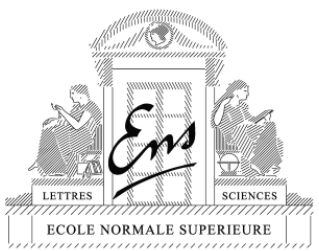
www.thalesgroup.com

Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves

Aurore Guillevic

C2, Dinard, France

THALES



Outline

Outline

Previous work

- *[Boneh, Goh and Nissim, TCC 2005]* First public-key homomorphic encryption scheme using composite-order groups and pairings
- Based on the Subgroup Decision Assumption
- For the last seven years, many protocols with interesting properties based on this assumption
- *[Freeman, Eurocrypt 2010]* Specific conversions to prime-order groups
- *[Lewko, Eurocrypt 2012]* Generic conversions to prime-order groups and nice security proofs

$\rightarrow$ It remains quite theoretical

Contributions

- Explicit parameter sizes for protocols based on the Subgroup Decision Assumption at common security levels
- Implementation in C and benchmarks

Outline

Subgroup Decision Assumption

given a group $\mathbb{G}$ of composite order $p_1 p_2 = N$ (e.g. an RSA modulus) without knowing its decomposition into p_1 and p_2, it is hard to decide whether a given element $g \in \mathbb{G}$ is in the subgroup of order p_1.

N must be infeasible to factor $\Rightarrow$ very large parameter sizes.

Bilinear Groups

1. $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of order N
2. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map i.e. for all $g_1, h_1 \in \mathbb{G}_1$ and $g_2, h_2 \in \mathbb{G}_2$, $e(g_1 \cdot h_1, g_2) = e(g_1, g_2) \cdot e(h_1, g_2)$ and $e(g_1, g_2 \cdot h_2) = e(g_1, g_2) \cdot e(g_1, h_2)$.

2'. for all $a, b \in \mathbb{Z}, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$.

BGN protocol: setup

1. Generate two random τ-bit primes p_1, p_2 and set $N = p_1 p_2$.
2. Generate a (symmetric) bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ with $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N.
3. Pick two random generators $g_1, u_1 \leftarrow \mathbb{G}_1$ and set $h_1 = u_1^{p_2} \Rightarrow h_1$ is a random generator of the subgroup of order p_1 of $\mathbb{G}_1$. Set $g_T = e(g_1, g_1)$ as generator of $\mathbb{G}_T$ and $h_T = e(g_1, h_1) = g_T^{p_2}$ as generator of the subgroup of order p_1 of $\mathbb{G}_T$.
4. $\mathcal{PK} = (N, \mathbb{G}_1, \mathbb{G}_T, e, g_1, h_1, g_T, h_T)$. $\mathcal{SK} = p_1$.

BGN protocol: encrypt/decrypt

Encrypt$(\mathcal{PK}, m)$: $m \in \mathbb{N}, m < p_2$. Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$. The ciphertext is

$$c = g_1^m \cdot h_1^r \in \mathbb{G}_1 \ .$$

Decrypt$(\mathcal{SK}, c \in \mathbb{G}_1)$: We have

$$c^{p_1} = (g_1^m \cdot h_1^r)^{p_1} = (g_1^{p_1})^m$$

$\rightarrow$ compute the discrete log of c^{p_1} in base g^{p_1}.
$\rightarrow$ very slow,
$\rightarrow$ or m must be very small (few bits).

BGN protocol: homomorphic add/mul

Add(c_1, c_2) **mod** N Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$.

$$c = c_1 \cdot c_2 \cdot h_1^r = g_1^{m_1+m_2 \mod N} \cdot h_1^{r'} \in \mathbb{G}_1 \ .$$

Mul(c_3, c_4) **mod** N (once) Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$.

$$c = e(c_3, c_4) \cdot h_T^r = g_T^{m_3 \cdot m_4 \mod N} \cdot h_T^{r'} \in \mathbb{G}_T \ .$$

Add(c_5, c_6) **mod** N Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$.

$$c = c_5 \cdot c_6 \cdot h_T^r = g_T^{m_5+m_6 \mod N} \cdot h_T^{r'} \in \mathbb{G}_T \ .$$

BGN in practice

- Suitable elliptic curve: easy to generate
  1. Let N a composite-order modulus (generated with e.g. `openssl`).
  2. Find the smallest integer h, $4 \mid h$, such that $hN - 1$ is prime.
  3. Let $p = hN - 1$. The elliptic curve $E(\mathbb{F}_p) : y^2 = x^3 - x$ is supersingular, of order $hN = p + 1$ and embedding degree 2.

  Moreover, an explicit isomorphism $\mathbb{G}_1 \to \mathbb{G}_2$ is available, hence the pairing is symmetric.
- Tate pairing only (one of the worst pairings in speed).
- Parameter sizes: $3072 \leqslant \log N \leqslant 3248$ (NIST–Ecrypt).

BGN variants

- Over composite-order groups made of several distinct primes
- Each information is hidden in a subgroup
- The parameter sizes depend on the Number Field Sieve (NFS) attack and the Elliptic Curve Method (ECM) attack

Outline

Freeman and Lewko Conversions

- As operations are much faster on a prime-order elliptic curve than a composite-order one, the protocol is built on this prime-order curve.
- It uses a vector of elements in the same prime-order group: each copy of the prime-order group corresponds to a subgroup in the composite-order setting.
- To distinguish between the different copies, elements are generated from different generators.
- The protocol security relies on the d-Linear Problem, an extension of the Diffie-Hellman Problem.
- New properties to achieve: projecting pairings and cancelling pairings.

BGN Conversion: setup

1. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ three cyclic groups of prime order n with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
2. Let $\mathbf{G}_1 = \mathbb{G}_1^2$, $\mathbf{G}_2 = \mathbb{G}_2^2$, $\mathbf{G}_T = \mathbb{G}_T^4$.
3. Choose random generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and let $g_T = e(g_1, g_2)$.
4. Choose random $\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}, \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \in SL_2(\mathbb{F}_n)$.
5. Let $\mathbf{H}_1$ be the subgroup of $\mathbf{G}_1$ generated by $\mathbf{h_1} = (g_1^{a_1}, g_1^{b_1})$, let $\mathbf{H}_2$ be the subgroup of $\mathbf{G}_2$ generated by $\mathbf{h_2} = (g_2^{a_2}, g_2^{b_2})$.
6. Define a pairing $\mathbf{e} : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$ by $\mathbf{e}([u_1, v_1], [u_2, v_2]) = [e(u_1, u_2), e(u_1, v_2), e(v_1, u_2), e(v_1, v_2)]$.
7. Let $\mathbf{H}_T = \langle \mathbf{e}(\mathbf{h_1}, \mathbf{h_2}), \mathbf{e}(\mathbf{h_1}, [g_2^{c_2}, g_2^{d_2}]), \mathbf{e}([g_1^{c_1}, g_1^{d_1}], \mathbf{h_2}) \rangle \subset \mathbf{G}_T$.

BGN Conversion: setup

8. The analogous of computing c^{p_1} is computing a projecting map π_1, π_2 or π_T from the groups $\mathbf{G}_1$, $\mathbf{G}_2$ or $\mathbf{G}_T$ s.t. $\mathbf{H}_* \subset \mathrm{Ker}(\pi_*)$ and $\mathbf{e}(\pi_1(\mathbf{u}), \pi_2(\mathbf{v})) = \pi_T(\mathbf{e}(\mathbf{u}, \mathbf{v}))$.
9. The public parameters are $(\mathbf{G}_1, \mathbf{H}_1, \mathbf{G}_2, \mathbf{H}_2, \mathbf{G}_T, \mathbf{G}_T', \mathbf{e}, g_1, g_2)$ and the secret trapdoors are π_1, π_2 and π_T which need the numbers $a_1, b_1, \ldots, d_2$, e.g. $\pi_1([u_1, v_1]) = [u_1^{-b_1 c_1} \cdot v_1^{a_1 c_1},\ u_1^{-b_1 d_1} \cdot v_1^{a_1 d_1}]$.

Homomorphic properties:

- The message is hidden in the exponent and a random blinding term from the subgroup is added.
- To decrypt, a discrete logarithm is needed.

Outline

LibCryptoLCH

The LibCryptoLCH is a proprietary cryptographic library developed inside the Crypto Lab (Laboratoire Chiffre) at THALES.

- $\mathbb{F}_p$ arithmetic with Montgomery representation
  - multiplication in Intelx86 assembly language can be activated (thanks to F. de Portzamparc)
- same high-level pairing optimizations as in the most efficient papers
- **generic design**: may use any p or elliptic curve
- modular approach

Curve, Pairing	k	$\log_2 m$	$\log_2 p$	Miller L.	F. Exp.	Pairing
Ssingular, Tate	2	256	1536	19.7 ms	20.5 ms	40.2 ms
BN, Opt. Ate	12	256	256	2.4 ms	3.0 ms	5.4 ms

Pairings with normal NIST parameter sizes for a 128-bit security level, PC Linux Intel x86, 2.6 GHz

Results

Operation	Composite-order EC		Prime-order EC		×
Enc,Add	1 exp. in $\mathbb{G}_1$	600 ms	1 exp. in $\mathbb{G}_1$ and $\mathbb{G}_2$	2.5 ms	240
Decrypt	$C^{p_1} \in \mathbb{G}_1$	300 ms	π_1: 4 exp. in $\mathbb{G}_1$ π_2: 4 exp. in $\mathbb{G}_2$	2.4 ms 7.6 ms	120 40
Multiply	1 pairing + 1 exp. in $\mathbb{G}_T$	1470 ms	1 exp. in $\mathbb{G}_1$, $\mathbb{G}_2$, 4×(3 pairings)	43.3 ms	34
Enc,Add	1 exp. in $\mathbb{G}_T$	170 ms	1 exp. in $\mathbb{G}_1$, $\mathbb{G}_2$ 4×(2 pairings)	33.7 ms	5
Decrypt	$C^{p_1} \in \mathbb{G}_T$	84 ms	$\pi_t(C)$ 16 exp. in $\mathbb{G}_T$	64.0 ms	1.3

Outline

Conclusion

- BGN introduced very nice properties and is practical but quite slow on a PC
- Conversions in the prime-order setting provide much faster timings

→ Add in $\mathbb{G}_1$ is 240 times faster, Mul is 34 times faster and Add in $\mathbb{G}_T$ 5 times faster.

Thank you for your attention.

aurore.guillevic@thalesgroup.com