Project in Computer Security at Technion

Pairing-based Short Signatures

Final Presentation

Amit Markel ***Leonid Nemirovskiy***

Supervisor. ***Barukh Ziv***

26/10/2014

Introduction

Introduction / abstract

- Recent progress in research and practical use of elliptic curves in public key cryptography motivates us to investigate the field.

- The discrete logarithm problem on elliptic curves is yet to be solved in sub exponential time, thus we benefit from the same level of security such as one would achieve using RSA.

- We do so whilst using substantially smaller key sizes and digital signatures, hence noticeably reducing expensive network traffic load in terms of power and transmission time, as well as storage size requirements in favor of computation complexity.

Introduction / shorter signatures

- We investigate different parameters of the digital signature in quest of improvement. One of the most important vectors in this pursuit is the signature length, on which we focus our attention.

- We use elliptic curves of two compromising types:
  - one of *lesser security* parameter which provides *better speed* (approximates to 1024-bit RSA),
  - and another kind which delivers *better security* (approximates to 2048-bit RSA).

- Our signatures (160 bits) are twice as short as ECDSA standard signatures (320 bits), for security level of 80 bits. (1024 bits for RSA).

Introduction / elliptic curves & pairings

- In order to provide really short signatures, pairings are currently necessary - otherwise data loss or cryptographic strength reduction occurs.

- Establishing pairings can only be defined over elliptic curve point groups - this is due to the required algebraic properties.

Introduction / optimizations

- We implemented a library in C++ which allows to generate such short signatures.

- The library's main highlight is its simplicity: friendly user interfaces in terms of usage and comprehension.

- Some optimizations we worked on resulted in better performance than results presented on some articles.

Theoretical background

Theoretical background / elliptic curves

- We use elliptic curves over a finite field $\mathbb{F}_q$ where q is a large prime, achieving an equation of the form:

$$E: \ Y^2 = X^3 + aX + b \ \textbf{where} \ a, b \in \mathbb{F}_q,$$

  and we also use an extension of the curve with the same $a, b \in \mathbb{F}_{q^\alpha}$ for an appropriate α.

- One may define an algebraic structure on a given set of points on the curve s.t. it is an additive group. Then one works with points as ordinary group under defined operations.

Theoretical background / projective coordinates

- One should distinguish between self adding of a point - doubling, and addition of two distinct points - adding.

- We would like to focus on an optimized approach for these operations for our use - *Jacobian projective coordinates*.

- Inversion operations are costly, therefore one may define points differently with an additional dimension in order to replace such heavy computations with other operations, as follows.

$$(x, y) \mapsto (X, Y, Z) \quad \text{s.t.} \quad x = \frac{X}{Z^2},\ y = \frac{Y}{Z^3}.$$

We convert standard coordinates to projective by setting $Z = 1$.

Theoretical background / scalar multiplication

- A scalar multiplication requires many addition and doubling operations.

- We optimize this costly operation by taking advantage of the different binary representations (such as *non-adjacent form*, NAF), as well as precomputing fixed-point data to by far reduce many smaller instructions.

- By prefetching the needed intermediate values, we can completely eliminate the doubling operations.

Theoretical background / pairing

- Our bilinear *map* e must be a function of the form: $e: G_1 \times G_2 \to G_T$ where (G_1, G_2) is a pair of elliptic curve cyclic groups and G_T is an ordinary number cyclic group - this is a necessary condition for satisfying a pairing's algebraic properties.

- We investigated several options for maps: *Weil* pairing, and the more efficient *Tate* pairing.
  - *Weil* requires more fine grained calculations,
  - *Tate* demands one additional costly operation in contrast.

- Let $G = \langle g \rangle$ of prime order r, and x, y, z integers in $[0, r-1]$.
  - ***Computational Diffie-Hellman Problem*** *(CDH).*
    - Given g, g^x, g^y, compute g^{xy}.
  - ***Decisional Diffie-Hellman Problem*** *(DDH).*
    - Given g, g^x, g^y, g^z, determine whether $xy = z$.
- We mainly use *Tate*'s pairing as our bilinear map, providing an easy solution to *DDH*, yet keeping *CDH* hardness;
  - this can be exploited to generate short signatures with the same level of security as long ones.

Our Short Signature algorithm

Our short signature algorithm / global parameters

- q - the base field size.
- E - an elliptic curve over $\mathbb{F}_q$.
- p - the large prime divisor of the curve's order.
- $\mathbf{P}$ over the base field and $\mathbf{Q}$ over the extension field - are two points of order p which we precompute.
- Let $G_1 = \langle \mathbf{P} \rangle$ and $G_2 = \langle \mathbf{Q} \rangle$ and e be the pairing map.

Our short signature algorithm / key generation

Algorithm 1. Key Generation

No Input.

Output. private key $x \in \mathbb{Z}_p$, public key $\mathbf{V} \in G_2$.

1. **Set** $x \leftarrow$ **random** *integer* **in** $\mathbb{Z}_p$,
2. **Set** $\mathbf{V} \leftarrow x\mathbf{Q}$.
3. **Output** $(x, \mathbf{V})$.

Our short signature algorithm / sign

Algorithm 2. Sign

INPUT. PRIVATE KEY $x \in \mathbb{Z}_p$, MESSAGE $M \in \{0,1\}^*$.
OUTPUT. SIGNATURE $s \in \mathbb{F}_q$.

1. **Set** $\mathbf{S} \leftarrow \frac{1}{H(m)+x}\mathbf{P}$.
2. **Output** *x-coordinate* **of** $\mathbf{S}$.

Algorithm 3. Verify

INPUT. PUBLIC KEY $\mathbf{V} \in G_2$, MESSAGE $M \in \{0,1\}^*$, SIGNATURE $s \in \mathbb{F}_q$.

OUTPUT. SIGNATURE VALIDITY.

1. **Try** (**Compute** y **in** $\mathbb{F}_q$ **such that** (s,y) **on** E)
   If Invalid (**Output** *INVALID*).
2. **Set** $\mathbf{S} \leftarrow (s,y)$.
3. **If** (*order* **of** $\mathbf{S}$) $\neq p$ **then**
   (a) **Output** *INVALID*.
4. **Set precomputed** $\Omega \leftarrow \left\{e(\mathbf{P},\mathbf{Q}), (e(\mathbf{P},\mathbf{Q}))^{-1}\right\}$.
5. **Output** (**Test if** $e(\mathbf{S}, H(m)\mathbf{Q} + \mathbf{V})$ **in** Ω).

Implementation

Implementation / overview

- The project was implemented as a `C++` library, **LibECq**.

- We used `GCC 4.9.1`, `NTL 6`, `GNU GMP` and `GNU MPFR` on a *Mac UNIX 03* conforming system, having four 2.8 GHz Intel i7 cores.

Library simplicity. Example.

```
1. MNT ec;
2. ShortSignature ss(ec,true);
3. ZZ_p signature = ss.sign("Pi");
```

Implementation / performance

	Base field (160 bits)	MNT Extended (1020 bits)	Barreto & Naehring (1920 bits)
Addition	$4.110ns$	$46.209ns$	$145.739ns$
Doubling	$3.447ns$	$40.269ns$	$129.832ns$
Scalar mult	$0.783ms$	$11.221ms$	$33.175ms$
.. (fixed point)	–	$3.732ms$	$11.417ms$
Random point	$37.008ns$	$49.428ms$	$0.332s$

Implementation / performance *(2)*

We compared two approaches for the *Tate* pairing: Miller's algorithm based and via faster Elliptic Nets.

	MNT (170 bits)	Barreto & Naehring (160 bits)
Initialization	$0.246s$	$0.799s$
Tate-Miller	$59.939ms$	$333.708ms$
Tate-Nets	**6.963ms**	**29.578ms**
Key generation	$3.956ms$	$11.714ms$
Sign	$0.539ms$	$0.488ms$
Verify	$11.126ms$	$41.932ms$

Remark. Some of our results are better than some presented in other articles. For example, one of the articles' execution time for computing Tate pairing via elliptic nets, using similar comparable parameters, takes about $130ms$, in contrast to our $29.578\,ms$ time.

Conclusions

Conclusions / our algorithm modifications

- ***Miller algorithm***. We applied Jacobian projective coordinates as well denominator accumulating, for reducing division operations notably.

- ***ZSS Short Signature Algorithm***. We extended the base underlying group with two different ones, along with pre-computation of various constant pairing values for enhanced speedups.

- ***Final exponentiation of Tate pairing***. We altered the strategy by adding an additional step in lieu of using a default exponentiation in the final stage.

WHAT WE HAVE LEARNT

We gained useful experience and knowledge of real-world algorithms and topics as well as interesting abstract mathematical concepts, which contribute to the understanding of our implementation thus allowing us to tweak and enhance it significantly.

Thank you.