An Introduction to Homomorphic Encryption for Statistics and Machine Learning

Louis J. M. Aslett (aslett@stats.ox.ac.uk)

Department of Statistics, University of Oxford

Warwick Algorithms Seminar
20 November 2015

i-like.org.uk

Outline

1. Standard Encryption
   - Discussion of encryption concepts to set the scene.
2. Homomorphic Encryption
   - Definition and high level discussion of homomorphic schemes.
3. Fan & Vercauteren (2012)
   - In depth look at this specific homomorphic encryption scheme.
   - Some further discussion on polynomial Chinese remainder Theorem.
4. Software
   - Discussion of implementation issues and `HomomorphicEncryption` R package.
5. Machine Learning
   - Novel encrypted random forest — joint with Pedro Esperança & Chris Holmes.

Standard Encryption

i-like.org.uk

Encryption basics (I)

Broadly speaking, an encryption scheme consists of:

- Unencrypted object, $m \in M$, referred to as a *message*.
    - M is the *message space*.
- Encrypted version, $c \in C$, referred to as a *cipher text*.
    - C is the *cipher text space*.
- Single $(k_s) \in K_s$, or pair $(k_s, k_p) \in K_s \times K_p$, of 'keys'.
    - Single key means secret key scheme;
    - Pair of keys means public key scheme.
- Injective map, $\mathtt{Enc} : K_p \times M \to C$.
    - not necessarily a function, message can encrypt to different cipher texts.
- Surjective function, $\mathtt{Dec} : K_s \times C \to M$.
- Enc and Dec satisfy:

$$m = \mathtt{Dec}(k_s, \mathtt{Enc}(k_p, m)) \quad \forall\, m \in M$$

i-like.org.uk

Encryption basics (II)

Fundamental point is ...

Easy

$$\texttt{Enc}(k_p, m) \rightleftharpoons c$$

Hard without k_s

$$\texttt{Dec}(k_s, c) = m$$

The *security level* of an encryption scheme is the order of the number of operations required to crack it (decrypt without k_s).

Clearly, an upper bound on the security of an encryption scheme is $O(|K_s|)$, since a brute force attack which tries every possible secret key will succeed.

Concepts: Public key -vs- private key

Presumably public key schemes are always better: can just choose not to distribute k_p?

Not really. Public key schemes tend to:

- have much larger cipher texts than messages, so are space inefficient.
- have greater computational cost, so are compute inefficient.
- rely on complex mathematical constructions rather than bit-level operations, so are hard to design custom hardware for.

Hence, private key schemes still involved in almost all cryptography, perhaps wrapped in a public key scheme. More anon …

Concepts: Semantic security

Definition (Semantic security)

An encryption scheme is said to be *semantically secure* if knowledge of the cipher text for some message has vanishingly small probability of revealing further information about any other encrypted message.

Informally: repeated encryption of same message renders different and seemingly unrelated cipher texts with high probability.

Why do we care? For private key scheme you don't. However, in a public key scheme where $|M|$ is small or probable messages are known, an attacker can perform a 'chosen plaintext attack' if not semantically secure — simply encrypt using the public key and compare.

Some common schemes (history, I)

- DES or Triple-DES. Secret-key scheme with 56-bit keys.
    - DES: block cipher algorithm … bit fiddling transformations which incorporate key.
    - TDEA: $\texttt{Enc}(., m) := \texttt{Enc}(k_{s3}, \texttt{Dec}(k_{s2}, \texttt{Enc}(k_{s1}, m)))$.
- RSA. Famously the first practical public-key scheme, based on prime number pairs.
    - $k_p = (n, e)$ where:
        - $n = pq$ for p, q prime;
        - e integer, $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$
        - Note, $\phi(n) = \phi(p)\phi(q)$
        - $\texttt{Enc}(k_p, m) := m^e \mod n$
    - $k_s = (d)$ where $d = e^{-1} \mod \phi(n)$
    - $\texttt{Dec}(k_s, c) := c^d \mod n$

Some common schemes (history, II)

PGP. Arguably first encryption software popular with regular users.

- Uses RSA to encrypt a Triple-DES key
- Uses Triple-DES to encrypt a compressed version of message

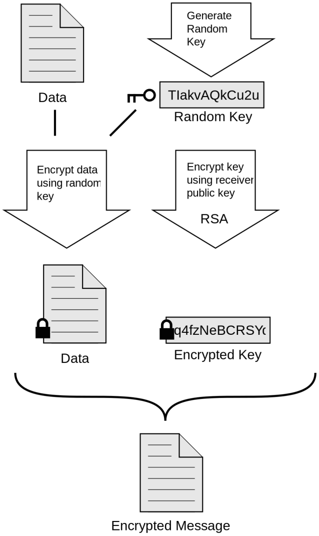


Image by xaedes & jfreax & Acdx [CC BY-SA 3.0]

Some common schemes (today)

- AES (Advanced Encryption Standard). Secret-key scheme which has superceded DES and Triple-DES. Now an industry standard.
  - Use wifi with WPA2? All traffic encrypted with AES unless you use TKIP for backwards compatability.
  - Own an iPhone/iPad? The internal flash storage is automatically encrypted using 256-bit AES.
  - Most Intel CPUs since 2010 include hardware AES acceleration.
  - Required for US federal encryption since 2014.
  - Brute force attacks on AES-128 require 2 billion years running 1 trillion machines capable of testing 1 billion keys a second.
- TLS/SSL. Every time you visit a secure website.
  - RSA typically still used to verify identity and exchange secret key.
  - Triple-DES or AES used to encrypt the webpage content.

Problem: ‘Brittle’ encryption

Most cryptography schemes are ‘brittle’ in that we can’t manipulate the message contained in the mathematical vault: must decrypt to compute, then encrypt the result. i.e. seems only useful for shipping round static data!

In other words, if

$$\begin{aligned} c_1 &:= \mathsf{Enc}(k_p, m_1) \\ c_2 &:= \mathsf{Enc}(k_p, m_2) \end{aligned}$$

then in general, for a given function $g(\cdot,\cdot)$, $\nexists\, f(\cdot,\cdot)$ (not requiring k_s) such that

$$\mathtt{Dec}(k_s, f(c_1, c_2)) = g(m_1, m_2) \quad \forall\, m_1, m_2 \in M$$

Homomorphic Encryption

Introduction

Rivest et al. (1978) hypothesised that a limited set of functions may be possible to compute encrypted: specifically those involving addition and multiplication.

Definition (Homomorphic encryption scheme)

An encryption scheme is said to be *homomorphic* if there is a set of operations $\circ \in \mathcal{F}_M$ acting in message space (such as addition) that have corresponding operations $\diamond \in \mathcal{F}_C$ acting in cipher text space satisfying the property:

$$\mathtt{Dec}(k_s, \mathtt{Enc}(k_p, m_1) \diamond \mathtt{Enc}(k_p, m_2)) = m_1 \circ m_2 \quad \forall\, m_1, m_2 \in M$$

A scheme is *fully homomorphic* if $\mathcal{F}_M = \{+, \times\}$ and an arbitrary number of such operations are possible.

The first fully homomorphic scheme was not found until Gentry (2009)

RSA as a homomorphic scheme

Recall RSA from the introduction: it is in fact a homomorphic encryption scheme!

$$\mathcal{F}_M = \{\times\}, \mathcal{F}_C = \{\times\}$$

$$\begin{aligned}\mathsf{Enc}(k_p, m_1) \times \mathsf{Enc}(k_p, m_2) &= (m_1^e \mod n) \times (m_2^e \mod n) \\ &= (m_1 m_2)^e \mod n \\ &= \mathsf{Enc}(k_p, m_1 m_2)\end{aligned}$$

Final equality indicates a lack of semantic security, so actually RSA is not great when we want to encrypt plain old integer data as it will be very vulnerable to chosen plaintext attack.

Why $+$ and $\times$?

Addition and multiplication seem pretty limiting, why all the excitement if this is all that is possible?

Note that if $M = \mathrm{GF}(2)$, then:

- $+ \equiv \veebar$, i.e. XOR, 'exclusive or'
- $\times \equiv \wedge$, i.e. AND, 'and'

Moreover, *any* electronic logic gate can be constructed using only XOR and AND gates. Therefore, theoretically any operation on a computer can be performed encrypted.

Limitations of homomorphic encryption

1. Message space
   - Commonly only easy to encrypt binary/integers
2. Cipher text size
   - Present schemes all inflate the size of data substantially (e.g. 1MB $\rightarrow$ 16.4GB)
3. Computational cost
   - 1000's additions per sec
   - $\approx$ 50 multiplications per sec
4. Division and comparison operations
   - Impossible!
5. Depth of operations
   - After a certain depth of multiplications, need to 'refresh' cipher text: hugely time consuming, so avoid!

‘Bootstrap’ — cipher text refreshing

Unrelated to statistical term ‘bootstrap’.

See in next section, operations with cipher texts in a semantically secure scheme increase noise component. After some number of operations noise will overwhelm the message.

Breakthrough by Gentry (2009) was constructing decryption algorithm simple enough to itself run encrypted.

Essentially, if you can do (v loosely speaking):

$$c' = \texttt{Dec}(\texttt{Enc}(k_p, k_s), c)$$

then c' will be a cipher text representing the same message as c, but with noise level reset to a fresh cipher text.

Fan & Vercauteren (2012)

Fan & Vercauteren (2012) scheme : notation

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively

i-like.org.uk

Fan & Vercauteren (2012) scheme : notation

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively
- $\Phi_n(x)$ is nth cyclotomic polynomial
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$

Cyclotomic polynomials

Definition (Cyclotomic polynomial)

For any positive integer n, the nth *cyclotomic polynomial* is

$$\Phi_n(x) := (x - \omega_1)(x - \omega_2) \dots (x - \omega_n)$$

where $\omega_1, \dots, \omega_n$ are the primitive nth roots of unity, $\omega_k := e^{\frac{2\pi i}{n}k}$

Equivalently and less formally, the nth cyclotomic polynomial is the polynomial which:

- divides $x^n - 1$;
- does not divide $x^m - 1$ for any $m < n$;
- has integer coefficients;
- and is irreducible (cannot be factorised).

Fan & Vercauteren (2012) scheme : notation (cont'd)

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively
- $\Phi_n(x)$ is nth cyclotomic polynomial
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$

i-like.org.uk

Fan & Vercauteren (2012) scheme : notation (cont'd)

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively
- $\Phi_n(x)$ is nth cyclotomic polynomial
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$
- Interest in elements of polynomial ring $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$
- Polynomials written $\underline{a}$ or $a(x)$

Fan & Vercauteren (2012) scheme : notation (cont'd)

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively
- $\Phi_n(x)$ is nth cyclotomic polynomial
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$
- Interest in elements of polynomial ring $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$
- Polynomials written $\underline{a}$ or $a(x)$
- $\underline{a} \sim R_q \implies$ uniform random draw from R_q
- $\underline{a} \sim \chi \implies$ discrete multivariate Gaussian draw in R_q

i-like.org.uk

Fan & Vercauteren (2012) scheme : notation (cont'd)

- $\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
- $[a]_q$ is unique integer in $\mathbb{Z}_q$ st $[a]_q = a \mod q$
- $\mathbb{Z}[x], \mathbb{Z}_q[x]$ denote polynomials with coefficients in $\mathbb{Z}$ and $\mathbb{Z}_q$ respectively
- $\Phi_n(x)$ is nth cyclotomic polynomial
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$
- Interest in elements of polynomial ring $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$
- Polynomials written $\underline{a}$ or $a(x)$
- $\underline{a} \sim R_q \implies$ uniform random draw from R_q
- $\underline{a} \sim \chi \implies$ discrete multivariate Gaussian draw in R_q

Messages $m(x) \in M \triangleq R_t$

Cipher texts $c \in C \triangleq R_q \times R_q$

Fan & Vercauteren (2012) scheme : setup

- **Parameters**
  - d, degree of both the polynomial rings M and C
  - t and q, coefficient sets of polynomial rings M and C
  - σ, magnitude of the discrete Gaussian randomness for semantic security

i-like.org.uk

Fan & Vercauteren (2012) scheme : setup

- **Parameters**
  - d, degree of both the polynomial rings M and C
  - t and q, coefficient sets of polynomial rings M and C
  - σ, magnitude of the discrete Gaussian randomness for semantic security
- **Key generation**
  - Secret key:

    $$\underline{k}_s \sim R_2$$

    (i.e. sample a 2^{d-1} binary vector for the polynomial coefficients).
  - Public key:

    $$k_p := ([-(\underline{a} \cdot \underline{k}_s + \underline{e})]_q, \underline{a})$$

    where $\underline{a} \sim R_q$ and $\underline{e} \sim \chi$.
    ($\underline{k}_s$ hard to extract due to ring LWE hardness, see Lyubashevsky et al. 2010)

i-like.org.uk

Fan & Vercauteren (2012) : encryption/decryption

- **Encode**
  Need $m \in \mathbb{Z}$ expressed as polynomial ring element. Write in b-bit binary representation, $m = \sum_{n=0}^{b-1} a_n 2^n$, then construct $\mathring{m}(x) = \sum_{n=0}^{2^{d-1}-1} a_n x^n \in R_t$ where $a_n = 0 \; \forall \; n \geq b$.

Fan & Vercauteren (2012) : encryption/decryption

- **Encode**
  Need $m \in \mathbb{Z}$ expressed as polynomial ring element. Write in b-bit binary representation, $m = \sum_{n=0}^{b-1} a_n 2^n$, then construct $\mathring{m}(x) = \sum_{n=0}^{2^{d-1}-1} a_n x^n \in R_t$ where $a_n = 0 \; \forall \, n \geq b$.
- **Encryption** $\mathsf{Enc}(k_p, m)$
  First encode $m \in \mathbb{Z}$ as $\underline{\mathring{m}} \in R_t$

$$c := ([\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}}]_q, [\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2]_q)$$

where $\underline{u}, \underline{e}_1, \underline{e}_2 \sim \chi$ and $\Delta = \left\lfloor \frac{q}{t} \right\rfloor$.

i-like.org.uk

Fan & Vercauteren (2012) : encryption/decryption

- **Encode**
  Need $m \in \mathbb{Z}$ expressed as polynomial ring element. Write in b-bit binary representation, $m = \sum_{n=0}^{b-1} a_n 2^n$, then construct $\mathring{m}(x) = \sum_{n=0}^{2^{d-1}-1} a_n x^n \in R_t$ where $a_n = 0 \; \forall \, n \geq b$.
- **Encryption** $\texttt{Enc}(k_p, m)$
  First encode $m \in \mathbb{Z}$ as $\underline{\mathring{m}} \in R_t$

$$c := ([\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}}]_q, [\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2]_q)$$

  where $\underline{u}, \underline{e}_1, \underline{e}_2 \sim \chi$ and $\Delta = \lfloor \frac{q}{t} \rfloor$.
- **Decryption** $\texttt{Dec}(k_s, c)$

$$\underline{\mathring{m}} = \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t$$

  so that $m = \mathring{m}(2)$ … note, bootstrappable.

Fan & Vercauteren (2012) : understanding

$$
\begin{aligned}
&\texttt{Dec}(k_s, c) \\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t
\end{aligned}
$$

Fan & Vercauteren (2012) : understanding

$$\begin{aligned}
&\mathtt{Dec}(k_s, c) \\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t[\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t
\end{aligned}$$

i-like.org.uk

Fan & Vercauteren (2012) : understanding

$$
\begin{aligned}
&\mathtt{Dec}(k_s, c)\\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t\\
&= \left[\left\lfloor \frac{t[\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t\\
&= \left[\left\lfloor \frac{t[-(\underline{a} \cdot \underline{k}_s + \underline{e}) \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{a} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t
\end{aligned}
$$

i-like.org.uk

Fan & Vercauteren (2012) : understanding

$$\begin{aligned}
&\texttt{Dec}(k_s, c) \\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[-(\underline{a} \cdot \underline{k}_s + \underline{e}) \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{a} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[-\underline{e} \cdot \underline{u} + \underline{e}_1 + \left\lfloor \frac{q}{t} \right\rceil \underline{\mathring{m}} + \underline{e}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t
\end{aligned}$$

Fan & Vercauteren (2012) : understanding

$$\begin{aligned}
&\mathtt{Dec}(k_s, c) \\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t[\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \mathring{\underline{m}} + (\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t[-(\underline{a} \cdot \underline{k}_s + \underline{e}) \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \mathring{\underline{m}} + (\underline{a} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t[-\underline{e} \cdot \underline{u} + \underline{e}_1 + \left\lfloor \frac{q}{t} \right\rceil \mathring{\underline{m}} + \underline{e}_2 \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t
\end{aligned}$$

i-like.org.uk

Fan & Vercauteren (2012) : understanding

$$
\begin{aligned}
&\texttt{Dec}(k_s, c) \\
&= \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[-(\underline{a} \cdot \underline{k}_s + \underline{e}) \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}} + (\underline{a} \cdot \underline{u} + \underline{e}_2) \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t[-\underline{e} \cdot \underline{u} + \underline{e}_1 + \left\lfloor \frac{q}{t} \right\rceil \underline{\mathring{m}} + \underline{e}_2 \cdot \underline{k}_s]_q}{q} \right\rceil \right]_t
\end{aligned}
$$

But, note that $\|-\underline{e} \cdot \underline{u} + \underline{e}_1 + \underline{e}_2 \cdot \underline{k}_s\|_\infty \ll \frac{q}{t}$ by construction, so that after multiplication by $\frac{t}{q}$ the only term surviving rounding is $\underline{\mathring{m}}$.

i-like.org.uk

Fan & Vercauteren (2012) : addition/multiplication

- **Addition**, $+$ Standard vector and polynomial addition with modulo reduction:

$$c_1 + c_2 = ([\underline{c}_{11} + \underline{c}_{21}]_q, [\underline{c}_{12} + \underline{c}_{22}]_q)$$

- **Multiplication** $\times$ Multiplication increases length of the cipher text vector:

$$c_1 \times c_2 = \left(\left[\left\lfloor \frac{t(\underline{c}_{11} \cdot \underline{c}_{21})}{q} \right\rceil \right]_q, \left[\left\lfloor \frac{t(\underline{c}_{11} \cdot \underline{c}_{22} + \underline{c}_{12} \cdot \underline{c}_{21})}{q} \right\rceil \right]_q, \left[\left\lfloor \frac{t(\underline{c}_{12} \cdot \underline{c}_{22})}{q} \right\rceil \right]_q \right)$$

Still possible to recover $\underline{\mathring{m}}$ by modifying decryption to be $\left[\left\lfloor \frac{t}{q} [\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s + \underline{c}_3 \cdot \underline{k}_s \cdot \underline{k}_s]_q \right\rceil \right]_t$, it is preferable to perform a 'relinearisation' procedure which compacts the cipher text to a vector of two polynomials again.

i-like.org.uk

Fan & Vercauteren (2012) : parameter choice

A reasonable default of:

$d = 4096$

$q = 2^{128} = 340282366920938463463374607431768211456$

$t = 32768$

$\sigma = 16$

gives approximately 128-bit security level and about 4 multiplications deep.

There are theoretical bounds on both multiplicative depth and security level in the literature (Lindner & Peikert (2011), Fan & Vercauteren (2012), Lepoint & Naehrig (2014))

Fan & Vercauteren (2012) : limitations overview

1. Message space
   - R_t, so must encode single datum as polynomials
2. Cipher text size
   - Single 4/8-byte value $\in \mathbb{Z}$ transformed to $R_q \times R_q \implies$ 128KB for parameters on previous slide
3. Computational cost
   - 1 message $+ \implies$ 8192 lots of 128-bit modular addition
   - 1 message $\times \implies$ 4 lots of 4096 degree polynomial multiplcations involving 128-bit values, plus 8192 lots of 128-bit addition, plus integer addition and multiplication followed by polynomial modular reduction.
4. Division and comparison operations
   - Impossible!
5. Depth of operations
   - multiplications limited because end up with products of $-\underline{e} \cdot \underline{u}$, $\underline{e}_1$ and $\underline{e}_2$ terms so that ultimately noise exceeds $\frac{q}{t}$

i-like.org.uk

Ameliorating computational burden

Theorem (Chinese Remainder Theorem)

Let $m_1, \dots, m_k \in \mathbb{Z}^+$ be pairwise coprime positive integers. Let $M = \prod_{i=1}^k m_i$ and let $a, x_1, \dots, x_k \in \mathbb{Z}$. Then there is exactly one integer x that satisfies the conditions:

$$a \le x < a + m \qquad \text{and} \qquad x \equiv x_i \mod m_i \; \forall \, 1 \le i \le k$$

Thus, an integer message $x \in [a, a+m)$ can be uniquely represented by the collection of smaller integers $\{x_i\}_{i=1}^k$ … this is a Residue Number System. Conversely, can also think of $\{x_i\}_{i=1}^k$ being represented by x.

Going $x \to \{x_i\}_{i=1}^k$ is simply taking modulo each m_i.

Going $\{x_i\}_{i=1}^k \to x$ can be constructed via the extended Euclidean algorithm.

Arithmetic with CRT

In particular, note that a Chinese Remainder Theorem representation preserves modular arithmetic.

Let $\{x_i\}_{i=1}^k, \{y_i\}_{i=1}^k$ be two collections of residue numbers, modulo $\{m_i\}_{i=1}^k$. Let x and y be the corresponding integers satisfying the Chinese Remainder Theorem. Then,

$$z = x + y \iff z \mod m_i = z_i = (x_i + y_i) \mod m_i$$

In other words, doing one addition $(x + y)$ actually gives k additions by looking at the single result modulo each m_i.

i-like.org.uk

Polynomial Chinese Remainder Theorem (I)

There is a corresponding CRT for polynomials.

Although $\Phi_n(x)$ is irreducible over $\mathbb{Q}[x]$, it is not necessarily irreducible over $\mathbb{Z}_t[x]$. Suppose it has r factors:

$$\Phi_n(x) = \prod_{j=1}^{r} f_j(x)$$

Then, we can encode a vector of polynomial messages $(\underline{\mathring{m}}_1, \ldots, \underline{\mathring{m}}_r)$ since by the Polynomial Chinese Remainder Theorem $\exists\ \underline{\mathring{m}} \in \mathbb{Z}_t[x]/\Phi_n(x)$ such that $\underline{m} \mod f_i(x) = \underline{\mathring{m}}_i$.

Upshot: if we now encrypt $\underline{m}$, then we have encrypted a CRT representation of r messages in just one cipher text.

Polynomial Chinese Remainder Theorem (II)

So, consider a collection of vectors of polynomials encoded in this way

$$\mathbb{Z}_t[x]/f_1(x) \times \cdots \times \mathbb{Z}_t[x]/f_r(x) \ni (\underline{\mathring{m}}_{i1}, \dots, \underline{\mathring{m}}_{ir}) \longrightarrow \underline{\mathring{m}}_i \in R_t$$

Then,

$$\left(\sum_i \underline{\mathring{m}}_i\right) \mod f_j(x) = \left(\sum_i \underline{\mathring{m}}_{ij}\right) \mod f_j(x) \quad \forall j = 1, \dots, r$$
$$\left(\prod_i \underline{\mathring{m}}_i\right) \mod f_j(x) = \left(\prod_i \underline{\mathring{m}}_{ij}\right) \mod f_j(x) \quad \forall j = 1, \dots, r$$

In other words, we can do SIMD on cipher texts. There also exist automorphism mappings which will allow slots to be exchanged and interacted. (Smart & Vercauteren 2014)

Software

Existing implementations

- `libfhe` (Minar 2010) compact single C file library implementing Gentry (2010)
- 'Scarab' (Perl et al. 2011) low level C library implementing Smart & Vercauteren (2010)
- 'HELib' (Halevi & Shoup 2014) most impressive library, in C++ implementing Brakerski et al. (2012) and lots beyond the bare bones cryptography (i.e. Polynomial Chinese Remainder Theorem + automorphisms)
- more besides ...

However, these all tend to be very low-level libraries.

HomomorphicEncryption R package (Aslett 2014)

All core code in high-performance multi-threaded C++, but accessible via simple R functions and overloaded operators:

```
library("HomomorphicEncryption")

p <- pars("FandV")
k <- keygen(p)
c1 <- enc(k$pk, c(42,34))
c2 <- enc(k$pk, c(7,5))
cres1 <- c1 + c2
cres2 <- c1 * c2
cres3 <- c1 %*% c2
dec(k$sk, cres1)
dec(k$sk, cres2)
dec(k$sk, cres3)
```

i-like.org.uk

Demo

ML

Random Forests

Want to build a random forest on some encrypted data. But,

- No comparisons possible to evaluate splits
- No max possible to find highest class vote
- No division possible to do average votes
- ...

So random forests (and other methods) need to be tailored for encrypted computation. This is where statistics and machine learning community can get involved!

Data representation

First, need to arrange data to at least enable decision trees to be evaluated.

1. Assume $x_{ij} \in \mathbb{R}$ (or categorical) and make partition of support of variable j, $\mathcal{K}_j = \{K_1^j, \ldots, K_m^j\}$.

$$x_{ij} \in \bigcup_{k=1}^{m} K_k^j \; \forall\, i, j \quad \text{and} \quad K_i^j \cap K_k^j = \varnothing \; \forall\, j, \forall\, i \neq k$$

2. Encode x_{ij} as indicator $\tilde{x}_{ijk} \in \{0,1\} \; \forall\, k$, where $\tilde{x}_{ijk} = 1 \iff x_{ij} \in K_k^j$ and $\tilde{x}_{ijl} = 0 \; \forall\, l \neq k$.

$$X = \begin{pmatrix} 0 & 1 & 1.7 \\ 1 & 2 & 1.9 \\ 0 & 3 & 1.6 \\ \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & x_{i3} \end{pmatrix} \rightarrow \tilde{X} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \vdots & & \vdots & & & & \vdots & & \\ \tilde{x}_{i11} & \tilde{x}_{i21} & \tilde{x}_{i22} & \tilde{x}_{i23} & \tilde{x}_{i31} & \tilde{x}_{i32} & \tilde{x}_{i33} & \tilde{x}_{i34} & \tilde{x}_{i35} \end{pmatrix}$$

Pseudo-comparisons

This data representation means:

$$\sum_{\forall\, k} \tilde{x}_{i_1jk}\tilde{x}_{i_2jk} = 1 \iff \text{obs } i_1 \text{ and } i_2 \text{ equal quantised value on var } j$$

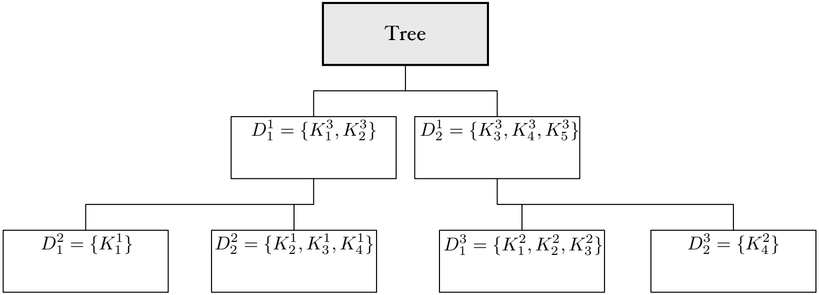
and

$$\sum_{k \in K} \tilde{x}_{ijk} = 1 \iff \text{obs } i \text{ has quantised value in set } K$$

which provide sufficient ‘psuedo-comparisons’ to construct a *completely random forest*.

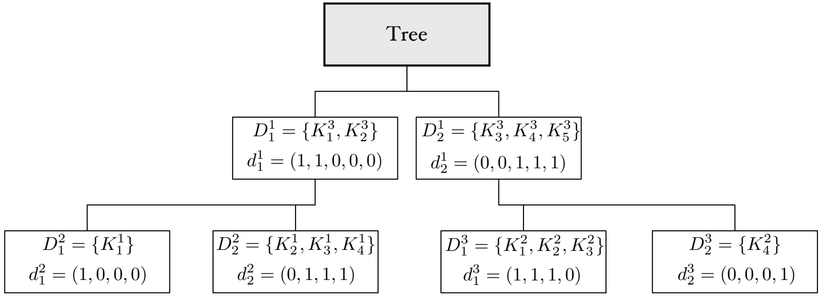
Completely Random Forests (I)

1. Select variable, j, to perform split on completely at random.
2. Select the split point completely at random. That is, choose level l decision splits D_1^l and D_2^l such that $D_1^l \cup D_2^l = \mathcal{K}_j$ and $D_1^l \cap D_2^l = \varnothing$.
3. Repeat to a prespecified tree depth.


Tree
$D_1^1 = \{K_1^3, K_2^3\}$
$D_2^1 = \{K_3^3, K_4^3, K_5^3\}$
$D_1^2 = \{K_1^1\}$
$D_2^2 = \{K_2^1, K_3^1, K_4^1\}$
$D_1^3 = \{K_1^2, K_2^2, K_3^2\}$
$D_2^3 = \{K_4^2\}$

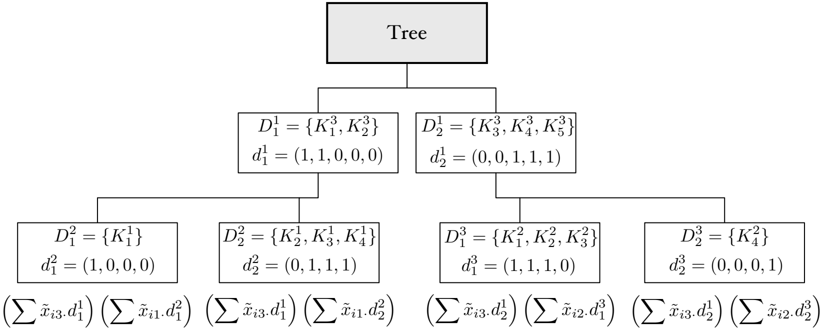
Completely Random Forests (II)

Encrypt under k_p indicators of the splits:



Completely Random Forests (II)

For each observation, evaluate every branch of the decision tree:

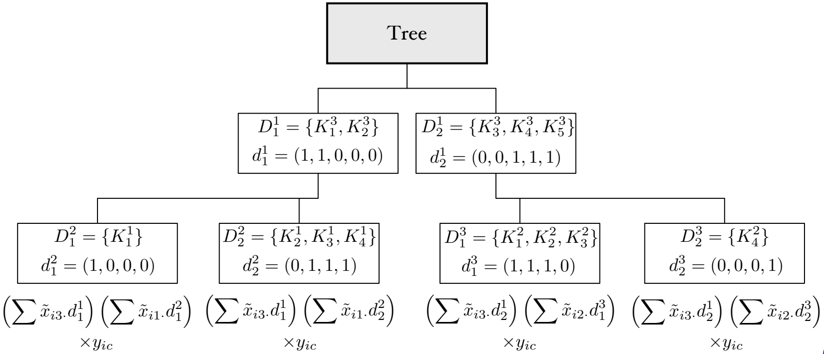

Tree
$D_1^1 = \{K_1^3, K_2^3\}$
$d_1^1 = (1,1,0,0,0)$
$D_2^1 = \{K_3^3, K_4^3, K_5^3\}$
$d_2^1 = (0,0,1,1,1)$
$D_1^2 = \{K_1^1\}$
$d_1^2 = (1,0,0,0)$
$D_2^2 = \{K_2^1, K_3^1, K_4^1\}$
$d_2^2 = (0,1,1,1)$
$D_1^3 = \{K_1^2, K_2^2, K_3^2\}$
$d_1^3 = (1,1,1,0)$
$D_2^3 = \{K_4^2\}$
$d_2^3 = (0,0,0,1)$
$\left(\sum \tilde{x}_{i3}.d_1^1\right)\left(\sum \tilde{x}_{i1}.d_1^2\right)$
$\left(\sum \tilde{x}_{i3}.d_1^1\right)\left(\sum \tilde{x}_{i1}.d_2^2\right)$
$\left(\sum \tilde{x}_{i3}.d_2^1\right)\left(\sum \tilde{x}_{i2}.d_1^3\right)$
$\left(\sum \tilde{x}_{i3}.d_2^1\right)\left(\sum \tilde{x}_{i2}.d_2^3\right)$

i-like.org.uk

Completely Random Forests (III)

Response class also expanded in a binary fashion, $y_i \to \tilde{y}_{ic} \in \{0,1\}$ for $c \in \{1,\dots,|\mathcal{C}|\}$.

Then observation is a 'vote' for class c from each terminal leaf if it survives:



Tree
$D_1^1 = \{K_1^3, K_2^3\}$
$d_1^1 = (1,1,0,0,0)$
$D_2^1 = \{K_3^3, K_4^3, K_5^3\}$
$d_2^1 = (0,0,1,1,1)$
$D_1^2 = \{K_1^1\}$
$d_1^2 = (1,0,0,0)$
$D_2^2 = \{K_2^1, K_3^1, K_4^1\}$
$d_2^2 = (0,1,1,1)$
$D_1^3 = \{K_1^2, K_2^2, K_3^2\}$
$d_1^3 = (1,1,1,0)$
$D_2^3 = \{K_4^2\}$
$d_2^3 = (0,0,0,1)$
$\left(\sum \tilde{x}_{i3}.d_1^1\right)\left(\sum \tilde{x}_{i1}.d_1^2\right) \times y_{ic}$
$\left(\sum \tilde{x}_{i3}.d_1^1\right)\left(\sum \tilde{x}_{i1}.d_2^2\right) \times y_{ic}$
$\left(\sum \tilde{x}_{i3}.d_2^1\right)\left(\sum \tilde{x}_{i2}.d_1^3\right) \times y_{ic}$
$\left(\sum \tilde{x}_{i3}.d_2^1\right)\left(\sum \tilde{x}_{i2}.d_2^3\right) \times y_{ic}$

i-like.org.uk

Completely Random Forests (IV)

Summing each terminal leaf over all observations renders total number of 'votes' for each class from the whole data set.

Thus:

- a tree is represented by:
  - the split partitions D;
  - the total votes in each terminal leaf for each class.
- prediction involves:
  - evaluating a new observation through all branches;
  - taking product with corresponding vote totals for each class;
  - summing across trees and across leaves to get total votes for each class.

Biggest problem

Every tree contributes according to raw votes, not how well separation occurs.

Thus, confused leaves with many votes can overwhealm certain ones with few.

To overcome this Random Forests usually use:

1. single vote per tree (requires comparison to find max)
2. relative class frequencies (requires division)

… develop novel method to achieve an unbiased approximation to 2.

Relative class frequencies

Hereinafter, consider just one leaf (drop spurious notation).

Let ν_c be the number of votes for class c in the leaf. The relative class frequency contribution should be:

$$\frac{\nu_c}{\sum_c \nu_c}$$

But, this belongs to $[0, 1]$ which we can't represent and involves division. Target equivalently:

$$\nu_c \left\lfloor \frac{N}{\sum_c \nu_c} \right\rceil$$

where N is the number of training observations.

- By construction $\sum_c \nu_c \leq N$, so $0 \leq \frac{\sum_c \nu_c}{N} \leq 1$
- Recall, $X \sim \text{Geometric}(p) \implies \mathbb{E}[X] = p^{-1}$

i-like.org.uk

Stochastic fraction estimate (I)

Thus, unbiased approximation to fraction is draw from Geometric distribution with probability $\frac{\sum_c \nu_c}{N}$.

Better than division?

Stochastic fraction estimate (I)

Thus, unbiased approximation to fraction is draw from Geometric distribution with probability $\frac{\sum_c \nu_c}{N}$.

Better than division?

Crucial observation: each ν_c arises from summing a binary vector $\{0, 1\}^N$.

Define $\nu_c := \sum_{i=1}^N \eta_{ci}$ (so η_{ci} is 1 if training obs. i was of class c and fell in this leaf of the decision tree).

$\implies$ blind random sampling from $\{\sum_c \eta_{ci} : i = 1, \ldots, N\}$ will produce 1 with probability exactly $\frac{\sum_c \nu_c}{N}$.

Stochastic fraction estimate (II)

Problem: count number of leading zeros in an encrypted Bernoulli process.

Stochastic fraction estimate (II)

Problem: count number of leading zeros in an encrypted Bernoulli process.

Inspiration from CPU hardware algorithm:

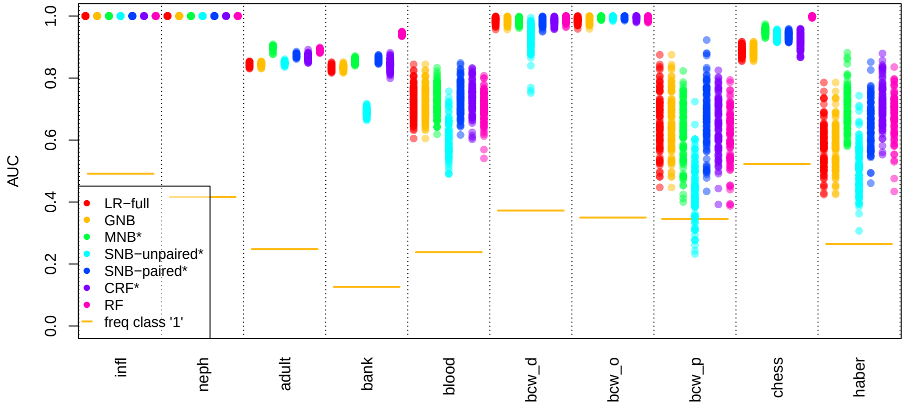
Let $\xi_1, \ldots, \xi_M$ be a resampled vector ($\xi_i = \sum_c \eta_{cj}$, some j) and assume M is a power of 2.

1. For $l \in \{0, \ldots, \log_2(M) - 1\}$:
   - Set $\xi_i = \xi_i \vee \xi_{i-2^l} = \xi_i + \xi_{i-2^l} - \xi_i \xi_{i-2^l} \quad \forall\, 2^l + 1 \leq i \leq M$
2. The number of leading zeros is $M - \sum_{i=1}^M \xi_i$

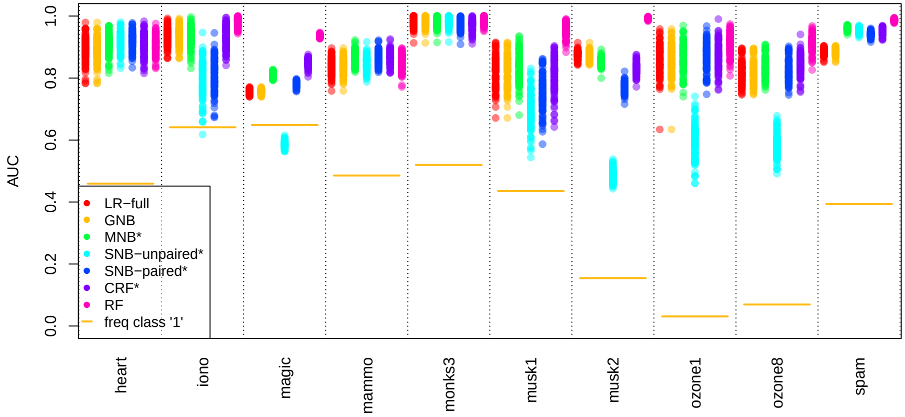
Corresponds to increasing power of 2 bit-shifts OR'd with itself, all computable encrypted.

$$\implies \left\lfloor \frac{N}{\sum_c \nu_c} \right\rfloor \approx M - \sum_{i=1}^{M} \xi_i + 1$$

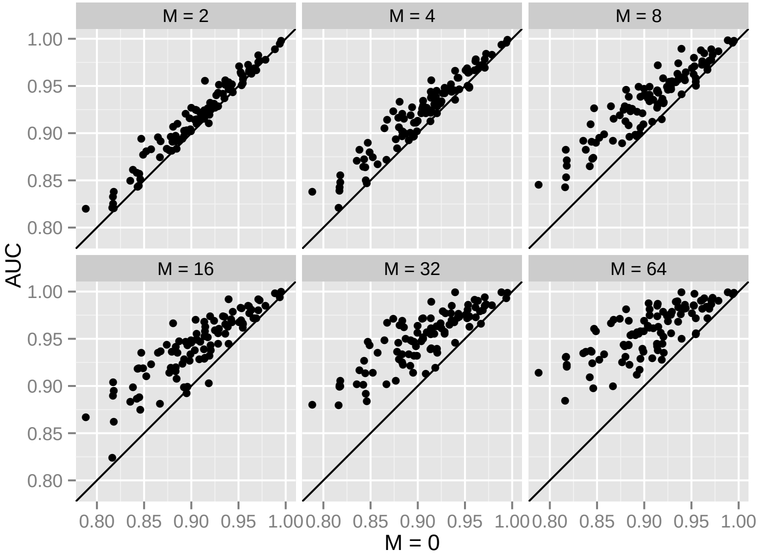
Results (I)


AUC
0.0
0.2
0.4
0.6
0.8
1.0
● LR−full
● GNB
● MNB*
● SNB−unpaired*
● SNB−paired*
● CRF*
● RF
— freq class '1'
infl
neph
adult
bank
blood
bcw_d
bcw_o
bcw_p
chess
haber

Results (II)

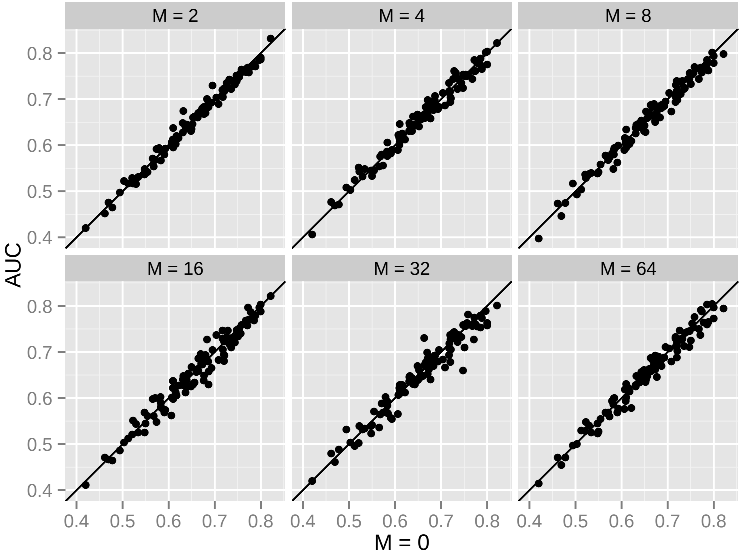

AUC
1.0
0.8
0.6
0.4
0.2
0.0
LR−full
GNB
MNB*
SNB−unpaired*
SNB−paired*
CRF*
RF
freq class '1'
heart
iono
magic
mammo
monks3
musk1
musk2
ozone1
ozone8
spam

Stochastic fraction effect (best)



i-like.org.uk

Stochastic fraction effect (worst)



References I

Aslett, L. J. M. (2014). HomomorphicEncryption: Fully homomorphic encryption. `http://www.louisaslett.com/HomomorphicEncryption/`.

Aslett, L. J. M., Esperança, P. M., & Holmes, C. C. (2015a). *A review of homomorphic encryption and software tools for encrypted statistical machine learning*. University of Oxford.

Aslett, L. J. M., Esperança, P. M., & Holmes, C. C. (2015b). Encrypted statistical machine learning: New privacy preserving methods. *arXiv:1508.06845 [stat.ML]*.

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2012). (Leveled) fully homomorphic encryption without bootstrapping. *Proceedings of the 3rd innovations in theoretical computer science conference*, pp. 309–25. ACM.

Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*.

Gentry, C. (2009). *A fully homomorphic encryption scheme* (PhD thesis). Stanford University. Retrieved from `<crypto.stanford.edu/craig>`

References II

Gentry, C. (2010). Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53/3: 97–105. ACM.

Halevi, S., & Shoup, V. (2014). HElib. `https://github.com/shaih/HElib`.

Lepoint, T., & Naehrig, M. (2014). A comparison of the homomorphic encryption schemes FV and YASHE. *Progress in cryptology–AFRICACRYPT 2014*, pp. 318–35. Springer.

Lindner, R., & Peikert, C. (2011). Better key sizes (and attacks) for LWE-based encryption. *Topics in cryptology–CT-rSA 2011*, pp. 319–39. Springer.

Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. *Proceedings of the 29th annual international conference on theory and applications of cryptographic techniques*. Springer-Verlag.

Minar, J. (2010). Libfhe. `https://github.com/rdancer/fhe/tree/master/libfhe`.

Perl, H., Brenner, M., & Smith, M. (2011). Scarab library. `https://hcrypt.com/scarab-library/`.

References III

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4/11: 169–80.

Smart, N. P., & Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes. *Public key cryptography–PKC 2010*, pp. 420–43. Springer.

Smart, N. P., & Vercauteren, F. (2014). Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71/1: 57–81.