Cryptographically secure multiparty evaluation of system reliability

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

Department of Statistics, University of Oxford
and Corpus Christi College, Oxford

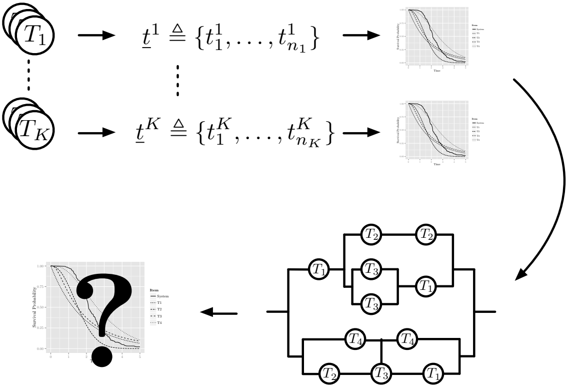
London Mathematical Society MMR Series
16 March 2015: Durham University

i-like.org.uk

Introduction

Introduction (I)

Objective: inference on system/network reliability given component test data.



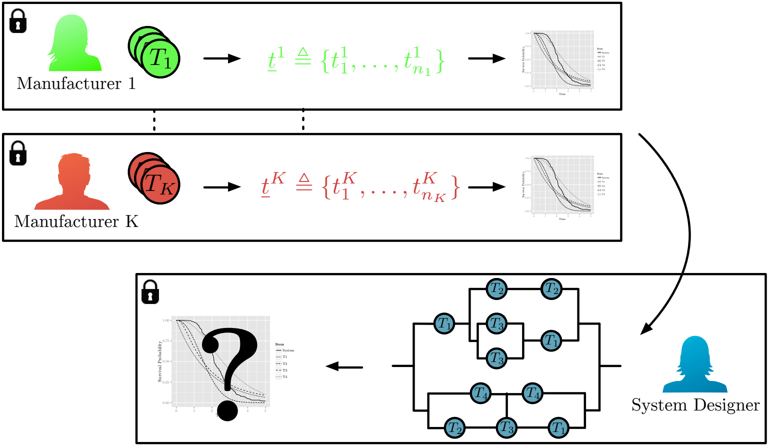
Aslett, L. J. M., Coolen, F. P. A., & Wilson, S. P. (2014). Bayesian inference for reliability of systems and networks using the survival signature. *Risk Analysis*.

i-like.org.uk

Introduction (II)

But, what are the privacy requirements of data owners?

New objective: inference on system/network reliability *maintaining privacy*.



i-like.org.uk

Introduction (III)

Developments in cryptography in 2009 solved an open problem which existed since 1978.

We'll see these developments enable preservation of privacy, almost completely, because the survival signature allows system lifetime to be expressed as a low order homogeneous polynomial.

$$\mathbb{P}(T_S > t) \stackrel{iid}{=} \sum_{l_1=0}^{m_1} \cdots \sum_{l_K=0}^{m_K} \left[\Phi(l_1, \ldots, l_K) \prod_{k=1}^{K} \binom{m_k}{l_k} [F_k(t)]^{m_k - l_k} [\bar{F}_k(t)]^{l_k} \right]$$

An accessible background on emerging area of encryption and statistics appearing soon:

Aslett, L. J. M., Esperança, P., & Holmes, C. C. (2015). Secure statistical analysis. *Technical report, University of Oxford*.

Bayesian Inference

Component inference (parametric)

Given test data directly on the components, inference is a well studied problem. For example, parametrically we can model the lifetime of a component of type k via likelihood function f_k

$$T \sim f_k(\cdot \,; \psi_k)$$

Then, given iid test data $\underline{t}^k = \{t_1^k, \dots, t_{n_k}^k\}$ for components of type k, posterior density is:

$$f_{\Psi_k \,|\, \underline{T}^k}(\psi_k \,|\, \underline{t}^k) \propto f_{\Psi_k}(\psi_k) \prod_{i=1}^{n_k} f_k(t_i^k; \psi_k)$$

Straight forward to use MCMC to generate posterior samples of ψ_k which encapsulates uncertainty in the parametric family.

Component inference (non-parametric I)

Or, non-parametrically we can observe that at fixed time t, probability a component of type k functions is Bernoulli(p_t^k) for some unknown p_t^k.

$\implies$ number functioning at time t in iid batch of n_k is Binomial(n_k, p_t^k).

Let $S_t^k \in \{0, 1, \dots, n_k\}$ be number of working components in test batch of n_k components of type k. Then,

$$S_t^k \sim \text{Binomial}(n_k, p_t^k) \;\; \forall \, t > 0$$

Given the same test data $\underline{t}^k = \{t_1^k, \dots, t_{n_k}^k\}$, for each t we can form corresponding observation from Binomial model

$$s_t^k \triangleq \sum_{i=1}^{n_k} \mathbb{I}(t_i^k > t)$$

Component inference (non-parametric II)

Taking prior $p_t^k \sim \text{Beta}(\alpha_t^k, \beta_t^k)$, exploit conjugacy result

$$p_t^k \mid s_t^k \sim \text{Beta}(\alpha_t^k + s_t^k, \beta_t^k + n_k - s_t^k)$$

Then, posterior predictive for number of components surviving in a new batch of m_k components is

$$C_t^k \mid s_t^k \sim \text{Beta-binomial}(m_k, \alpha_t^k + s_t^k, \beta_t^k + n_k - s_t^k)$$

Component inference (non-parametric II)

Taking prior $p_t^k \sim \text{Beta}(\alpha_t^k, \beta_t^k)$, exploit conjugacy result

$$p_t^k \mid s_t^k \sim \text{Beta}(\alpha_t^k + s_t^k, \beta_t^k + n_k - s_t^k)$$

Then, posterior predictive for number of components surviving in a new batch of m_k components is

$$C_t^k \mid s_t^k \sim \text{Beta-binomial}(m_k, \alpha_t^k + s_t^k, \beta_t^k + n_k - s_t^k)$$

Summary: for any fixed t, s_t^k provides a minimal sufficient statistic for computing posterior predictive distribution of the number of components surviving to t in a new batch, without any parametric model for component lifetime being assumed.

Propagate uncertainty: naïve approach

In principle, the structure function can be used to propagate component lifetime uncertainty to the system.

$$\phi(\underline{x}) = \prod_{j=1}^{s} \left(1 - \prod_{i \in C_j} (1 - x_i) \right)$$

where $\{C_1, \dots, C_s\}$ is the collection of minimal cut sets of the system. Then,

$$P(T_{S^*} > t \mid s_t^1, \dots s_t^K)$$
$$= \int \cdots \int \phi(p_t^{x_1}, \dots, p_t^{x_n}) P(p_t^1 \mid s_t^1) \dots P(p_t^K \mid s_t^K) \, dp_t^1 \dots dp_t^K$$

where $p_t^{x_i}$ is the element of $\{p_t^1, \dots, p_t^K\}$ corresponding to component i.

Have fun with that integral for large $K \dots$!

Survival signature

Coolen & Coolen-Maturi (2012) rethought signatures with the objective of retaining separation of structure and component lifetimes for multiple component types.

Definition (Survival signature)

Consider a system comprising K component types, with m_k components of type $k \in \{1, \ldots, K\}$. Then the *survival signature* $\Phi(l_1, \ldots, l_K)$, with $l_k \in \{0, 1, \ldots, m_k\}$, is the probability that the system functions given precisely l_k of its components of type k function.

$$\Phi(l_1, \ldots, l_K) = \left[\prod_{k=1}^{K} \binom{m_k}{l_k}^{-1} \right] \sum_{\underline{x} \in S_{l_1, \ldots, l_K}} \varphi(\underline{x})$$

where $S_{l_1,\ldots,l_K} = \{\underline{x} : \sum_{i=1}^{m_k} x_i^k = l_k \quad \forall\, k\}$

Propagate uncertainty: survival signature

$$
\begin{aligned}
&P(T_{S^*} > t \mid \underline{t}^1, \dots \underline{t}^K) \\
&= \int \cdots \int \left[\sum_{l_1=0}^{m_1} \cdots \sum_{l_K=0}^{m_K} \Phi(l_1, \dots, l_K) \right. \\
&\qquad \left. \times \prod_{k=1}^{K} \binom{m_k}{l_k} [F_k(t; \psi_k)]^{m_k - l_k} [1 - F_k(t; \psi_k)]^{l_k} \right] \\
&\qquad \times f_{\Psi_1 \mid \underline{T}^1}(d\psi_1 \mid \underline{t}^1) \dots f_{\Psi_K \mid \underline{T}^K}(d\psi_K \mid \underline{t}^K) \\
&= \sum_{l_1=0}^{m_1} \cdots \sum_{l_K=0}^{m_K} \Phi(l_1, \dots, l_K) \\
&\qquad \times \prod_{k=1}^{K} \binom{m_k}{l_k} \int [F_k(t; \psi_k)]^{m_k - l_k} [1 - F_k(t; \psi_k)]^{l_k} f_{\Psi_k \mid \underline{T}^k}(d\psi_k \mid \underline{t}^k)
\end{aligned}
$$

i-like.org.uk

Final term post pred of l_k comp of type k surviving to t.

Homomorphic Encryption

Introduction to cryptography

- Unencrypted number, $m \in M$, is referred to as a *message*.
- Encrypted version, $c \in C$, is referred to as a *cipher text*.
- Pair of 'keys' (k_s, k_p), secret and public.
- Injective map (*not* function), $\texttt{Enc} : M \to C$.
- Surjective function, $\texttt{Dec} : C \to M$.

Fundamental point

Easy

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

Hard without k_s

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

$\therefore$ crucial relation:

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

'Brittle' encryption

Most cryptography schemes are 'brittle' in that we can't manipulate the contents of 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

Rivest et al. (1978) hypothesised that a limited set of functions may be possible to compute encrypted: specifically those involving addition and multiplication (theoretically exciting → computational complexity & polynomial approx).

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:

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

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

* 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
- $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$
- 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
- **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]

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** $\mathtt{Enc}(k_p, m)$
  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 $\Delta = \left\lfloor \frac{q}{t} \right\rfloor$.
- **Decryption** $\mathtt{Dec}(k_s, c)$

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

  so that $m = \mathring{m}(2)$

* 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.

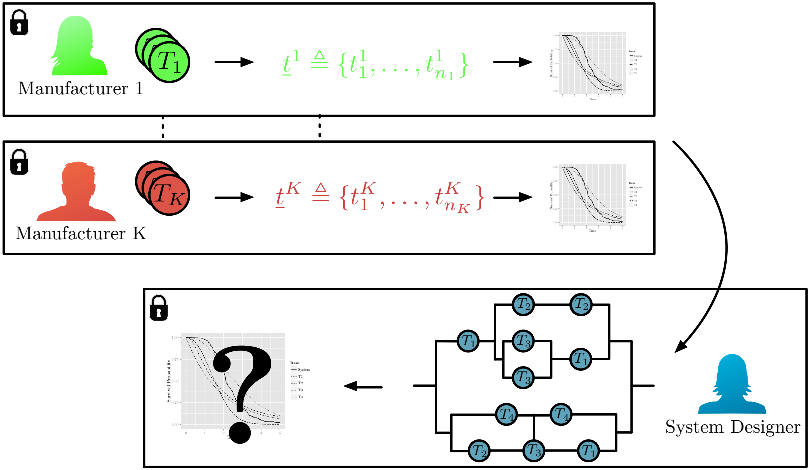
i-like.org.uk

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!

Privacy Preserving Protocol

Back to the problem at hand ...



Step 1: Encrypt system design (I)

System designer:

1. Generate public/private keys $(k_s, k_p) = \mathsf{Keygen}()$
2. Compute the full survival signature $\Phi(l_1, \dots, l_K) \; \forall \, l_k \in \{0, \dots, m_k\}$
3. Arrange the survival signature table into a matrix, encrypting the survival signature probability:

$$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \mathsf{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 0) \rceil\right) \\ 0 & \cdots & 1 & \mathsf{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \mathsf{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1, \dots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \mathsf{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1, \dots, m_K) \rceil\right) \end{pmatrix}$$

   where ν is the required number of decimal places of precision in the final uncertainty quantification.
4. Decide times to compute reliability, $\underline{t} = \{t_1, \dots, t_T\}$

i-like.org.uk

Step 2: Communication to manufacturer 1

System designer sends to manufacturer 1:

- k_p, designer's public key
- ν, decimal places of accuracy
- $\underline{\lambda}_1 \triangleq \Xi_{\cdot,1}$, type 1 component quantities
- $\eta = \{\underline{\eta}^1, \dots, \underline{\eta}^T\}$ where $\underline{\eta}^j \triangleq \Xi_{\cdot,K+1} \; \forall j$, that is T copies of encrypted survival signature probabilities
- $\underline{t}$, times to evaluate reliability

Thus,

- the manufacturer will see how many of their own components are being used in the system ($\underline{\lambda}_1$ is unencrypted);
- due to repetition, the manufacturer will have weak impression of overall system size (*open Q*);
- no knowledge of the survival signature probability, so can't solve for exact component numbers or layout.

i-like.org.uk

Step 3: Manufacturer i computation, $i \in \{1, \ldots, K\}$

$\therefore$ manufacturer i will posess k_p, ν, $\underline{\lambda}_i$, η and $\underline{t}$.

Manufacturer i then:

- Constructs vectors $\eta^{j*}, j \in \{1, \ldots, T\}$, where element i is:

$$\eta_i^{j*} = \binom{m_1}{\lambda_{1i}} \frac{\mathrm{B}(\lambda_{1i} + \alpha_{t_j}^1 + s_{t_j}^1, m_1 - \lambda_{1i} + \beta_{t_j}^1 + n_1 - s_{t_j}^1)}{\mathrm{B}(\alpha_{t_j}^1 + s_{t_j}^1, \beta_{t_j}^1 + n_1 - s_{t_j}^1)}$$

- Updates all elements of η received by:

$$\eta_i^j = \eta_i^j \mathsf{Enc}(k_p, \lfloor 10^\nu \eta_i^{j*} \rceil)$$

Step 4: Further communication

Manufacturer i sends to manufacturer $i+1$, $i \in \{1, \dots, K-1\}$

- η, updated collection of encrypted signature vectors at times selected for reliability evaluation

System designer sends to manufacturer $i+1$:

- k_p, designer's public key
- ν, decimal places of accuracy
- $\underline{\lambda}_{i+1} \triangleq \Xi_{\cdot,i+1}$, type $i+1$ component quantities
- $\underline{t}$, times to evaluate reliability

Thus, designer doesn't see incrementally updated η, so cannot infer component reliabilities stepwise.

Step 5: Final computation & communication

Once manufacturer K has completed regular computation step, there is one additional computation step:

- For each j, compute homomorphically:

$$\tau^j = \sum_i \eta_i^j$$

Then, $\underline{\tau}$ is finally returned to the system designer. Upon decryption, due to homogeneity of polynomial:

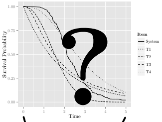
$$\begin{aligned}
\tau^j &= 10^{(K+1)\nu} \sum_{l_1=0}^{m_1} \cdots \sum_{l_K=0}^{m_K} \Phi(l_1, \dots, l_K) \\
&\quad \times \prod_{k=1}^{K} \binom{m_k}{l_k} \frac{\mathrm{B}(l_k + \alpha_{t_j}^k + s_{t_j}^k, m_k - l_k + \beta_{t_j}^k + n_k - s_{t_j}^k)}{\mathrm{B}(\alpha_{t_j}^k + s_{t_j}^k, \beta_{t_j}^k + n_k - s_{t_j}^k)} \\
&= 10^{(K+1)\nu} P(T_{S^*} > t_j \mid \underline{t}^1, \dots \underline{t}^K)
\end{aligned}$$

but designer never saw $\underline{t}^1, \dots \underline{t}^K$, manufacturers never saw S.

k_s k_p

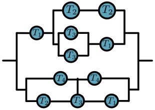
System Designer

ν



$\underline{t} = \{t_1, \dots, t_T\}$

$\Phi(l_1, \dots, l_K)$

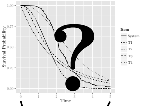


i-like.org.uk

k_s k_p

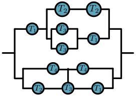
System Designer

ν



$\underline{t} = \{t_1, \ldots, t_T\}$

$\Phi(l_1, \ldots, l_K)$

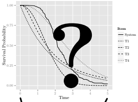


$$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \texttt{Enc}\left(k_p, \lfloor 10^{\nu} \Phi(0, \ldots, 0) \rceil\right) \\ 0 & \cdots & 1 & \texttt{Enc}\left(k_p, \lfloor 10^{\nu} \Phi(0, \ldots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \texttt{Enc}\left(k_p, \lfloor 10^{\nu} \Phi(l_1, \ldots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \texttt{Enc}\left(k_p, \lfloor 10^{\nu} \Phi(m_1, \ldots, m_K) \rceil\right) \end{pmatrix}$$

k_s 🔒 k_p

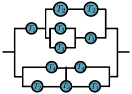
System Designer

ν



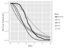
$\underline{t} = \{t_1, \dots, t_T\}$

$\Phi(l_1, \dots, l_K)$

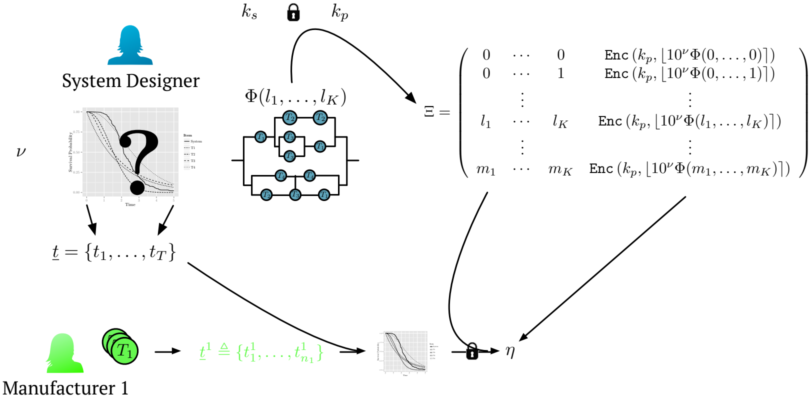


$$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 0) \rceil\right) \\ 0 & \cdots & 1 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1, \dots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1, \dots, m_K) \rceil\right) \end{pmatrix}$$

$\rightarrow \underline{t}^1 \triangleq \{t^1_1, \dots, t^1_{n_1}\} \rightarrow$



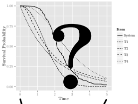
Manufacturer 1


k_s
k_p
System Designer
$\Phi(l_1, \ldots, l_K)$
ν
$$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \ldots, 0) \rceil\right) \\ 0 & \cdots & 1 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \ldots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1, \ldots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1, \ldots, m_K) \rceil\right) \end{pmatrix}$$
$\underline{t} = \{t_1, \ldots, t_T\}$
T_1
$\underline{t}^1 \triangleq \{t_1^1, \ldots, t_{n_1}^1\}$
η
Manufacturer 1

k_s k_p

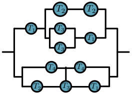
System Designer

ν



$\underline{t} = \{t_1, \dots, t_T\}$

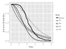
$\Phi(l_1, \dots, l_K)$



$$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0,\dots,0)\rceil\right) \\ 0 & \cdots & 1 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0,\dots,1)\rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1,\dots,l_K)\rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1,\dots,m_K)\rceil\right) \end{pmatrix}$$

Manufacturer 1

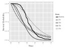
T_1 → $\underline{t}^1 \triangleq \{t^1_1, \dots, t^1_{n_1}\}$ →



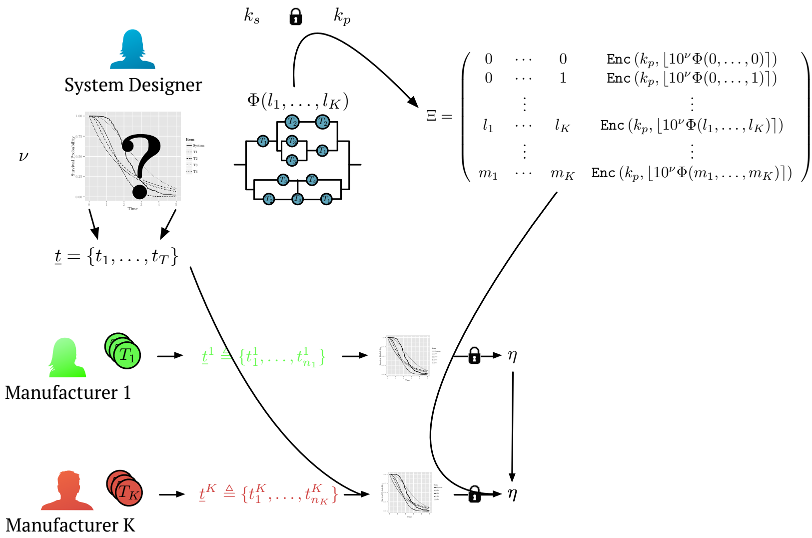
→ η

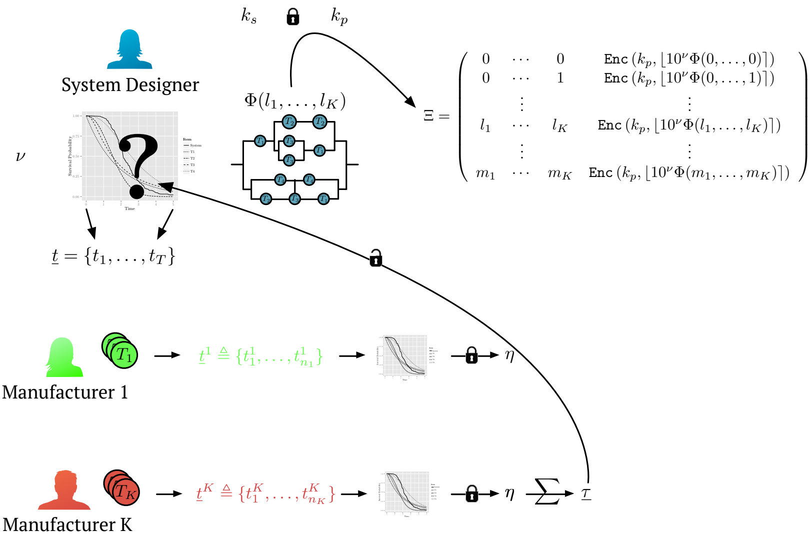
Manufacturer K

T_K → $\underline{t}^K \triangleq \{t^K_1, \dots, t^K_{n_K}\}$ →



→ η


k_s
k_p
System Designer
$\Phi(l_1, \dots, l_K)$
ν
$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \mathtt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 0) \rceil\right) \\ 0 & \cdots & 1 & \mathtt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \mathtt{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1, \dots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \mathtt{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1, \dots, m_K) \rceil\right) \end{pmatrix}$
$\underline{t} = \{t_1, \dots, t_T\}$
Manufacturer 1
T_1
$\underline{t}^1 \triangleq \{t^1_1, \dots, t^1_{n_1}\}$
η
Manufacturer K
T_K
$\underline{t}^K \triangleq \{t^K_1, \dots, t^K_{n_K}\}$
η


k_s
k_p
System Designer
$\Phi(l_1, \dots, l_K)$
ν
$\underline{t} = \{t_1, \dots, t_T\}$
$\Xi = \begin{pmatrix} 0 & \cdots & 0 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 0) \rceil\right) \\ 0 & \cdots & 1 & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(0, \dots, 1) \rceil\right) \\ & \vdots & & \vdots \\ l_1 & \cdots & l_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(l_1, \dots, l_K) \rceil\right) \\ & \vdots & & \vdots \\ m_1 & \cdots & m_K & \texttt{Enc}\left(k_p, \lfloor 10^\nu \Phi(m_1, \dots, m_K) \rceil\right) \end{pmatrix}$
Manufacturer 1
T_1
$\underline{t}^1 \triangleq \{t^1_1, \dots, t^1_{n_1}\}$
η
Manufacturer K
T_K
$\underline{t}^K \triangleq \{t^K_1, \dots, t^K_{n_K}\}$
η
$\sum$
$\underline{\tau}$

i-like.org.uk

Practicalities (I): survival signature

Survival signature easily computed using `ReliabilityTheory` package (Aslett 2012).

```
library(ReliabilityTheory)

g <- graph.formula(s -- 1 -- 2:4:5, 2 -- 3 -- t, 4:5 -- 6 -- t,
        s -- 7 -- 8 -- t, s -- 9 -- 10 -- 11 -- t, 7 -- 10 -- 8)

V(g)$compType <- NA
V(g)$compType[match(c("1","6","11"), V(g)$name)] <- "T1"
V(g)$compType[match(c("2","3","9"), V(g)$name)] <- "T2"
V(g)$compType[match(c("4","5","10"), V(g)$name)] <- "T3"
V(g)$compType[match(c("7","8"), V(g)$name)] <- "T4"

sig <- computeSystemSurvivalSignature(g)
```

```
sig
```

```
##      T1 T2 T3 T4 Probability
## 1     0  0  0  0  0.00000000
## 2     0  0  0  1  0.00000000
## 3     0  0  0  2  1.00000000
## 4     0  0  1  0  0.00000000
## 5     0  0  1  1  0.00000000
## 6     0  0  1  2  1.00000000
## 7     0  0  2  0  0.00000000
## 8     0  0  2  1  0.00000000
## 9     0  0  2  2  1.00000000
## 10    0  0  3  0  0.00000000
## 11    0  0  3  1  0.00000000
## 12    0  0  3  2  1.00000000
## 13    0  1  0  0  0.00000000
## 14    0  1  0  1  0.00000000
## 15    0  1  0  2  1.00000000
## 16    0  1  1  0  0.00000000
## 17    0  1  1  1  0.05555556
## 18    0  1  1  2  1.00000000
## 19    0  1  2  0  0.00000000
## 20    0  1  2  1  0.11111111
```

Practicalities (II): homomorphic encryption

Homomorphic encryption without knowing any abstract algebra/number theory in `HomomorphicEncryption` package (Aslett 2014).

```
library(HomomorphicEncryption)

p <- pars("FandV")
keys <- keygen(p)

Xi <- enc(keys$pk, round(10^5*sig$Probability))
Xi
```

```
## Vector of 192 Fan and Vercauteren cipher texts
```

```
dec(keys$sk, Xi[17])
```

```
## [1] 5556
```

```
dec(keys$sk, sum(Xi))
```

```
## [1] 11685185
```

```
sum(sig$Probability)
```

```
## [1] 116.8519
```

Open Q

How much can someone learn by seeing $\underline{\lambda}_i$?

- Not yet proved how much one can learn about component makeup. For example, it may be that only one possible quantity of other components could result in the vector $\underline{\lambda}_i$.
  - e.g. $\underline{\lambda}_1 = (0, 0, 1, 1, 2, 2) \implies$ there is exactly one component of one other type in the system
  - Q: only this trivial example or do others exist?
- However,
  - without seeing the survival signature probability, seems you can say *nothing* about the layout.
  - Q: may be a problem mainly in smaller systems where combinatorics against you?

References

Aslett, L. J. M. (2012). *ReliabilityTheory: Tools for structural reliability analysis*.

Aslett, L. J. M. (2014). *HomomorphicEncryption: Fully homomorphic encryption*.

Aslett, L. J. M., Coolen, F. P. A., & Wilson, S. P. (2014). Bayesian inference for reliability of systems and networks using the survival signature. *Risk Analysis*.

Aslett, L. J. M., Esperança, P., & Holmes, C. C. (2015). *Secure statistical analysis*. University of Oxford.

Coolen, F. P. A., & Coolen-Maturi, T. (2012). Generalizing the signature to systems with multiple types of components. *Complex systems and dependability*, pp. 115–30. Springer.

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

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