadata

A \xrightarrow{\text{token?}} x \xleftarrow{I}
Private metadata

A

I

token?

σ^{(b)}

σ^{(b)}

request, σ^{(b)}

b
The (formal) problem

Issuance protocol:

\[ \sigma \leftarrow \langle U(pp, t), I(sk, b) \rangle \]

Redemption algorithm:

\[ \{0, 1, \perp\} \leftarrow V(sk, t, \sigma) \]
Security notions

- Unlinkability
Security notions

- Unlinkability

\[ \mathcal{A} \]

\[ (t_i, \sigma_i) \]

\[ i \]

\[ u_1 \]

\[ u_2 \]

\[ \vdots \]

\[ u_n \]
Security notions

- Unlinkability
- One-more unforgeability
Security notions

- Unlinkability
- One-more unforgeability
Security Notions

- Unnlinkability
- One-more unforgeability
- Privacy of the metadata bit

\[ \mathcal{I}(sk, b=0) \overset{\text{ind.}}{=} \mathcal{I}(sk, b=1) \]
Standardization

**W3C: Trust Token API**

```javascript
fetch('https://iacr.org/.well-known/trust-token', {
  trustToken: {
    type: 'token-request',
    issuer: 'ens.fr'
  }
});
```

[Example derived from the original proposal.]
Standardization

W3C: Trust Token API

```javascript
fetch('https://iacr.org/.well-known/trust-token', {
  trustToken: {
    type: 'token-request',
    issuer: 'ens.fr'
  }
});
```

```
fetch('https://eprint.iacr.org/2020/072.pdf', {
  trustToken: {
    type: 'raw-token-redemption',
    issuer: 'ens.fr'
  }
});
```

[Example derived from the original proposal.]
Standardization

W3C: Trust Token API

```
fetch('https://iacr.org/.well-known/trust-token', {
  trustToken: {
    type: 'token-request',
    issuer: 'ens.fr'
  }
});
```

IETF: Privacy Pass draft

```
1. Introduction

   In some situations, it may only be necessary to check that a client has been previously authorized by a service; without learning any other information. Such lightweight authorization mechanisms can be useful in quickly assessing the reputation of a client in latency-sensitive communication.
```

```
fetch('https://eprint.iacr.org/2020/072.pdf', {
  trustToken: {
    type: 'raw-token-redemption',
    issuer: 'ens.fr'
  }
});
```

[Example derived from the original proposal.] [Draft version 00]
Our contribution
Our contribution

- Formalization of Anonymous Tokens;
Our contribution

- Formalization of Anonymous Tokens;
- Private Medatada extension;
Our contribution

- Formalization of Anonymous Tokens;
- Private Metadata extension;
- New techniques for removal of zk proofs.
Our contribution

- Formalization of Anonymous Tokens;
- Private Metadata extension;
- New techniques for removal of zk proofs.
Our contribution

- Formalization of Anonymous Tokens;
- Private Medatada extension;
- New techniques for removal of zk proofs.
Related works
Related works

- Anonymous Credentials
Related works

- Anonymous Credentials
- Algebraic MACs
Related works

- Anonymous Credentials
- Algebraic MACs
- Blind Signatures
Privacy Pass

User

Issuer
Privacy Pass

User

\[ \Gamma := (p, G, G) \]
\[ X = xG \]

Issuer
Privacy Pass

User

\( r \leftarrow \mathbb{Z}_p^* \)

\( T' := r^{-1}H(t) \)

Issuer

\( \Gamma := (p, G, G) \)

\( X = xG \)
\[ \Gamma := (p, G, G) \]
\[ X = xG \]

User:
\[
\begin{align*}
  r &\leftarrow \mathbb{Z}_p^* \\
  T' &:= r^{-1}H(t) \\
  W &:= rW'
\end{align*}
\]

Issuer:
\[
\begin{align*}
  T' &
  \\
  W' &:= xT'
\end{align*}
\]
Privacy Pass

User

\[ \Gamma := (p, G, G) \]
\[ X = xG \]

\[ r \leftarrow \mathbb{Z}_p^* \]
\[ T' := r^{-1}H(t) \]

\[ W := rW' \]

Issuer

\[ T' \]

\[ W' \]

\[ W' := xT' \]

\[ \cdots \text{redemption}\cdots \]

\[ t, W \]

1. check \( xH(t) = W \)
2. add \( t \) to spent tokens.
Privacy Pass

User

\[ \Gamma := (p, G, G) \]
\[ X = xG \]
\[ \begin{align*}
    r &\leftarrow \mathbb{Z}_p^* \\
    T' &:= r^{-1} H(t)
\end{align*} \]

check \( \pi \)

\[ W := rW' \]

Issuer

\[ T' \]

\[ W', \pi \]

\[ W' := xT' \]

\[ \pi := \text{zkp} \left\{ x \begin{bmatrix} G \\ T' \end{bmatrix} = \begin{bmatrix} X \\ W' \end{bmatrix} \right\} \]

\[ t, W \]

1. check \( xH(t) = W \)
2. add \( t \) to spent tokens.
Private metadata?

User

\[ \Gamma := (p, \mathbb{G}, G) \]
\[ X = x_b G, \ b \in \{0, 1\} \]

\[ r \leftarrow \mathbb{Z}_p^* \]
\[ T' := r^{-1} H(t) \]

check \( \pi \)
\[ W := r W' \]

Issuer

\[ T' \]
\[ W', \pi \]
\[ W' := x_b T' \]
\[ \pi := \text{zkp} \left\{ x_b \begin{bmatrix} G \\ T' \end{bmatrix} = \begin{bmatrix} X_b \\ W' \end{bmatrix} \right\} \]

\[ t, W \]

1. check \( b \) s.t. \( x_b H(t) = W \)
2. add \( t \) to spent tokens.
\[ \Gamma := (p, \mathbb{G}, G) \]
\[ X_b = x_b G, \quad b \in \{0, 1\} \]

- \( r, s \leftarrow \mathbb{Z}_p^* \)
- \( T' := r^{-1} H(t) \)
- \( S' := s^{-1} H(t) \)

\[ W' := x_0 T' \]
\[ V' := x_1 S' \]
\[ \Gamma := (p, \mathbb{G}, G) \]
\[ X_b = x_b G, \quad b \in \{0, 1\} \]

\[ r, s \leftarrow \mathbb{Z}_p^* \]
\[ T' := r^{-1} H(t) \]
\[ S' := s^{-1} H(t) \]

\[ rW' \overset{?}{=} sV' \]
Privacy Pass variant

User

\[ \Gamma := (p, \mathbb{G}, G, H) \]
\[ X = xG + yH \]
\[ r \leftarrow \mathbb{Z}_p^* \]
\[ T' := r^{-1}H(t) \]

Issuer

\[ s \leftarrow \{0, 1\}^\lambda; \ S' := H(T', s) \]
\[ W := xT' + yS' \]
Privacy Pass variant

User

\[ \Gamma := (p, G, G, H) \]
\[ X = xG + yH \]
\[ r \leftarrow \mathbb{Z}_p^* \]
\[ T' := r^{-1}H(t) \]
\[ W := rW' \]
\[ S := rH(T', s) \]

Issuer

\[ s \leftarrow \{0, 1\}^\lambda; \quad S' := H(T', s) \]
\[ W := xT' + yS' \]

\[ s, W', \pi \]

\[ t, S, W \]

1. check \( xH(t) + yS = W \)
2. add \( t \) to spent tokens.
Privacy Pass variant

**User**

\[ \Gamma := (p, G, G, H) \]

\[ X = xG + yH \]

\[ r \leftarrow \mathbb{Z}_p^* \]

\[ T' := r^{-1}H(t) \]

check \( \pi \)

\[ W := rW' \]

\[ S := rH(T', s) \]

**Issuer**

\[ s \leftarrow \{0, 1\}^\lambda; \]

\[ S' := H(T', s) \]

\[ W := xT' + yS' \]

\[ \pi := \text{zkp}\left\{ x \begin{bmatrix} G \\ T' \end{bmatrix} + y \begin{bmatrix} H \\ S' \end{bmatrix} = \begin{bmatrix} X \\ W' \end{bmatrix} \right\} \]

\[ \cdots \text{redemption} \cdots \]

\[ t, S, W \]

1. check \( xH(t) + yS = W \)
2. add \( t \) to spent tokens.
\[ \pi := (p, G, G, H) \]

\[ X_b = x_b G + y_b H, \quad b \in \{0, 1\} \]

\[ r \leftarrow \mathbb{Z}_p^* \]

\[ T' := r^{-1} H(t) \]

check \( \pi \)

\[ W := r W' \]

\[ S := r H(T', s) \]

\[ s \leftarrow \{0, 1\}^\lambda; \quad S' := H(T', s) \]

\[ W := x_b T' + y_b S' \]

\[ \pi := \text{zkp} \left\{ x_b \begin{bmatrix} G \\ T' \end{bmatrix} + y_b \begin{bmatrix} H \\ S' \end{bmatrix} = \begin{bmatrix} X_b \\ W' \end{bmatrix} \right\} \]

\[ \cdots \text{redemption} \cdots \]

\[ t, S, W \]

1. check b s.t. \( x_b H(t) + y_b S = W \)

2. add \( t \) to spent tokens.
Removing the zk proof

**User**

\[ r, \rho \leftarrow \mathbb{Z}_p^* \]
\[ T' := r(H(t) - \rho G) \]
\[ W := r^{-1}W' + \rho X \]

**Issuer**

\[ \Gamma := (p, G, G) \]
\[ X = xG \]
\[ T' \]
\[ W', \pi \]

\[ W' := xT' \]

\[ t, W \]

1. check \( xH(t) = W \)
2. add \( t \) to spent tokens.
Concrete security
Concrete security

- One-more Diffie-Hellman is not extensively studied;
Concrete security

- One-more Diffie-Hellman is not extensively studied;
- Token Hijacking;
Concrete security

- One-more Diffie-Hellman is not extensively studied;
- Token Hijacking;
- Engineering issues.
fn it_works() {
    let mut csrng = rand::rngs::OsRng;
    // generate a keypair
    let keypair = KeyPair::generate(&mut csrng);

    // get the public parameters
    let pp = PublicParams::from(&keypair);
    // client's first message (the blinded token)
    let blinded_token = pp.generate_token(&mut csrng);
    // server's response (the signed token) with hidden metadata bit 0
    let signed_token = keypair.sign(&mut csrng, &blinded_token.to_bytes(), 0);
    // client's unblinding (the final token)
    let token = blinded_token.unblind(signed_token);
    assert!(token.is_ok());

    // verification of the token
    assert!(keypair.verify(&token.unwrap()).is_ok());
}
Future directions
Future directions

- public metadata
Future directions

- public metadata

- public verifiability
  - blind BLS
  - blind Okamoto-Schnorr? broken :( 
Future directions

- public metadata

- public verifiability
  - blind BLS
  - blind Okamoto-Schnorr? *broken* :

- batching proofs
Future directions

• public metadata

• public verifiability
  ▪ blind BLS
  ▪ blind Okamot-Schnorr? broken :(  

• batching proofs
Future directions

- public metadata

- public verifiability
  - blind BLS
  - blind Okamoto-Schnorr? broken :(

- batching proofs
Future directions

- public metadata

- public verifiability
  - blind BLS
  - blind Okamoto-Schnorr? broken :(

- batching proofs