Fast and simple constant-time hashing to the BLS12-381 elliptic curve

(and other curves, too!)

Riad S. Wahby, Dan Boneh

Stanford

August 26^{th}, 2019

Motivation

Why do we need hashes to elliptic curves?

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, ...

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, ...

Why simple and constant time?

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, ...

Why simple and constant time?

- Side channels (e.g., Dragonblood [VR19])

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs,

fixed-modulus arithmetic only

Why simple and constant time?

- Side channels (e.g., Dragonblood [VR19])

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, ...

Why simple and constant time?

- Side channels (e.g., Dragonblood [VR19])
- Embedded systems often have fixed-modulus hardware acceleration but *slow* generic bigint

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, ...

Why simple and constant time?

- Side channels (e.g., Dragonblood [VR19])
- Embedded systems often have fixed-modulus hardware acceleration but *slow* generic bigint

Why the BLS12-381 pairing-friendly elliptic curve?

- Widely used curve for $\approx$120-bit security level

Motivation

Why do we need hashes to elliptic curves?

- Our initial motivation: BLS signatures [BLS01]
- Also: VRFs, OPRFs, PAKEs, IBE, . . .

Why simple and constant time?

- Side channels (e.g., Dragonblood [VR19])
- Embedded systems often have fixed-modulus hardware acceleration but *slow* generic bigint

Why the BLS12-381 pairing-friendly elliptic curve?

- Widely used curve for $\approx$120-bit security level
  - ☞ ZK proofs, signatures, IBE, ABE, . . .

Our contributions

1. An “indirect” map to pairing-friendly curves that sidesteps limitations of existing maps

Our contributions

1. An “indirect” map to pairing-friendly curves that sidesteps limitations of existing maps

2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   ✓ On par with the fastest existing maps

Our contributions

1. An “indirect” map to pairing-friendly curves that sidesteps limitations of existing maps

2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   - ✓ On par with the fastest existing maps
   - ✓ Fast impls are simple and constant time

Our contributions

1. An "indirect" map to pairing-friendly curves that sidesteps limitations of existing maps

2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   - ✓ On par with the fastest existing maps
   - ✓ Fast impls are simple and constant time
   - ✓ Applies to essentially any prime-field curve

Our contributions

1. An “indirect” map to pairing-friendly curves that sidesteps limitations of existing maps

2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   - ✓ On par with the fastest existing maps
   - ✓ Fast impls are simple and constant time
   - ✓ Applies to essentially any prime-field curve

3. Impl and eval of 34 hash variants for BLS12-381

Our contributions

1. An "indirect" map to pairing-friendly curves that sidesteps limitations of existing maps
2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   - ✓ On par with the fastest existing maps
   - ✓ Fast impls are simple and constant time
   - ✓ Applies to essentially any prime-field curve
3. Impl and eval of 34 hash variants for BLS12-381
   - ✓ 1.3–2$\times$ faster than prior constant-time hashes, $\leq$ 9% slower than *non*-CT deterministic hashes

Our contributions

1. An "indirect" map to pairing-friendly curves that sidesteps limitations of existing maps
2. An optimization to the map of [BCIMRT10] that reduces its cost to 1 exponentiation
   - ✓ On par with the fastest existing maps
   - ✓ Fast impls are simple and constant time
   - ✓ Applies to essentially any prime-field curve
3. Impl and eval of 34 hash variants for BLS12-381
   - ✓ 1.3–2$\times$ faster than prior constant-time hashes, $\leq$ 9% slower than *non*-CT deterministic hashes
   - ☞ Open-source impls in C, Rust, Python, ...

Roadmap

1. Hash functions to elliptic curves

2. Optimizing the map of [BCIMRT10]

3. Evaluation results

Notation

$\mathbb{F}_p$ is the finite field of integers mod a prime p

Notation

$\mathbb{F}_p$ is the finite field of integers mod a prime p

$H_p : \{0,1\}^\star \to \mathbb{F}_p$ modeled as a random oracle

Notation

$\mathbb{F}_p$ is the finite field of integers mod a prime p

$H_p : \{0,1\}^\star \to \mathbb{F}_p$ modeled as a random oracle

$E(\mathbb{F}_p)$ is the elliptic curve group with identity $\mathcal{O}$ and points $\{(x, y) : x, y \in \mathbb{F}_p, y^2 = x^3 + ax + b\}$
☞ multiplicative notation

Notation

$\mathbb{F}_p$ is the finite field of integers mod a prime p

$H_p : \{0,1\}^\star \to \mathbb{F}_p$ modeled as a random oracle

$E(\mathbb{F}_p)$ is the elliptic curve group with identity $\mathcal{O}$
and points $\{(x,y) : x, y \in \mathbb{F}_p, y^2 = x^3 + ax + b\}$
☞ multiplicative notation

$\mathbb{G} \subseteq E(\mathbb{F}_p)$ is a subgroup of prime order q.
$\#E(\mathbb{F}_p) = hq$; h is the *cofactor*.

Hash and check

$\text{HashToCurve}_{\text{H\&C}}(\text{msg})$:

$\text{ctr} \leftarrow 0$

$y \leftarrow \perp$

while $y = \perp$:

$\quad x \leftarrow H_p(\text{ctr} \,||\, \text{msg})$

$\quad \text{ctr} \leftarrow \text{ctr} + 1$

$\quad ySq \leftarrow x^3 + ax + b$

$\quad y \leftarrow \text{sqrt}(ySq)$ // $\perp$ if ySq is non-square

$P \leftarrow (x, y)$

return P^h // map to $\mathbb{G}$ via cofactor mul

Hash and check

HashToCurve$_{\mathsf{H\&C}}$(msg):

```
ctr ← 0
y ← ⊥
while y = ⊥:
    x ← H_p(ctr || msg)
    ctr ← ctr + 1
    ySq ← x^3 + ax + b
    y ← sqrt(ySq)        // ⊥ if ySq is non-square
P ← (x, y)
return P^h               // map to 𝔾 via cofactor mul
```

Hash and check

$\mathsf{HashToCurve}_{\mathsf{H\&C}}(\mathsf{msg})$:

$\quad \mathsf{ctr} \leftarrow 0$

$\quad y \leftarrow \perp$

$\quad$ while $y = \perp$:

$\quad\quad x \leftarrow H_p(\mathsf{ctr} \,||\, \mathsf{msg})$

$\quad\quad \mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

$\quad\quad ySq \leftarrow x^3 + ax + b$

$\quad\quad y \leftarrow \mathsf{sqrt}(ySq)$ $\quad$ // $\perp$ if ySq is non-square

$\quad P \leftarrow (x, y)$

$\quad$ return P^h $\quad$ // map to $\mathbb{G}$ via cofactor mul

☞ $E(\mathbb{F}_p) = \{(x, y) : x, y \in \mathbb{F}_p, y^2 = x^3 + ax + b\}$

Hash and check

$\mathsf{HashToCurve}_{\mathsf{H\&C}}(\mathsf{msg})$:

$\mathsf{ctr} \leftarrow 0$

$y \leftarrow \perp$

while $y = \perp$:

$x \leftarrow H_p(\mathsf{ctr} \,||\, \mathsf{msg})$

$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

$ySq \leftarrow x^3 + ax + b$

$y \leftarrow \mathsf{sqrt}(ySq)$ // $\perp$ if ySq is non-square

$P \leftarrow (x, y)$

return P^h // map to $\mathbb{G}$ via cofactor mul

Hash and check

$\mathsf{HashToCurve}_{\mathsf{H\&C}}(\mathsf{msg})$:

$\quad \mathsf{ctr} \leftarrow 0$

$\quad y \leftarrow \perp$

$\quad$ while $y = \perp$:

$\qquad x \leftarrow H_p(\mathsf{ctr} \,||\, \mathsf{msg})$

$\qquad \mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

$\qquad ySq \leftarrow x^3 + ax + b$

$\qquad y \leftarrow \mathsf{sqrt}(ySq) \qquad$ // $\perp$ if ySq is non-square

$\quad P \leftarrow (x, y)$

$\quad$ return $P^h \qquad$ // map to $\mathbb{G}$ via cofactor mul

Not constant time; "bad" inputs are easy to find.

Hash and check

$\mathrm{HashToCurve}_{\mathrm{H\&C}}(\mathrm{msg})$:

$$
\begin{array}{l}
\quad \mathrm{ctr} \leftarrow 0 \\
\quad y \leftarrow \perp \\
\quad \text{while } y = \perp: \\
\qquad x \leftarrow H_p(\mathrm{ctr} \,||\, \mathrm{msg}) \\
\qquad \mathrm{ctr} \leftarrow \mathrm{ctr} + 1 \\
\qquad ySq \leftarrow x^3 + ax + b \\
\qquad y \leftarrow \mathrm{sqrt}(ySq) \qquad // \perp \text{ if } ySq \text{ is non-square} \\
\quad P \leftarrow (x, y) \\
\quad \text{return } P^h \qquad // \text{ map to } \mathbb{G} \text{ via cofactor mul}
\end{array}
$$

Not constant time; "bad" inputs are easy to find.

Loop a fixed number of times?

Hash and check

$\mathsf{HashToCurve}_{\mathsf{H\&C}}(\mathsf{msg})$:

- $\mathsf{ctr} \leftarrow 0$
- $y \leftarrow \perp$
- while $y = \perp$:
  - $x \leftarrow H_p(\mathsf{ctr} \,||\, \mathsf{msg})$
  - $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
  - $ySq \leftarrow x^3 + ax + b$
  - $y \leftarrow \mathsf{sqrt}(ySq)$ // $\perp$ if ySq is non-square
- $P \leftarrow (x, y)$
- return P^h // map to $\mathbb{G}$ via cofactor mul

Not constant time; "bad" inputs are easy to find.

✗ Loop a fixed number of times?

Slow; well-meaning "optimization" breaks CT.

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M	**Restrictions**	**Cost**
[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M	**Restrictions**	**Cost**
[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
[SW06]	none	3 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19. . .]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3, a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4, ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4, ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19. . .]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	$p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	$p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3, \quad a = 0, \quad 2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	✗ $p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	$p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	✗ $p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	$p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3$, $a = 0$, $2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	✗ $p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	✗ $p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	$b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3$, $a = 0$, $2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	✗ $p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	none	3 exp
SWU	[Ulas07]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	✗ $p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	✗ $b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3$, $a = 0$, $2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	✗ $p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	✓ none	3 exp
SWU	[Ulas07]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	✗ $p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	✗ $b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		$ab \neq 0$	1 exp
		none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3$, $a = 0$, $2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Deterministic maps to elliptic curves

$M : \mathbb{F}_p \to E(\mathbb{F}_p)$, where $E : y^2 = x^3 + ax + b$ and $p > 5$:

Map M		**Restrictions**	**Cost**
	[BF01]	✗ $p \equiv 2 \bmod 3$, $a = 0$	1 exp
	[SW06]	✓ none	3 exp
SWU	[Ulas07]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	3 exp
	[Icart09]	✗ $p \equiv 2 \bmod 3$	1 exp
S-SWU	[BCIMRT10]	✗ $p \equiv 3 \bmod 4$, $ab \neq 0$	2 exp
Elligator	[BHKL13]	✗ $b \neq 0, 2 \mid \#E(\mathbb{F}_p)$	1 exp
This work		✗ $ab \neq 0$	1 exp
		✓ none	1^+ exp

BLS12-381: $p \equiv 1 \bmod 3$, $\quad a = 0$, $\quad 2 \nmid \#E(\mathbb{F}_p)$

[SS04,Ska05,FSV09,FT10a,FT10b,KLR10,CK11,Far11,FT12,FJT13,BLMP19...]

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ // $\{0,1\}^\star \to \mathbb{F}_p$

$P \leftarrow M(t)$ // $\mathbb{F}_p \to E(\mathbb{F}_p)$

return P^h // $E(\mathbb{F}_p) \to \mathbb{G}$

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ // $\{0,1\}^\star \to \mathbb{F}_p$

$P \leftarrow M(t)$ // $\mathbb{F}_p \to E(\mathbb{F}_p)$

return P^h // $E(\mathbb{F}_p) \to \mathbb{G}$

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ // $\{0,1\}^\star \to \mathbb{F}_p$

$P \leftarrow M(t)$ // $\mathbb{F}_p \to E(\mathbb{F}_p)$

return P^h // $E(\mathbb{F}_p) \to \mathbb{G}$

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ // $\{0,1\}^\star \to \mathbb{F}_p$

$P \leftarrow M(t)$ // $\mathbb{F}_p \to E(\mathbb{F}_p)$

return P^h // $E(\mathbb{F}_p) \to \mathbb{G}$

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ $\quad // \ \{0,1\}^\star \rightarrow \mathbb{F}_p$

$P \leftarrow M(t)$ $\quad // \ \mathbb{F}_p \rightarrow E(\mathbb{F}_p)$

return P^h $\quad // \ E(\mathbb{F}_p) \rightarrow \mathbb{G}$

Possible issue: M is not a bijection: $\#E(\mathbb{F}_p) \neq p$

☞ output distribution is nonuniform

Hash functions from deterministic maps

Compose H_p and M in a natural way:

$\mathsf{HashToCurve}_{\mathsf{NU}}(\mathsf{msg})$:

$t \leftarrow H_p(\mathsf{msg})$ // $\{0,1\}^\star \to \mathbb{F}_p$

$P \leftarrow M(t)$ // $\mathbb{F}_p \to E(\mathbb{F}_p)$

return P^h // $E(\mathbb{F}_p) \to \mathbb{G}$

Possible issue: M is not a bijection: $\#E(\mathbb{F}_p) \neq p$

☞ output distribution is nonuniform

This *could* be OK—but what if we need uniformity?

Uniform hashing from deterministic maps

For uniformity [BCIMRT10,FFSTV13]:

$$\begin{aligned}
&\mathsf{HashToCurve}(\mathsf{msg}): \\
&\quad P_1 \leftarrow M(H_p(0 \,||\, \mathsf{msg})) \\
&\quad P_2 \leftarrow M(H_p(1 \,||\, \mathsf{msg})) \\
&\quad P \leftarrow P_1 \cdot P_2 \\
&\quad \text{return } P^h
\end{aligned}$$

Uniform hashing from deterministic maps

For uniformity [BCIMRT10,FFSTV13]:

HashToCurve(msg) :

$P_1 \leftarrow M(H_p(0 \,||\, \mathrm{msg}))$

$P_2 \leftarrow M(H_p(1 \,||\, \mathrm{msg}))$

$P \leftarrow P_1 \cdot P_2$

return P^h

☞ M needs to be *well distributed*: "not too lumpy"

✓ All of the M we've seen are well distributed.

Uniform hashing from deterministic maps

For uniformity [BCIMRT10,FFSTV13]:

HashToCurve(msg) :

$$P_1 \leftarrow M(H_p(0 \,||\, \mathrm{msg}))$$
$$P_2 \leftarrow M(H_p(1 \,||\, \mathrm{msg}))$$
$$P \leftarrow P_1 \cdot P_2$$
$$\text{return } P^h$$

☞ M needs to be *well distributed*: "not too lumpy"
  ✓ All of the M we've seen are well distributed.

☞ HashToCurve is *indifferentiable* from RO [MRH05]

Roadmap

1. Hash functions to elliptic curves
2. Optimizing the map of [BCIMRT10]
3. Evaluation results

The Simplified SWU map [BCIMRT10]

$$E : y^2 = f(x) = x^3 + ax + b,\ ab \neq 0.$$

Idea: pick x s.t. $f(ux) = u^3 f(x)$.

☞ For u non-square $\in \mathbb{F}_p$, $f(x)$ or $f(ux)$ is square.

The Simplified SWU map [BCIMRT10]

$$E : y^2 = f(x) = x^3 + ax + b,\ ab \neq 0.$$

Idea: pick x s.t. $f(ux) = u^3 f(x)$.

☞ For u non-square $\in \mathbb{F}_p$, $f(x)$ or $f(ux)$ is square.

$$\begin{aligned} u^3x^3 + aux + b &= u^3(x^3 + ax + b) \\ \therefore \qquad x &= -\frac{b}{a}\left(1 + \frac{1}{u^2 + u}\right) \end{aligned}$$

The Simplified SWU map [BCIMRT10]

$E : y^2 = f(x) = x^3 + ax + b$, $ab \neq 0$.

Idea: pick x s.t. $f(ux) = u^3 f(x)$.

☞ For u non-square $\in \mathbb{F}_p$, $f(x)$ or $f(ux)$ is square.

$$u^3x^3 + aux + b = u^3(x^3 + ax + b)$$

$$\therefore \qquad x = -\frac{b}{a}\left(1 + \frac{1}{u^2 + u}\right)$$

☞ If $p \equiv 3 \bmod 4$, $u = -t^2$ is non-square

The Simplified SWU map [BCIMRT10]

$$E : y^2 = f(x) = x^3 + ax + b,\ ab \neq 0.$$

Idea: pick x s.t. $f(ux) = u^3 f(x)$.

☞ For u non-square $\in \mathbb{F}_p$, $f(x)$ or $f(ux)$ is square.

$$u^3x^3 + aux + b = u^3(x^3 + ax + b)$$

$$\therefore \qquad x = -\frac{b}{a}\left(1 + \frac{1}{u^2 + u}\right)$$

☞ If $p \equiv 3 \bmod 4$, $u = -t^2$ is non-square, so:

$$X_0(t) \triangleq -\frac{b}{a}\left(1 + \frac{1}{t^4 - t^2}\right) \qquad X_1(t) \triangleq -t^2 X_0(t)$$

Evaluating the S-SWU map

$$\text{S-SWU}(t) \triangleq \begin{cases} (X_0(t), \sqrt{f(X_0(t))}) & \text{if } f(X_0(t)) \text{ is square} \\ (X_1(t), \sqrt{f(X_1(t))}) & \text{otherwise} \end{cases}$$

Evaluating the S-SWU map

$$\text{S-SWU}(t) \triangleq \begin{cases} \left(X_0(t), \sqrt{f(X_0(t))}\right) & \text{if } f(X_0(t)) \text{ is square} \\ \left(X_1(t), \sqrt{f(X_1(t))}\right) & \text{otherwise} \end{cases}$$

Attempt #1 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{\frac{p+1}{4}}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow f(x_1)^{\frac{p+1}{4}}$ // ✗ expensive

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

Evaluating the S-SWU map

$$\text{S-SWU}(t) \triangleq \begin{cases} (X_0(t), \sqrt{f(X_0(t))}) & \text{if } f(X_0(t)) \text{ is square} \\ (X_1(t), \sqrt{f(X_1(t))}) & \text{otherwise} \end{cases}$$

Attempt #1 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{\frac{p+1}{4}}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow f(x_1)^{\frac{p+1}{4}}$ // ✗ expensive

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

Evaluating the S-SWU map

$$\text{S-SWU}(t) \triangleq \begin{cases} \left(X_0(t), \sqrt{f(X_0(t))}\right) & \text{if } f(X_0(t)) \text{ is square} \\ \left(X_1(t), \sqrt{f(X_1(t))}\right) & \text{otherwise} \end{cases}$$

Attempt #1 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{\frac{p+1}{4}}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow f(x_1)^{\frac{p+1}{4}}$ // ✗ expensive

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

Evaluating the S-SWU map

$$\text{S-SWU}(t) \triangleq \begin{cases} \left(X_0(t), \sqrt{f(X_0(t))}\right) & \text{if } f(X_0(t)) \text{ is square} \\ \left(X_1(t), \sqrt{f(X_1(t))}\right) & \text{otherwise} \end{cases}$$

Attempt #1 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{\frac{p+1}{4}}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow f(x_1)^{\frac{p+1}{4}}$ // ✗ expensive

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

Requires two exponentiations! Can we do better?

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$f(x_1)^{\frac{p+1}{4}} = \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}}$$

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$\begin{aligned} f(x_1)^{\frac{p+1}{4}} &= \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}} \\ &= t^3 \left(-f(x_0)\right)^{\frac{p+1}{4}} = t^3 \sqrt{-f(x_0)} \end{aligned}$$

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$\begin{aligned} f(x_1)^{\frac{p+1}{4}} &= \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}} \\ &= t^3 \left(-f(x_0)\right)^{\frac{p+1}{4}} = t^3 \sqrt{-f(x_0)} \end{aligned}$$

☞ We have $f(x_0)^{\frac{p+1}{4}}$. Can we use this?

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$f(x_1)^{\frac{p+1}{4}} = \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}}$$
$$= t^3 \left(-f(x_0)\right)^{\frac{p+1}{4}} = t^3 \sqrt{-f(x_0)}$$

☞ We have $f(x_0)^{\frac{p+1}{4}}$. Can we use this?

$$\left(f(x_0)^{\frac{p+1}{4}}\right)^2 = f(x_0)^{\frac{p+1}{2}} = f(x_0) \cdot f(x_0)^{\frac{p-1}{2}}$$

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$
\begin{aligned}
f(x_1)^{\frac{p+1}{4}} &= \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}} \\
&= t^3 \left(-f(x_0)\right)^{\frac{p+1}{4}} = t^3 \sqrt{-f(x_0)}
\end{aligned}
$$

☞ We have $f(x_0)^{\frac{p+1}{4}}$. Can we use this?

$$
\left(f(x_0)^{\frac{p+1}{4}}\right)^2 = f(x_0)^{\frac{p+1}{2}} = f(x_0) \cdot \boxed{f(x_0)^{\frac{p-1}{2}}}
$$

Legendre symbol!

Eliminating an exponentiation

Recall: $f(x_1) = -t^6 f(x_0)$. So:

$$\begin{aligned} f(x_1)^{\frac{p+1}{4}} &= \left(-t^6 f(x_0)\right)^{\frac{p+1}{4}} \\ &= t^3 \left(-f(x_0)\right)^{\frac{p+1}{4}} = t^3 \sqrt{-f(x_0)} \end{aligned}$$

☞ We have $f(x_0)^{\frac{p+1}{4}}$. Can we use this?

$$\begin{aligned} \left(f(x_0)^{\frac{p+1}{4}}\right)^2 &= f(x_0)^{\frac{p+1}{2}} = f(x_0) \cdot f(x_0)^{\frac{p-1}{2}} \\ &= -f(x_0) \qquad \text{if } f(x_0) \text{ is non-square} \end{aligned}$$

✓ $f(x_0)^{\frac{p+1}{4}}$ is $\sqrt{-f(x_0)}$ when $f(x_0)$ is non-square!

Evaluating the S-SWU map—faster!

Attempt #2 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{(p+1)/4}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow t^3 y_0$ // ✓ cheap!

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

Evaluating the S-SWU map—faster!

Attempt #2 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{(p+1)/4}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow t^3 y_0$ // ✓ cheap!

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

✓ Prior work [BDLSY12] lets us avoid inversions.

Evaluating the S-SWU map—faster!

Attempt #2 (assume $p \equiv 3 \bmod 4$):

$x_0 \leftarrow X_0(t)$

$y_0 \leftarrow f(x_0)^{(p+1)/4}$ // ✗ expensive

$x_1 \leftarrow -t^2 x_0$ // a.k.a. $X_1(t)$

$y_1 \leftarrow t^3 y_0$ // ✓ cheap!

if $y_0^2 = f(x_0)$: return (x_0, y_0)

else: return (x_1, y_1)

✓ Prior work [BDLSY12] lets us avoid inversions.

✓ Straightforward to generalize to $p \equiv 1 \bmod 4$.

Supporting BLS12-381: the $ab = 0$ case

Issue: S-SWU still does not work with $ab = 0$.

☞ Rules out pairing-friendly curves [BLS03,BN06,...]

Supporting BLS12-381: the $ab = 0$ case

Issue: S-SWU still does not work with $ab = 0$.

☞ Rules out pairing-friendly curves [BLS03,BN06,...]

Idea: map to a curve E' having $ab \neq 0$ and an efficiently-computable homomorphism to E.

Supporting BLS12-381: the $ab = 0$ case

Issue: S-SWU still does not work with $ab = 0$.
☞ Rules out pairing-friendly curves [BLS03,BN06,...]

Idea: map to a curve E' having $ab \neq 0$ and an efficiently-computable homomorphism to E.

Specifically: Find $E'(\mathbb{F}_p)$ d-isogenous to E, d small.
☞ Defines a degree $\approx d$ rational map $E'(\mathbb{F}_p) \to E(\mathbb{F}_p)$

Supporting BLS12-381: the $ab = 0$ case

Issue: S-SWU still does not work with $ab = 0$.

☞ Rules out pairing-friendly curves [BLS03,BN06,. . .]

Idea: map to a curve E' having $ab \neq 0$ and an efficiently-computable homomorphism to E.

Specifically: Find $E'(\mathbb{F}_p)$ d-isogenous to E, d small.

☞ Defines a degree $\approx d$ rational map $E'(\mathbb{F}_p) \to E(\mathbb{F}_p)$

Then: S-SWU to $E'(\mathbb{F}_p)$, isogeny map to $E(\mathbb{F}_p)$.

✓ Preserves well-distributedness of S-SWU.

Roadmap

1. Hash functions to elliptic curves

2. Optimizing the map of [BCIMRT10]

3. Evaluation results

Implementation, baselines, setup, method

BLS12-381 defines $\mathbb{G}_1 \subset E_1(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E_2(\mathbb{F}_{p^2})$.

Implementation, baselines, setup, method

BLS12-381 defines $\mathbb{G}_1 \subset E_1(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E_2(\mathbb{F}_{p^2})$.

For $\mathbb{G}_1$ and $\mathbb{G}_2$, we implement:

Maps: hash-and-check; [SW06]; this work

Styles: full bigint; field ops only, non-CT and CT

Hashes: non-uniform; uniform

In total: 34 hash variants, 3520 lines of C.

Implementation, baselines, setup, method

BLS12-381 defines $\mathbb{G}_1 \subset E_1(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E_2(\mathbb{F}_{p^2})$.

For $\mathbb{G}_1$ and $\mathbb{G}_2$, we implement:

- Maps: hash-and-check; [SW06]; this work
- Styles: full bigint; field ops only, non-CT and CT
- Hashes: non-uniform; uniform

In total: 34 hash variants, 3520 lines of C.

Implementation, baselines, setup, method

BLS12-381 defines $\mathbb{G}_1 \subset E_1(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E_2(\mathbb{F}_{p^2})$.

For $\mathbb{G}_1$ and $\mathbb{G}_2$, we implement:

Maps: hash-and-check; [SW06]; this work

Styles: full bigint; field ops only, non-CT and CT

Hashes: non-uniform; uniform

In total: 34 hash variants, 3520 lines of C.

Setup: Xeon E3-1535M v6 (no hyperthreading or frequency scaling); Linux 5.2; GCC 9.1.0.

Implementation, baselines, setup, method

BLS12-381 defines $\mathbb{G}_1 \subset E_1(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E_2(\mathbb{F}_{p^2})$.

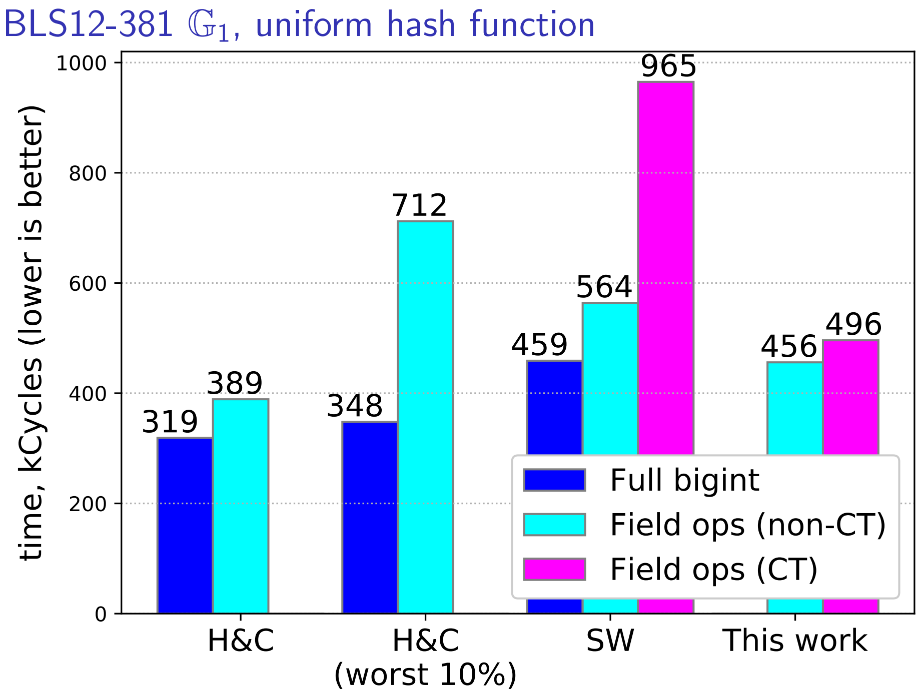
For $\mathbb{G}_1$ and $\mathbb{G}_2$, we implement:

- Maps: hash-and-check; [SW06]; this work
- Styles: full bigint; field ops only, non-CT and CT
- Hashes: non-uniform; uniform

In total: 34 hash variants, 3520 lines of C.

Setup: Xeon E3-1535M v6 (no hyperthreading or frequency scaling); Linux 5.2; GCC 9.1.0.

Method: run each hash 10^6 times; record #cycles.


BLS12-381 $\mathbb{G}_1$, uniform hash function
time, kCycles (lower is better)
0
200
400
600
800
1000
319
389
348
712
459
564
965
456
496
H&C
H&C (worst 10%)
SW
This work
Full bigint
Field ops (non-CT)
Field ops (CT)

Recap and conclusion

Contributions:

✓ Optimizations to the map of [BCIMRT10]

✓ "Indirect" approach to expand applicability

✓ Fast impls are simple and constant time

Recap and conclusion

Contributions:

✓ Optimizations to the map of [BCIMRT10]

✓ "Indirect" approach to expand applicability

✓ Fast impls are simple and constant time

Result: hash-to-curve costs 1^{+} exponentiation for essentially any prime-field elliptic curve.

Recap and conclusion

Contributions:

✓ Optimizations to the map of [BCIMRT10]

✓ "Indirect" approach to expand applicability

✓ Fast impls are simple and constant time

Result: hash-to-curve costs 1^{+} exponentiation for essentially any prime-field elliptic curve.

☞ State of the art for BLS, BN, NIST, secp256k1, and other curves not covered by Elligator or Icart.

Recap and conclusion

Contributions:

✓ Optimizations to the map of [BCIMRT10]

✓ "Indirect" approach to expand applicability

✓ Fast impls are simple and constant time

Result: hash-to-curve costs 1^{+} exponentiation for essentially any prime-field elliptic curve.

☞ State of the art for BLS, BN, NIST, secp256k1, and other curves not covered by Elligator or Icart.

`https://github.com/kwantam/bls12-381_hash`

`https://github.com/kwantam/bls_sigs_ref`

`rsw@cs.stanford.edu`