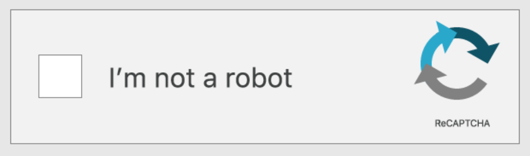
Anonymous Tokens

Serge Vaudenay

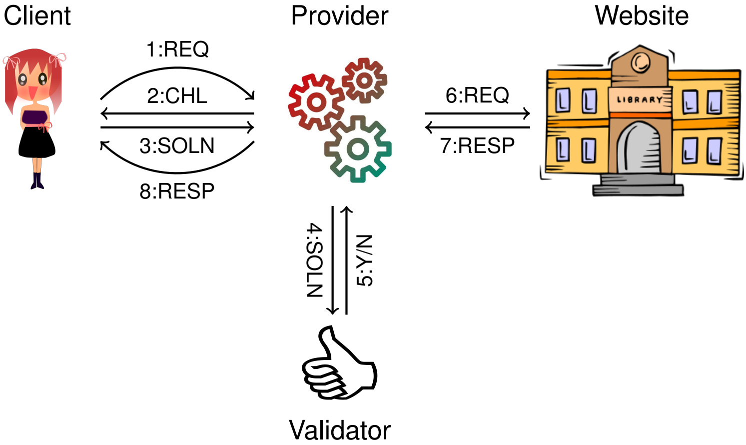
Captcha - Proof of Humanhood



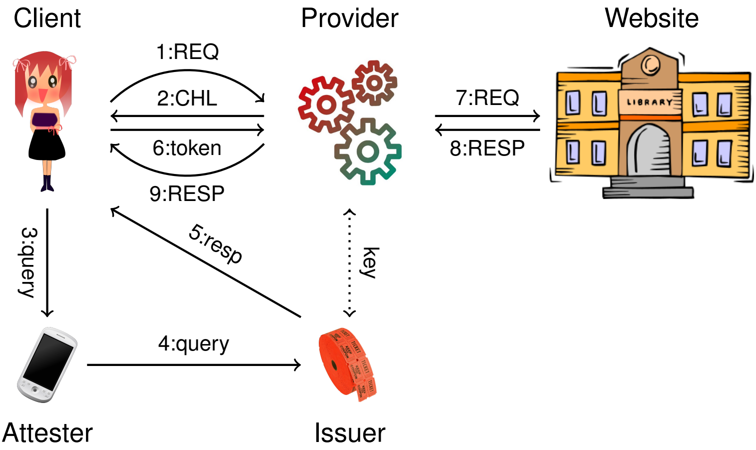
- not a good UX
- sometimes ambiguous
- not really secure
- free human labor to train AI

$\rightarrow$ really unpleasant

Browsing Model (e.g. with Captcha)

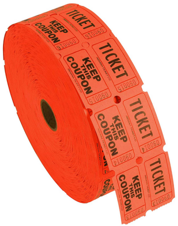


Model to Eliminate Captchas

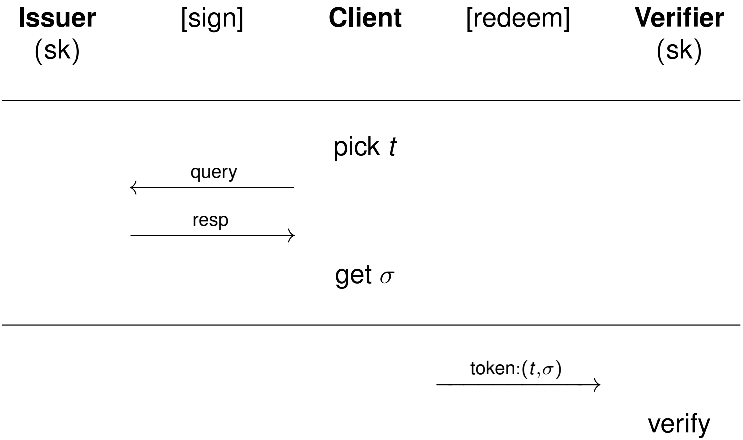


Applications

- separate authorization from service
- let the client carry its own authorization
- ticketing: issuer=cashier verifier=server



Privacy-Preserving e-Ticketing



Privacy Pass (Simplified)

key generation: $Y = \mathrm{sk} \cdot X$

Issuer	[sign]	**Client**	[redeem]	**Verifier**
(sk)		(X, Y)		(sk)
		pick t		
		$r \xleftarrow{\$} \mathbf{Z}_q$		
$Q \leftarrow \mathrm{sk} \cdot P$	$\xleftarrow{P}$	$P \leftarrow r \cdot H(t)$		
$\pi \leftarrow \mathrm{DLEQ}\begin{pmatrix} X & Y \\ P & Q \end{pmatrix}$	$\xrightarrow{Q,\pi}$	verify π		
(π: proof of $\log_X Y = \log_P Q$)		verify $X \neq 0$		
		$W \leftarrow \frac{1}{r} \cdot Q$		
			$\xrightarrow{t,W}$	is t fresh?
				$W \stackrel{?}{=} \mathrm{sk} \cdot H(t)$

$\approx$ *symmetric blind signature*

DLEQ from Schnorr Generalized + Fiat-Shamir

- group homomorphism $\varphi : \mathbf{Z}_q^m \to G^n$, prime q
- relation $R((\varphi, \vec{V}), \vec{w})$: $\varphi(\vec{w}) = \vec{V}$
- Σ-protocol with Fiat-Shamir:

Prover		**Verifier**
$\vec{w}$ st $\varphi(\vec{w}) = \vec{V}$	$\vec{V}$	
pick $\vec{k} \in \mathbf{Z}_q^m$		
$\vec{R} \leftarrow \varphi(\vec{k})$	$\xrightarrow{\vec{R}}$	
	$\xleftarrow{e}$	$e \leftarrow H(\vec{V}, \vec{R})$
$\vec{s} \leftarrow e\vec{w} + \vec{k}$	$\xrightarrow{\vec{s}}$	$\vec{R} + e\vec{V} \stackrel{?}{=} \varphi(\vec{s})$

- $\pi = (\vec{R}, \vec{s})$
- DLEQ: discrete log equality

m	n	$\vec{w}$	$\vec{V}$	$\varphi(\vec{w})$
1	2	sk	(Y, Q)	$(\mathrm{sk} \cdot X, \mathrm{sk} \cdot P)$

Privacy Pass with Batch Signature

Issuer		**Client**
(sk)	$Y = \text{sk} \cdot X$	(X, Y)
		pick $\vec{t}$
		$r_i \xleftarrow{\$} \mathbf{Z}_q$
$\vec{Q} \leftarrow \text{sk} \cdot \vec{P}$	$\xleftarrow{\quad \vec{P} \quad}$	$P_i \leftarrow r_i \cdot H(t_i)$
pick seed $\mapsto \vec{c}$		
$M = \sum_i c_i \cdot P_i$		
$Z = \sum_i c_i \cdot Q_i$		
$\pi \leftarrow \text{DLEQ}\begin{pmatrix} X & Y \\ M & Z \end{pmatrix}$	$\xrightarrow{\quad \vec{Q},\pi,\text{seed} \quad}$	verify π, seed
		$W_i \leftarrow \frac{1}{r_i} \cdot Q_i$

- batch proof with a pseudorandom linear combination
- add seed in the proof
- Client gets N tokens (t_i, W_i)

Privacy Pass with Request Authorization

Client		**Verifier** (sk)
$\mu \leftarrow \mathrm{MAC}_{\mathrm{KDF}(t,W)}(R)$	$\xrightarrow{t,R,\mu}$	is t fresh?
		$\mu \stackrel{?}{=} \mathrm{MAC}_{\mathrm{KDF}(t,\mathrm{sk}\cdot H(t))}(R)$

- use (t, W) to derive a one-time MAC key
- use the MAC to authorize request R

No Double-Spending

- t must be fresh (nonce)
- use a Bloom filter to detect t reuse
- update sk frequently (expire tokens)

OPRF

- PP is an oblivious computation (OPRF) of:

$$\mathrm{PRF}(t) = \mathsf{sk} \cdot H(t)$$

- PP is a “verifiable” by the client (VOPRF) using DLEQ
- we can make it universally verifiable using pairing and $\hat{Y} = \mathsf{sk} \cdot \hat{X}$:

$$e(\mathrm{PRF}(t), \hat{X}) = e(H(t), \hat{Y})$$

- we can use other OPRF
- we can use “randomized PRF” (algebraic MAC)

From OPRF to Algebraic MAC

- instead of a PRF, how about a (non-deterministic) authentication code?
- with secret (x, y)

$$\mathrm{MAC}_{x,y}(m) \to (P, (x + ym)P)$$

- with secret x

$$\mathrm{MAC}_{x}(m) \to \left(r, s, \frac{1}{x + s}(G_1 + mG_2 + rG_3)\right)$$

- can easily replace m by a vector of scalar attributes

Anonymous Token with Hidden Metadata (ATHM)

Client		Issuer
pp $= (\mathrm{gp}, q, G, Z, Y'')$, m'		sk $= (x, y, y', y'', z)$, $b \in \{0,1\}$, m
$r, t_C \leftarrow \mathbf{Z}_q$		$(Z = zG)$ $(Y'' = y''G)$
$T \leftarrow m'Y'' + t_C Z + rG$		
	$\xrightarrow{\quad T \quad}$	$t_S \leftarrow \mathbf{Z}_q$
		$d \leftarrow \mathbf{Z}_q^*$
		$U \leftarrow dG$
		$V \leftarrow d(xG + byG + my'G + t_S zG + T)$
verify m, U, V, t_S, π	$\xleftarrow{\quad m, U, V, t_S, \pi \quad}$	$\pi \leftarrow$ proof
verify $U \neq 0$		
$c \leftarrow \mathbf{Z}_q^*$		
$P \leftarrow cU$		private bit metadata by issuer
$Q \leftarrow c(V - rU)$		public metadata
$t \leftarrow t_C + t_S$		private metadata by client
$\sigma \leftarrow (P, Q)$		
output: m, m', t, σ		

redeem: verify $P \neq 0$ and $Q = (x + by + my' + m'y'' + tz)P$

Tricky Part about Private Metadata by Issuer

- can be used as a marker
- $\rightarrow$ degrages unlinkability
- we must enforce that the information is limited (one bit)
- we must define unlinkability “up to one bit”

Extension: Anonymous Credentials

- **Anonymous Credentials**:
  redeem part is a ZK proof (multi-use credentials)
  verifiable without secret
- **Keyed-Verification Anonymous Credentials** (KVAC):
  same but with a secret to verify

Extension: Non-Transferability

- nominative + anonymous token !!!
- idea: redeem requires client's long-term secret
- assume that client is identified during issuance
- (later) client proves possession of a valid identity

Non-Transferable Anonymous Token (NTAT)

Client		Issuer
Client	pp: G_1, G_2, G_3, G_4	**Issuer**
server public key: Y		client identity: X
secret: x ($X = xG_1$)		secret: y ($Y = yG_2$)
$r \leftarrow \mathbf{Z}_q$, $\delta \leftarrow \mathbf{Z}_q^*$		
$T \leftarrow \delta \cdot (X + rG_3 + G_4)$		
$\pi_C \leftarrow$ proof$(r, x; X, T)$	$\xrightarrow{T, \pi_C}$	verify π_C with X, T
		$s \leftarrow \mathbf{Z}_q - \{-y\}$
		$S \leftarrow \frac{1}{y+s} T$
verify π_S with Y, s, S	$\xleftarrow{s, S, \pi_S}$	$\pi_S \leftarrow$ proof$(y; Y, s, S)$
$\sigma \leftarrow \frac{1}{\delta} S$		
$(\sigma = \frac{1}{y+s}(xG_1 + rG_3 + G_4))$		
output: σ		
private output: r, s		
$\sigma' \leftarrow xG_1 + rG_3 + G_4 - s\sigma$	$\xrightarrow{\sigma, \sigma'}$	verify $\sigma' = y\sigma$
	$\xleftrightarrow{\text{PoK}(x, r, s)}$	$(xG_1 + rG_3 - s\sigma = \sigma' - G_4)$

Privacy: Unlinkability

"signing and redeeming are unlinkable"

Game UNLINK_b:

1: setup
2: $\mathcal{A} \to X, Y$
3: pick t_0, t_1
4: compute P_0, P_1
5: $\mathcal{A}(P_0, P_1) \to Q_0, \pi_0, Q_1, \pi_1$
6: verify π_0, π_1
7: compute W_0, W_1
8: $\mathcal{A}(t_b, W_b) \to z$
9: **return** z

Oracle $\text{RO}(z)$:

10: **return** $H(z)$

$$\text{Adv} = \Pr[z = 1 | b = 1] - \Pr[z = 1 | b = 0]$$

Theorem

For any $\mathcal{A}$, we have $\text{Adv} \leq 2\frac{2+\#\{H \text{ queries}\}}{q}$ *in ROM.*

Proof

- By using the Difference Lemma, we reduce UNLINK to Game 1

$$|\mathrm{Adv} - \mathrm{Adv}_1| \leq 2\Pr[\neg \log_X Y = \log_{P_0} Q_0 = \log_{P_1} Q_1]$$

- $\mathrm{Adv}_1 = \mathrm{Adv}_2 = \mathrm{Adv}_3$
- Game 3 does not use b so $\mathrm{Adv}_3 = 0$

Game 1:

setup
$\mathcal{A} \to X, Y$
$\mathrm{sk} \leftarrow \log_X Y$
pick t_0, t_1, r_0, r_1
$P_i \leftarrow r_i \cdot H(t_i),\ i = 0, 1$
$\mathcal{A}(P_0, P_1)$
$Q_i \leftarrow \mathrm{sk} \cdot P_i,\ i = 0, 1$
$W_i \leftarrow \frac{1}{r_i} \cdot Q_i,\ i = 0, 1$
$\mathcal{A}(t_b, W_b) \to z$
return z

$\to$

Game 2:

setup
$\mathcal{A} \to X, Y$
$\mathrm{sk} \leftarrow \log_X Y$
pick t_0, t_1
pick P_0, P_1
$\mathcal{A}(P_0, P_1)$
$W_i \leftarrow \mathrm{sk} \cdot H(t_i),\ i = 0, 1$
$\mathcal{A}(t_b, W_b) \to z$
return z

$\to$

Game 3:

setup
$\mathcal{A} \to X, Y$
$\mathrm{sk} \leftarrow \log_X Y$
pick t
pick P_0, P_1
$\mathcal{A}(P_0, P_1)$
$W \leftarrow \mathrm{sk} \cdot H(t)$
$\mathcal{A}(t, W) \to z$
return z

Soundness of DLEQ

Prover		Verifier
	$\varphi(\mathrm{sk}) = \mathrm{sk} \cdot (X, P)$	
sk st $\varphi(\mathrm{sk}) = (Y, Q)$	(Y, Q)	
pick $k \in \mathbf{Z}_q$		
$\vec{R} \leftarrow \varphi(k)$	$\xrightarrow{\vec{R}}$	
	$\xleftarrow{e}$	$e \leftarrow H(Y, Q, \vec{R})$
$s \leftarrow e \cdot \mathrm{sk} + k$	$\xrightarrow{s}$	$\vec{R} + e \cdot (Y, Q) \stackrel{?}{=} \varphi(s)$
	$\pi = (\vec{R}, s)$	

- set $E = \varphi(\mathbf{Z}_q)$
- if $(Y, Q) \notin E$, then $\Pr[\vec{R} + H(Y, Q, \vec{R}) \cdot (Y, Q) \in E] \leq \frac{1}{q}$
- for each $(Y, Q, \vec{R})$ query to H, the probability it defines a correct π is bounded by $\frac{1}{q}$ if $\log_X Y \neq \log_P Q$
- the probability that a non-query gives a valid π is $\frac{1}{q}$

Security: One-More-Unforgeability

"cannot redeem ℓ times after $\ell - 1$ signatures"

Game OMUF:

```
setup, key generation
set 𝒜's view to X, Y
for i = 1 to ℓ − 1 do
    𝒜 → P_i
    compute Q_i, π_i
    add to 𝒜's view
end for
for i = 1 to ℓ do
    𝒜 → t_i, W_i
    redeem (t_i, W_i)
end for
```

Oracle RO(z):

```
return H(z)
```

$$\mathrm{Adv} = \Pr[\text{all redeems succeed and all } t_i \text{ different}]$$

Theorem

For any PPT $\mathcal{A}$, we have $\mathrm{Adv} = \mathrm{negl}$*, assuming the hardness of OMCDH in ROM.*

One-More CDH

"cannot compute $\ell + 1$ power-sk from ℓ queries"

Game OMCDH:

1: setup
2: pick sk
3: cnt $\leftarrow 0$
4: $C \xleftarrow{\$} (C_1, \ldots, C_{\ell+1})$
5: $\mathcal{B}(C) \rightarrow (D_1, \ldots, D_{\ell+1})$

Oracle $\mathcal{O}(Z)$:

6: increment cnt
7: **if** cnt $> \ell$ **then** abort
8: **return** sk $\cdot Z$

$$\mathrm{Adv} = \Pr[D_i = \mathrm{sk} \cdot C_i \text{ for all } i]$$

Theorem

For any PPT $\mathcal{A}$ playing OMUF, there is a PPT $\mathcal{B}$ playing OMCDH such that $\mathrm{Adv}_{\mathcal{A}} \leq \mathrm{Adv}_{\mathcal{B}} + \mathrm{negl}$.

Proof of PP in ROM

- to construct $\mathcal{B}(C)$:
  - set $X = C_{\ell+1}$
  - call $\mathcal{O}(X)$ and set $D_{\ell+1} = Y = \mathcal{O}(X)$
  - run $\mathcal{A}(X, Y)$
  - whenever $\mathcal{A}$ returns P_i, call $\mathcal{O}(P_i) \to Q_i$ and forge π_i using ROM programmability (negl loss)
  - whenever $\mathcal{A}$ calls RO(t), return $H_t = \sum_{j=1}^{\ell} r(t)^{j-1} \cdot C_j$ where $r(\cdot)$ is a random function
  - in the end, invert a Vandermonde matrix with the $r(t_i)$, multiply to $\vec{W}$ to get $(D_1, \ldots, D_\ell)$
- at the end of the game, assume that every RO(t_i) was queried in winning cases (negl loss)
  $\to$ deduce $W_i = \text{sk} \cdot H_i$, $i = 1, \ldots, \ell$
- deduce $D_i = \text{sk} \cdot C_i$, $i = 1, \ldots, \ell + 1$

OMCDH in the Algebraic Group Model (AGM)

$\mathcal{B}$ must provide an expression of the D_i and P_i in terms of the C_i and $Q_i = \mathsf{sk} \cdot P_i$: $\vec{D} = \mathcal{D}\vec{C} + \bar{\mathcal{D}}\vec{Q}$, $\vec{P} = \mathcal{P}\vec{C} + \bar{\mathcal{P}}\vec{Q}$ ($\bar{\mathcal{P}}$ triangular)

- $(\mathcal{I} - \mathsf{sk}\bar{\mathcal{P}})\vec{P} = \mathcal{P}\vec{C}$ so $\vec{P} = (\mathcal{I} + \mathsf{sk}\bar{\mathcal{P}} + \cdots + \mathsf{sk}^{\ell-1}\bar{\mathcal{P}}^{\ell-1})\mathcal{P}\vec{C}$

$$\underbrace{\vec{D} - \mathsf{sk}\vec{C}}_{0 \text{ if win}} = \left(\underbrace{\mathcal{D} + \mathsf{sk}\bar{\mathcal{D}}(\mathcal{I} + \mathsf{sk}\bar{\mathcal{P}} + \cdots + \mathsf{sk}^{\ell-1}\bar{\mathcal{P}}^{\ell-1})\mathcal{P} - \mathsf{sk}\mathcal{I}}_{\mathsf{MatPoly}(\mathsf{sk})}\right)\vec{C}$$

- in the winning case:

case 1 MatPoly(sk) $\neq 0$: $\rightarrow \vec{C}$ in a non-trivial kernel ($\rightarrow$ solve Dlog)

case 2 $\mathcal{D} = 0$: $\rightarrow \mathsf{sk}\vec{C} = \vec{D} = \mathsf{sk}\bar{\mathcal{D}}\vec{P}$ so $\vec{C} = \bar{\mathcal{D}}\vec{P}$ we generate $\vec{C}$ from a $< \ell + 1$ vector $\vec{P}$ ($\rightarrow$ solve Dlog)

case 3 other: $\rightarrow$ find sk as a root of MatPoly ($\rightarrow$ solve sk with $\mathcal{O}$)

Theorem

In AGM, solving OMCDH implies solving $(\ell - 1)$-Dlog:

$$G, \mathsf{sk}G, \ldots, \mathsf{sk}^{\ell-1}G \mapsto \mathsf{sk}$$

Conclusion

	anonymous tokens		credentials
non verifiable	OPRF	(O)MAC	KVAC
univ. verifiable		blind signature	anon. credentials

- many cryptographic primitives for authorization
- many options, efficient

References

- Privacy Pass: Bypassing Internet Challenges Anonymously
  PoPETs 2018 (Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, Filippo Valsorda)
  `https://doi.org/10.1515/popets-2018-0026`
- Anonymous Tokens with Stronger Metadata Bit Hiding from Algebraic MACs
  CRYPTO 2023 (Melissa Chase, F. Betül Durak, Serge Vaudenay)
  `https://doi.org/10.1007/978-3-031-38545-2_14`
  `https://eprint.iacr.org/2022/1622`
- Non-Transferable Anonymous Tokens by Secret Binding
  CCS 2024 (F. Betül Durak, Laurane Marco, Abdullah Talayhan, Serge Vaudenay)
  (to appear)
  `https://eprint.iacr.org/2024/711`
- Anonymous Credentials Zoo (Michele Orrù)
  `https://tokenzoo.github.io`