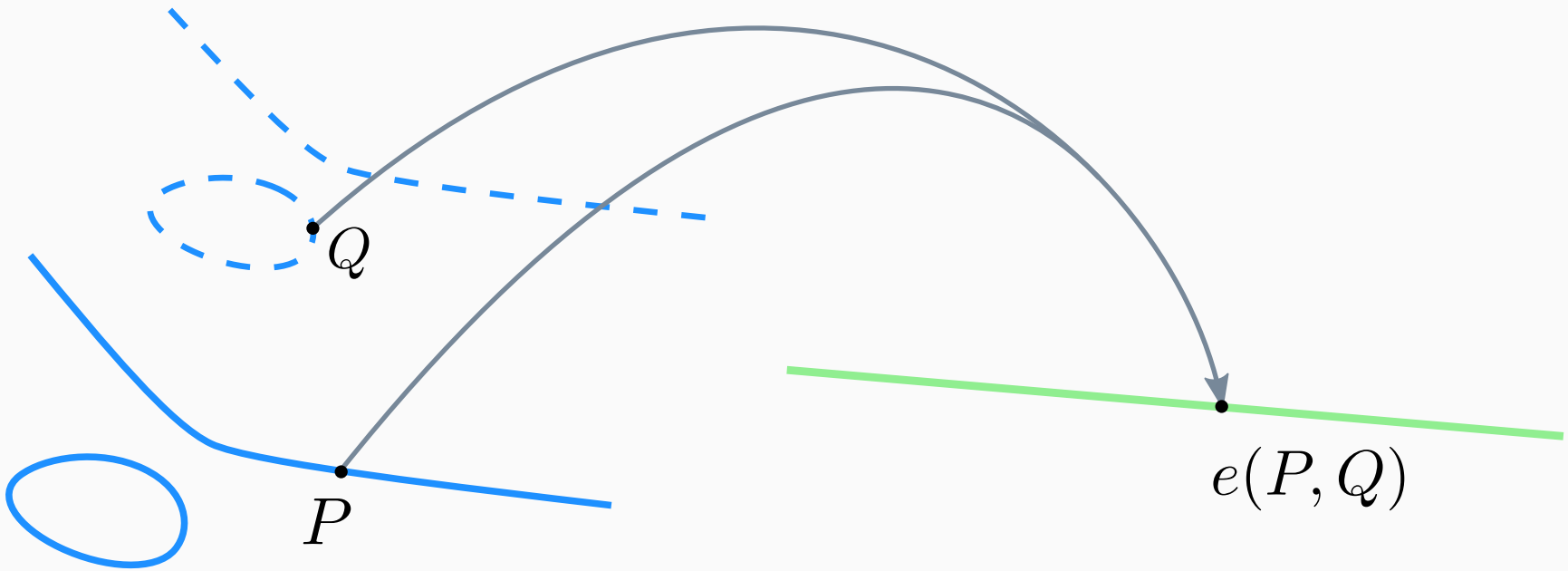
Implement all the pairings in software!

CARAMBA Seminar

Diego F. Aranha

Department of Engineering – Aarhus University

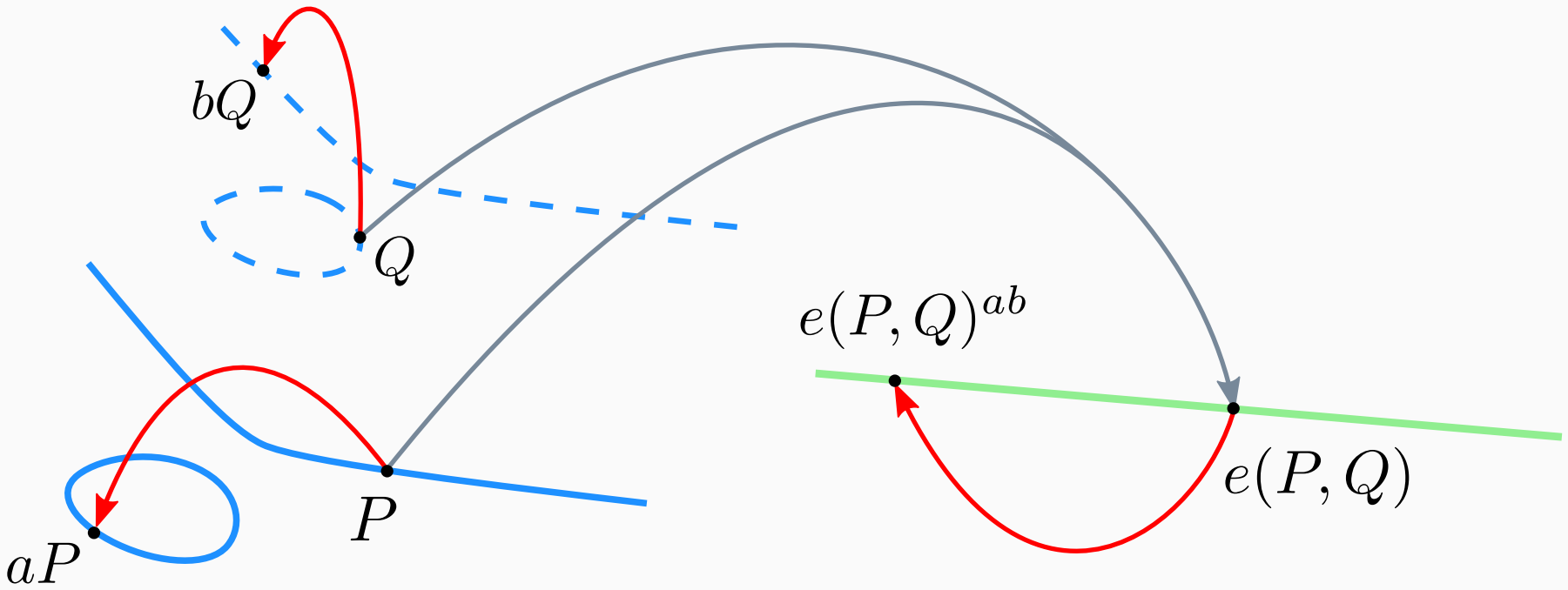
Bilinear pairings



Bilinear pairings

$$e(P+R,Q) = e(P,Q) \cdot e(R,Q) \text{ and } e(P,Q+S) = e(P,Q) \cdot e(P,S)$$

.



Introduction

Pairing-Based Cryptography (PBC) enables many elegant solutions to cryptographic problems:

- **Implicit certification schemes** (IBE, CLPKC, etc.)
- **Short signatures** (in group elements, BLS, BBS)
- **More efficient key agreements** (Joux's 3DH, NIKDS)
- **Low-depth homomorphic encryption** (BGN and variants)
- **Zero-knowledge proof systems** (LegoSNARK and Sonic)
- **Isogeny-based cryptography** (key compression and VDFs)

Not dead: Pairings are not only interesting **solely** for research, but actually deployed in practice!

Classic: IBE in Voltage's SecureMail

Implemented with supersingular curve over large characteristic [BF01].

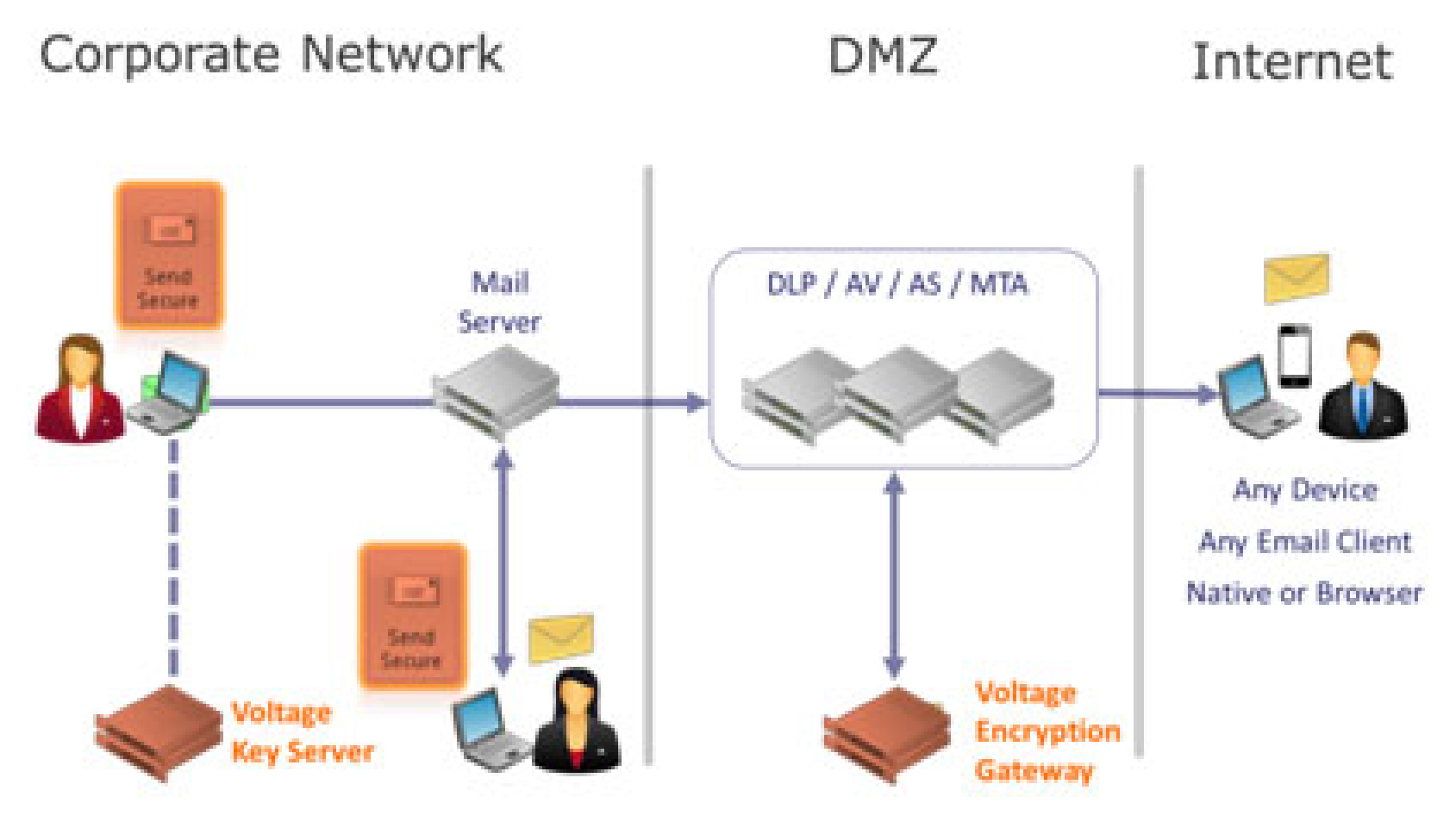


Figure 1: Source: `http://www.securemailworks.com/SecureMail.asp`

Modern applications

IBE in Cloudflare's Geo Key Manager

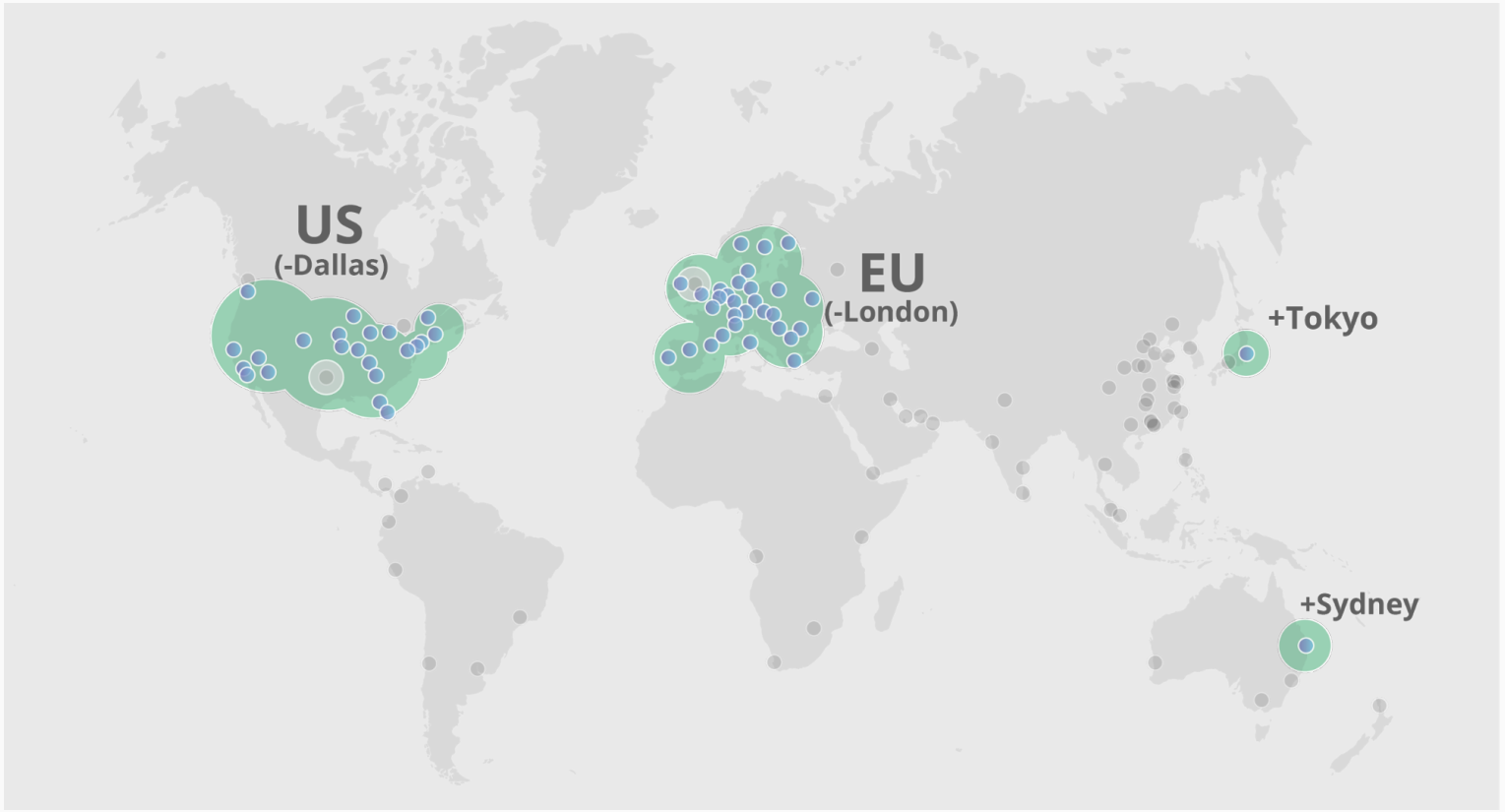


Figure 2:

`https://blog.cloudflare.com/geo-key-manager-how-it-works/`

IBE in Cloudflare's Geo Key Manager

Implemented using a 256-bit Barreto-Naehrig curve [BN05]

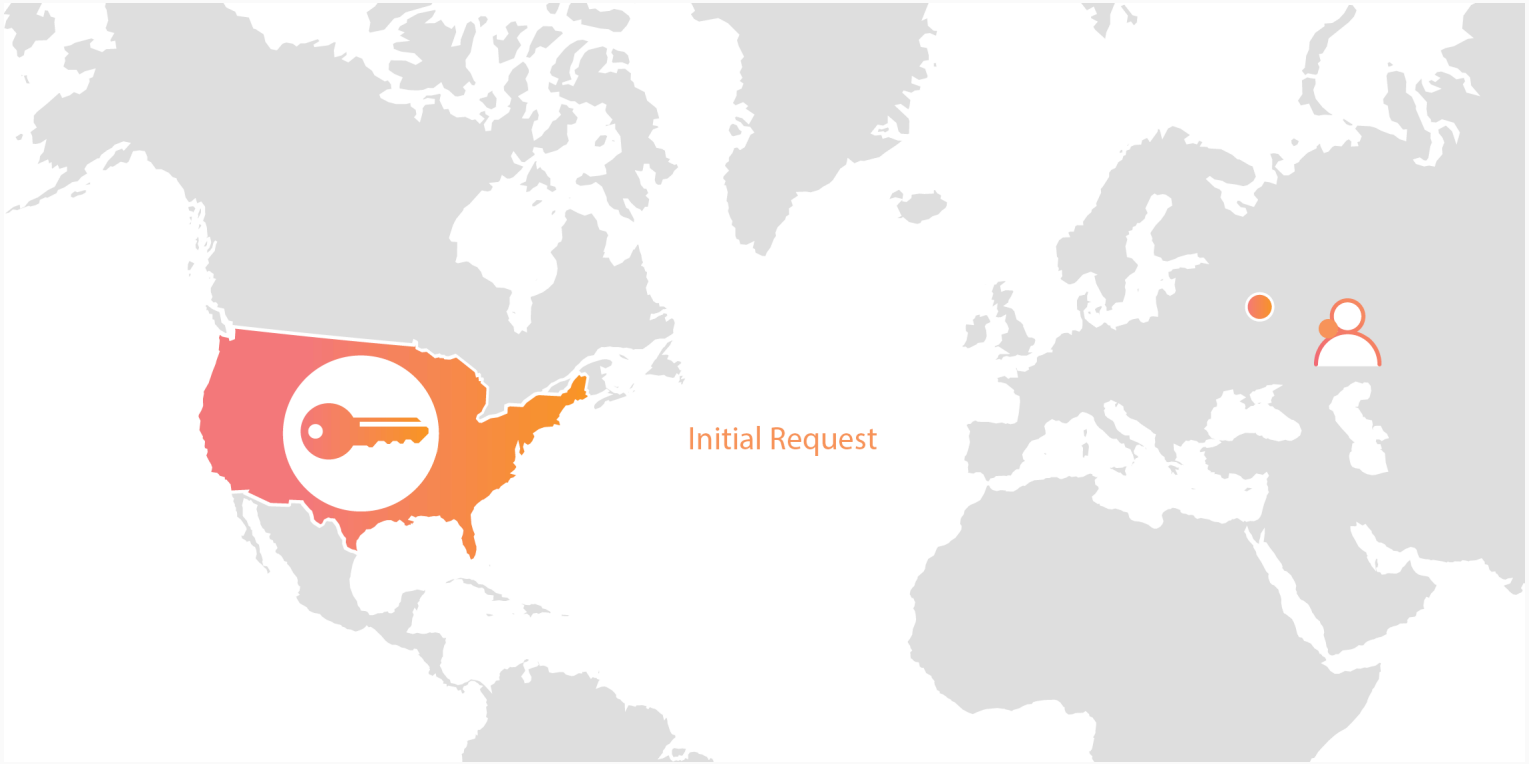


Figure 3:

`https://blog.cloudflare.com/geo-key-manager-how-it-works/`

Remote attestation in Intel SGX

Remote attestation scheme employs a pairing-based anonymous group signature by Brickell and Li (EPID) [BL12].

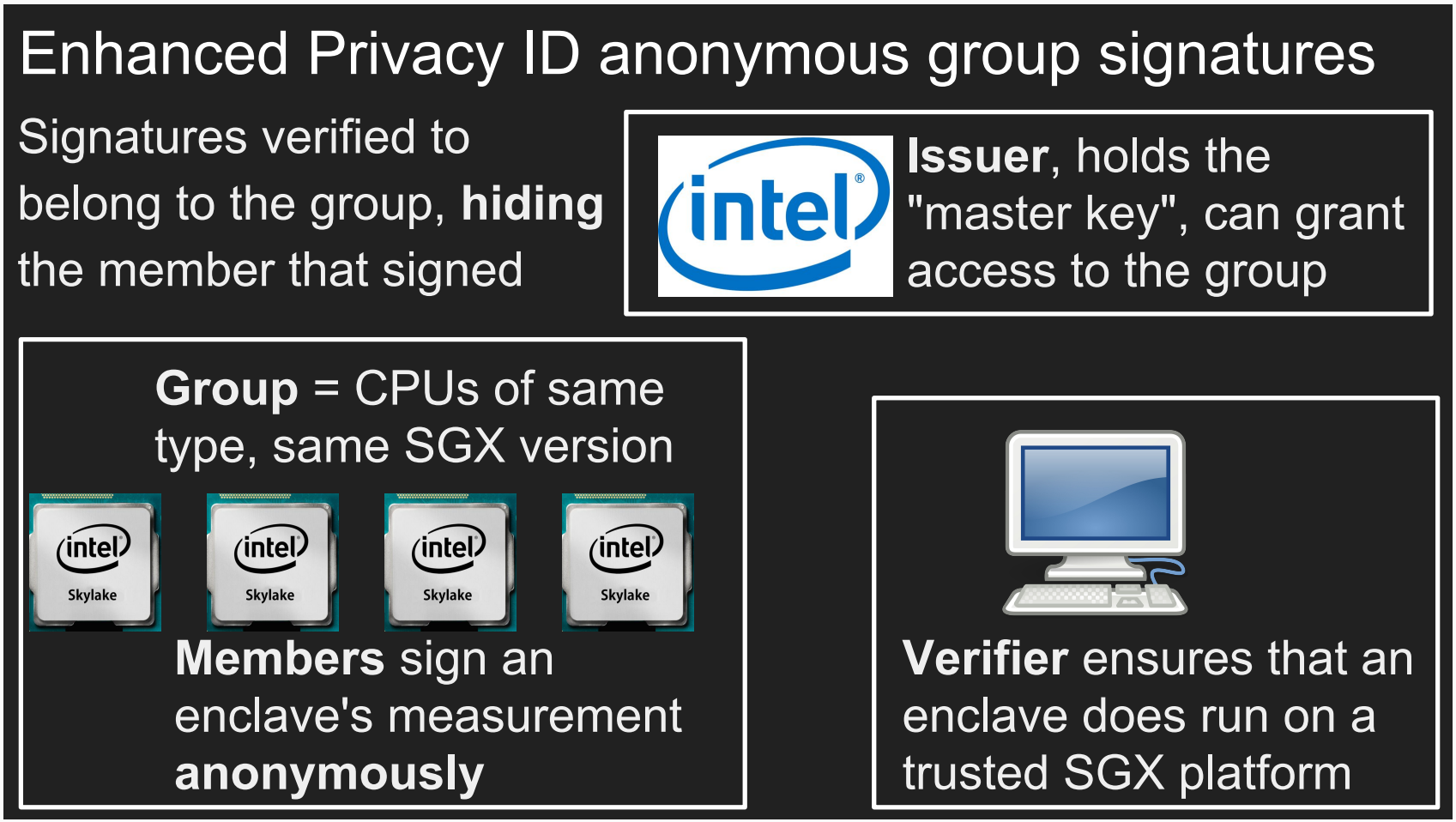


Figure 4: Slides from BlackHat 2016 talk by Aumasson and Merino [AM16].

Implemented using a 256-bit Barreto-Naehrig curve [BN05].

EPID implementation

Not in microcode, too complex

Not in SGX libs, but in the **QE and PVE binaries**

Undocumented implementation details:

- Scheme from https://eprint.iacr.org/2009/095
- Barretto-Naehrig curve, optimal Ate pairing
- Code allegedly based on https://eprint.iacr.org/2010/354

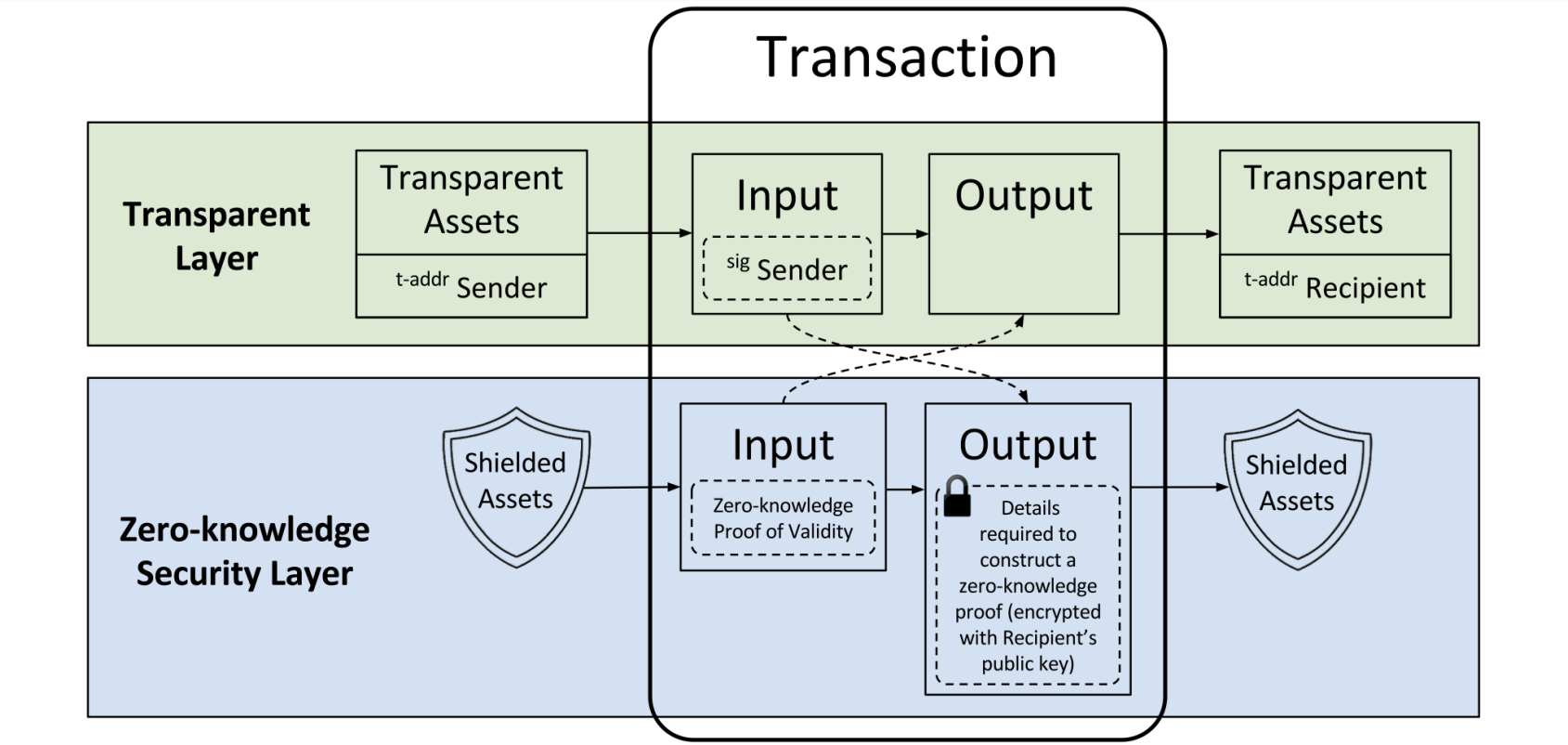
Pubkey and parameters provided by Intel Attestation Service (IAS)

```
epid_random_func
epidMember_create
epidMember_createCompressed
epidMember_delete
epidMember_registerBaseName
epidMember_computePreSignature
epidMember_join
epidMember_isPrivKeyValid
epidMember_signMessagePartial
epidMember_checkSigRLHeader
epidMember_nrProve
epidMember_signMessage
deleteEPID2Params
newEPID2ParamsFromOctStr
```

Figure 5: Slides from BlackHat 2016 talk by Aumasson and Merino [AM16].

Zcash cryptocurrency

zk-SNARKs by Ben-Sasson et al. [BCG+14] for privacy-preserving cryptocurrencies, also recently adopted by Ethereum.



Background

Pairing groups

Let $\mathbb{G}_1 = \langle P \rangle$ and $\mathbb{G}_2 = \langle Q \rangle$ be additive groups and $\mathbb{G}_T$ be a multiplicative group such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| =$ prime r.

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

- $\mathbb{G}_1$ is typically a subgroup of $E(\mathbb{F}_p)$.
- $\mathbb{G}_2$ is typically a subgroup of $E(\mathbb{F}_{p^k})$.
- $\mathbb{G}_T$ is a multiplicative subgroup of $\mathbb{F}^*_{p^k}$.

Hence pairing-based cryptography involves arithmetic in $\mathbb{F}_{p^k}$, for **embedding degree** k, the main tool used to balance security.

Pairing operations

A general pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Cryptographic schemes require multiple operations in pairing groups:

1. **Exponentiation**, **membership testing**, **compression** in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.
2. **Hashing** strings to $\mathbb{G}_1$, $\mathbb{G}_2$.
3. **Efficient maps** between $\mathbb{G}_1$ and $\mathbb{G}_2$.
4. Efficient **pairing computation**.

Curve families

At some point, pairing-based cryptography had an **explosion** of parameter choices to choose from:

BN curves: $k = 12,\ \rho \approx 1$ $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1, \quad t(x) = 6z^2 + 1$
BLS12 curves: $k = 12,\ \rho \approx 1.5$ $p(x) = (x-1)^2(x^4 - x^2 + 1)/3 + x,$ $r(x) = x^4 - x^2 + 1, \quad t(x) = x + 1$
KSS18 curves: $k = 18,\ \rho \approx 4/3$ $p(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21$ $r(x) = (x^6 + 37x^3 + 343)/343,\ \ t(x) = (x^4 + 16z + 7)/7$
BLS24 curves: $k = 24,\ \rho \approx 1.25$ $p(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x,$ $r(x) = x^8 - x^4 + 1, \quad t(x) = x + 1$

Barreto-Naehrig curves

Let $x \in \mathbb{Z}$ such that $p(x)$ and $r(x)$ are prime:

- $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$
- $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$

Then $E : y^2 = x^3 + b$, $b \in \mathbb{F}_p$ is a curve of **order** r and **embedding degree** $k = 12$ [BN05] and E' its **twist** of degree $d = 6$.

For curve BN-254, fix $x = -(2^{62} + 2^{55} + 1)$ and $b = 2$, the towering can be:

- $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 - \beta)$, where $\beta = -1$
- $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[s]/(s^2 - \epsilon)$, where $\xi = 1 + i$
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$, where $\xi = 1 + i$
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[v]/(t^3 - s)$ or $\mathbb{F}_{p^6}[w]/(w^2 - v)$

Until recently: BN curves **were** king at the 128-bit security level and got even close to **standardization** (IETF RFC).

Barreto-Naehrig curves

Instantiating pairings over BN curves had **many** performance features:

1. Implementation-friendly parameters, with **fast towering** and compact generators [GJNB11].
2. **Prime-order** group $\mathbb{G}_1$, facilitating membership testing.
3. Twist of **maximum degree**, reducing size of $\mathbb{G}_2$.
4. Gallant-Lambert-Vanstone [GLV01] **endomorphism** in $\mathbb{G}_1$.
5. Galbraith-Scott **homomorphism** [GS08] in $\mathbb{G}_2$, $\mathbb{G}_T$.
6. Compressed squarings for **exponentiation** in $\mathbb{G}_T$.

Barreto-Naehrig curves

Instantiating pairings over BN curves had **many** performance features:

1. Implementation-friendly parameters, with **fast towering** and compact generators [GJNB11].
2. **Prime-order** group $\mathbb{G}_1$, facilitating membership testing.
3. Twist of **maximum degree**, reducing size of $\mathbb{G}_2$.
4. Gallant-Lambert-Vanstone [GLV01] **endomorphism** in $\mathbb{G}_1$.
5. Galbraith-Scott **homomorphism** [GS08] in $\mathbb{G}_2$, $\mathbb{G}_T$.
6. Compressed squarings for **exponentiation** in $\mathbb{G}_T$.

Alfred Menezes, 2007

"These curves should not exist, they are too good to be true."

Updating the security of pairings

Recent results have undermined the security of pairings in some contexts:

1. **Pairings over small char**, due to many advances in the DLP, including a quasi-polynomial algorithm by Barbulescu et al. [BGJT14]. Impact: Pairings may not be that viable in **resource-constrained** devices anymore.

Updating the security of pairings

Recent results have undermined the security of pairings in some contexts:

1. **Pairings over small char**, due to many advances in the DLP, including a quasi-polynomial algorithm by Barbulescu et al. [BGJT14]. Impact: Pairings may not be that viable in **resource-constrained** devices anymore.
2. **Smooth embedding degree**, affected by Kim-Barbulescu attack on medium-prime case [KB16]. Impact: Security of BN-254 degraded to around 100 bits.
3. **Miller inversion problem**, shown to be easy for supersingular curves with $k = 2$ [Sat19]. Impact: These curves may be not just inefficient, but dangerous.

Curve families

And now we are somewhat **back** to that situation again. Recently proposed parameters, from the most conservative:

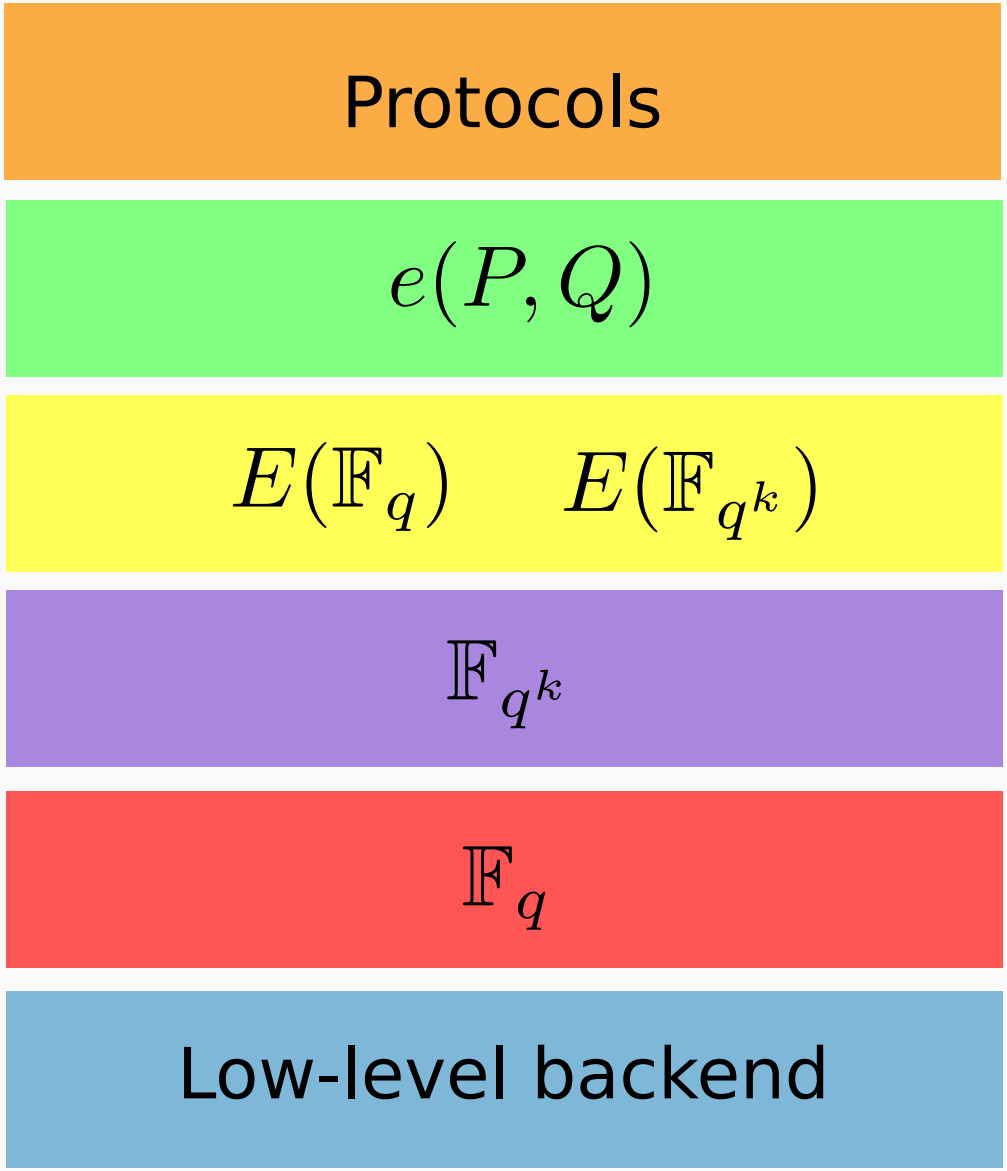
1. Elliptic curves with embedding degree $k = 1$ (**large base field**) [CMR17]
2. Symmetric pairings with prime embedding degree $k = 2, 3$ (**still large base field**) [Sco05, ZW13]
3. Elliptic curves with **less smooth** embedding degrees (ordinary with $k = 9, 13, 15, 21, 27$) [CM18, BMG19]
4. Cocks-Pinch curves with **moderate** embedding degrees [GMT19]
5. Optimal TNFS-resistant families [FM18]

→ **Adjusted** field sizes and smooth embedding degrees such as Barreto-Lynn-Scott (BLS) and Kachisa-Scott-Schaefer (KSS) curves [BLS02, KSS08].

What do we want?

Implementation techniques

Arithmetic levels



Software libraries

There are many different open-source software implementations of pairings:

- **PBC**: on top of GMP, **outdated**.
- **Panda**: not as efficient anymore, but **constant-time**.
- **Ate-pairing:** CINVESTAV, **previous** state of the art.
- **MIRACL**: special support for constrained platforms.
- **Apache Milagro**: fast C and bindings to many languages.
- **OpenPairing**: OpenSSL patch, never merged.
- **libsnark:** BN-254 and ZKPs.
- **pairing:** BLS12-381 implementation from ZCash in Rust.
- **mcl:** BN and BLS12 over multiple fields by Shigeo Mitsunari.

Software libraries

There are many different open-source software implementations of pairings:

- **PBC**: on top of GMP, **outdated**.
- **Panda**: not as efficient anymore, but **constant-time**.
- **Ate-pairing:** CINVESTAV, **previous** state of the art.
- **MIRACL**: special support for constrained platforms.
- **Apache Milagro**: fast C and bindings to many languages.
- **OpenPairing**: OpenSSL patch, never merged.
- **libsnark:** BN-254 and ZKPs.
- **pairing:** BLS12-381 implementation from ZCash in Rust.
- **mcl:** BN and BLS12 over multiple fields by Shigeo Mitsunari.

→ **RELIC**: flexible and **current** state of the art, under **heavy** development again.

Finite field arithmetic

Target platform: Desktop processor.

1. An efficient 64-bit implementation of the base field arithmetic typically employs:
   - **Montgomery** representation.
   - Wide multiplication instructions `MUL` and `MULX`.
   - **Lazy reduction**:

$$(a \cdot b) \bmod p + (c \cdot d) \bmod p = (a \cdot b + c \cdot d) \bmod p$$

2. Techniques for extension field arithmetic:
   - **Small** quadratic/cubic non-residues and **change of representation**.
   - **Fastest** formulas available in the literature (asymmetric squarings due to [CH07].
   - **General** lazy reduction: k reductions for $\mathbb{F}_{p^k}$ arithmetic [AKL+11].

Operations in $\mathbb{G}_1$ and $\mathbb{G}_2$

Scalar multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$ follow standard techniques, such as projective coordinates and signed recodings.

Scalars can be decomposed using the GLV method when **endomorphism** ψ is available: $\ell \equiv \ell_0 + \lambda\ell_1 \pmod{r} \rightarrow [\ell]P = [\ell_0]P + [\ell_1]\psi(P)$.

Hashing to $\mathbb{G}_1$ and $\mathbb{G}_2$ involves hashing to point and multiplying by **cofactor** represented in base p [SBC+09, FKR11]. More recent approaches are **indifferentiable** from random oracles [WB19, FT12].

Operations in $\mathbb{G}_T$

Pairing result is an element of the **cyclotomic subgroup** $\mathbb{G}_{\phi_k}(\mathbb{F}_{p^{k/d}})$.

Given $C(g)$, efficient to compute $C(g^2)$ as shown by Karabina in [Kar13].

Idea: $g^{|u|=2^a-2^b+1}$ can now be computed in three steps:

1. Compute $C(g^{2^i})$ for $1 \leq i \leq a$ and store $C(g^{2^b})$ and $C(g^{2^a})$
2. Compute $D(C(g^{2^a})) = g^{2^a}$ and $D(C(g^{2^b})) = g^{2^b}$
3. Compute $g^{|x|} = g^{2^a} \cdot (g^{2^b})^{k/2} \cdot g$

Remark 1: Montgomery's simultaneous inversion allows **simultaneous decompression**.

Remark 2: For dense exponent, plain cyclotomic squarings can be used instead [GS10]. **Signed recodings** can be used because inversion is **conjugation**, and base-$(t-1)$ expansions due to $g^p = g^{t-1}$.

Pairing computation

Algorithm 1 Tate pairing [BKLS02].

Input: $r = \sum_{i=0}^{\log_2 r} r_i 2^i, P, Q$.

Output: $e_r(P, Q)$.

1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
4: $\quad T \leftarrow 2T$
5: $\quad f \leftarrow f^2 \cdot l_{T,T}(Q)$
6: $\quad$ **if** $r_i = 1, i \neq 0$ **then**
7: $\quad\quad T \leftarrow T + P$
8: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)$
9: $\quad$ **end if**
10: **end for**
11: **return** $f^{(q^k - 1/r)}$

Pairing computation

A pairing computation essentially consists in the **Miller loop** followed by the **final exponentiation**.

1. An efficient implementation of the Miller loop requires:
   - **Low Hamming weight** of the integer parameter.
   - Efficient formulas for **curve arithmetic** (homogeneous coordinates).
   - Curve arithmetic combined together with computation of the **line evaluations**.

2. And the final exponentiation:
   - For even k, split the final exponent as $(p^k - 1)/\phi_k(p) \cdot \phi_k(p)/r$.
   - Easy part computed with **Frobenius**.
   - Hard part computed with decomposition in base p and **vectorial addition chain**.
   - Compressed squarings in cyclotomic subgroup.

Pairing computation

Other optimizations are possible:

1. **Optimal ate construction** to minimize integer parameter by $\phi(k)$ [Ver10].
2. **Fixed argument pairings** precomputes Miller loop when argumets are fixed [CS10].
3. **Product of pairings** to share final exponentiation when evaluating $\prod_{i=0}^{m} e(P_i, Q_i)$.

Subgroup security

A security property mandating that cofactors have only large prime factors to prevent small subgroup attacks [BCM+15]. Started as "$\mathbb{G}_T$-strong" notion of security [Sco13].

In general, **subgroup membership testing** is easy in $\mathbb{G}_1$ (validity or scalar multiplication).

In $\mathbb{G}_2$, we can exploit $n = p - t + 1$ and check if $[p]Q = [t - 1]Q$.

Subgroup security

A security property mandating that cofactors have only large prime factors to prevent small subgroup attacks [BCM+15]. Started as "$\mathbb{G}_T$-strong" notion of security [Sco13].

In general, **subgroup membership testing** is easy in $\mathbb{G}_1$ (validity or scalar multiplication).

In $\mathbb{G}_2$, we can exploit $n = p - t + 1$ and check if $[p]Q = [t-1]Q$.

Faster: protocols can be modified instead to **multiply by cofactors**.

In a subgroup-secure curve with prime $\phi_k(p)/r$, membership testing in $\mathbb{G}_T$ is easy by checking if $g^{\phi_k(p)} = 1$.

Impact: subgroup-secure curves slightly penalize pairing computation but save on membership tests.

New results

Implementation

Characteristics of the implementation:

- Target platform: Intel Skylake 64-bit processors.
- Library: RELIC is an Efficient LIbrary for Cryptography (`github.com/relic-toolkit/relic`)
- Compiler: GCC 8.3.0 with flags `-O3 -fomit-frame-point -funroll-loops`

Implementation

Characteristics of the implementation:

- Target platform: Intel Skylake 64-bit processors.
- Library: RELIC is an Efficient LIbrary for Cryptography (`github.com/relic-toolkit/relic`)
- Compiler: GCC 8.3.0 with flags `-O3 -fomit-frame-point -funroll-loops`

Comparison between two sets of parameters:

1. BN with increasing field sizes.
2. OTNFS8 vs BN-446 vs BLS12-455 curves.

BN with increasing field sizes

Parameters: BN-254 curve, Subgroup-secure BN-382, new BN-446 curve.

Operation	BN-254	BN-382	BN-446
kP in $\mathbb{G}_1$	194	553	804
kQ in $\mathbb{G}_2$	434	1501	2269
g^k in $\mathbb{G}_T$	681	2277	3786
H to $\mathbb{G}_1$ [1]	146	448	607
H to $\mathbb{G}_2$	234	746	1063
Test $\mathbb{G}_1$	0.415	0.691	0.905
Test $\mathbb{G}_2$	155	530	645
Test $\mathbb{G}_T$	260	725 [2]	1243
$e(P, Q)$ (M+F)	570+392=962	1950+1291=3241	3196+1871=5067

Table 1: Timings from RELIC in 10^3 cycles in Skylake processor measured as average of 10^4 executions (HT and TB disabled). Pairing computation is split between Miller loop (M) and Final exponentiation (F).

[1] (*) Hashing through SWU.

[2] (*) Faster test in $\mathbb{G}_{\phi_k}(\mathbb{F}_{p^{k/d}})$.

Multiple curves at "new" 128-bit security

Parameters: new 446-bit BN curve, Jacobi Quartic over 511-bit field [FM18], BLS12-455 by Mike Scott.

Operation	BN-446	OTNFS8-511	BLS12-455
kP in $\mathbb{G}_1$	804	954	680
kQ in $\mathbb{G}_2$	2269	2870	1919
g^k in $\mathbb{G}_T$	3786	-	2772
H to $\mathbb{G}_1$[3]	607	-	1104
H to $\mathbb{G}_2$	1063	-	1709
Test $\mathbb{G}_1$	0.905	827	523
Test $\mathbb{G}_2$	645	1210	798
Test $\mathbb{G}_T$	1243	-	1037
$e(P, Q)$ (M+F)	3196+1871=5067	3086+5704=8790	2379+2463=4842

Table 2: Timings from RELIC in 10^3 cycles in Skylake processor measured as average of 10^4 executions (HT and TB disabled). Pairing computation is split between Miller loop (M) and Final exponentiation (F).

[3](*) Hashing through SWU.

History of pairing implementations

Implementation	Curve	(10^6 cycles)
MOV92	Supersingular	Billions
HMS08	256-bit BN	10.0
NNS10	256-bit BN	4.38
BDM+10	256-bit BN	2.33
AKL+11	254-bit BN	1.56
M13	254-bit BN	1.16
ABLR13	254-bit BN	1.17
ECC17	254-bit BN	0.96
ECC17 (progressive)	381-bit BLS12	2.82
This work (conservative)	455-bit BLS12	4.84

Table 3: Speed records for pairing computation in the past decades.

Conclusions

Adjusting the parameters for new attacks has impacted performance of pairings **substantially**. There may be difficulties with standardization, which usually lead to **fragmentation**.

Future research:

1. Vector instructions improve the asymptotically faster **Residue Number Systems** (RNS)
2. **Optimal towerings** for newly-proposed families of curves
3. Faster exponentiation and hashing methods for alternative families of curves
4. Support to **verifiable** finite field arithmetic (Evercrypt, Fiat-Crypto) to better understand performance impact

Questions?

D. F. Aranha

dfaranha@eng.au.dk

@dfaranha

References i


Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López.
Faster explicit formulas for computing pairings over ordinary curves.
In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68. Springer, 2011.


Jean Philippe Aumasson and Luis Merino.
Sgx secure enclaves in practice: Security and crypto review.
BlackHat, 2016.


Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza.
Zerocash: Decentralized anonymous payments from bitcoin.
In *IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society, 2014.

References ii

Paulo S. L. M. Barreto, Craig Costello, Rafael Misoczki, Michael Naehrig, Geovandro C. C. F. Pereira, and Gustavo Zanon.
Subgroup security in pairing-based cryptography.
In *LATINCRYPT*, volume 9230 of *Lecture Notes in Computer Science*, pages 245–265. Springer, 2015.

Dan Boneh and Matthew K. Franklin.
Identity-based encryption from the weil pairing.
In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

References iii


Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé.
A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic.
In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2014.


Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott.
Efficient algorithms for pairing-based cryptosystems.
In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.

References iv


Ernie Brickell and Jiangtao Li.
Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities.
IEEE Trans. Dependable Sec. Comput., 9(3):345–360, 2012.


Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott.
Constructing elliptic curves with prescribed embedding degrees.
In *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, 2002.


Razvan Barbulescu, Nadia El Mrabet, and Loubna Ghammam.
A taxonomy of pairings, their security, their complexity.
Cryptology ePrint Archive, Report 2019/485, 2019.
`https://eprint.iacr.org/2019/485`.

References v


Paulo S. L. M. Barreto and Michael Naehrig.
Pairing-friendly elliptic curves of prime order.
In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.


Jaewook Chung and M. Anwar Hasan.
Asymmetric squaring formulae.
In *IEEE Symposium on Computer Arithmetic*, pages 113–122. IEEE Computer Society, 2007.


Chitchanok Chuengsatiansup and Chloe Martindale.
Pairing-friendly twisted hessian curves.
In *INDOCRYPT*, volume 11356 of *Lecture Notes in Computer Science*, pages 228–247. Springer, 2018.

References vi


Sanjit Chatterjee, Alfred Menezes, and Francisco Rodríguez-Henríquez.
On instantiating pairing-based protocols with elliptic curves of embedding degree one.
IEEE Trans. Computers, 66(6):1061–1070, 2017.


Craig Costello and Douglas Stebila.
Fixed argument pairings.
In *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2010.


Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez.
Faster hashing to $\{\mathbb G\}_2$.
In *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 412–430. Springer, 2011.

References vii


Georgios Fotiadis and Chloe Martindale.
Optimal tnfs-secure pairings on elliptic curves with even embedding degree.
Cryptology ePrint Archive, Report 2018/969, 2018.
https://eprint.iacr.org/2018/969.


Pierre-Alain Fouque and Mehdi Tibouchi.
Indifferentiable hashing to barreto-naehrig curves.
In *LATINCRYPT*, volume 7533 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2012.


C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto.
A family of implementation-friendly BN elliptic curves.
Journal of Systems and Software, 84(8):1319–1326, 2011.

References viii

Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone.
Faster point multiplication on elliptic curves with efficient endomorphisms.
In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.

Aurore Guillevic, Simon Masson, and Emmanuel Thom.
Cocks-pinch curves of embedding degrees five to eight and optimal ate pairing computation.
Cryptology ePrint Archive, Report 2019/431, 2019.
`https://eprint.iacr.org/2019/431`.

References ix


Steven D. Galbraith and Michael Scott.
Exponentiation in pairing-friendly groups using homomorphisms.
In *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2008.


Robert Granger and Michael Scott.
Faster squaring in the cyclotomic subgroup of sixth degree extensions.
In *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2010.


Koray Karabina.
Squaring in cyclotomic subgroups.
Math. Comput., 82(281):555–579, 2013.

References x

Taechan Kim and Razvan Barbulescu.
Extended tower number field sieve: A new complexity for the medium prime case.
In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 543–571. Springer, 2016.

Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott.
Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field.
In *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer, 2008.

Takakazu Satoh.
Miller inversion is easy for the reduced tate pairing on trace zero supersingular curves.
Cryptology ePrint Archive, Report 2019/385, 2019.
https://eprint.iacr.org/2019/385.

Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa.
Fast hashing to G_2 on pairing-friendly curves.
In *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2009.

Michael Scott.
Computing the tate pairing.
In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005.


Michael Scott.
Unbalancing pairing-based key exchange protocols.
IACR Cryptology ePrint Archive, 2013:688, 2013.


Frederik Vercauteren.
Optimal pairings.
IEEE Trans. Information Theory, 56(1):455–461, 2010.


Riad S. Wahby and Dan Boneh.
Fast and simple constant-time hashing to the bls12-381 elliptic curve.
Cryptology ePrint Archive, Report 2019/403, 2019.
`https://eprint.iacr.org/2019/403`.


Xusheng Zhang and Kunpeng Wang.
Fast symmetric pairing revisited.
In *Pairing*, volume 8365 of *Lecture Notes in Computer Science*, pages 131–148. Springer, 2013.