Optimal Qos Attributes In Security Key Management

Akash K Singh, PhD
IBM Corporation Sacramento, USA

Abstract

System NP Hardness problem is addressed and Mathematical framework is proposed for certificates and enhancement of security systems

Keywords- Key Management, Certificates, NP

I. INTRODUCTION

The problem of computing minimum cost delegation chains supporting all the attributes in certificate cert is NP-hard, even for configurations with no authority classes. In fact, the minimum set cover problem reduces to it in polynomial time, as formally stated by the following theorem.

THEOREM A.1 (NP-HARDNESS). The problem of computing minimum-cost delegation chains supporting all the attributes in a certificate is NP-hard.

PROOF. The proof is a reduction from the NP-hard problem of theminimum set cover, which can be formulated as follows: given a collection S of subsets of a finite set U, determine a subset S_ ⊆ S such that every element in U belongs to at least one set in S_ and the cardinality of S_ is minimized. Given a finite set U and a collection S of subsets of U, the problem of computing a minimum set cover for U can be translated into an equivalent instance of the problem of computing a minimum cost supporting chain, as follows. Each element in the finite set U translates to an attribute certified by cert and that belongs to TT. Any subset s in S translates to a delegation certificate in Deleg Certs, issued by an authority in the authoritative clause of TT, supporting the attributes in s, and with cost equal to 1. The minimum-cost supporting chain for cert corresponds to a subset S_ of S that completely covers U and with minimum cardinality.

We consider the following anycast field equations defined over an open bounded piece of network and /or feature space $\Omega \subset R^d$. They describe the dynamics of the mean anycast of each of p node populations.

$$\begin{cases} (\frac{d}{dt}+l_i)V_i(t,r) = \sum_{j=1}^{p}\int_{\Omega} J_{ij}(r,\bar{r})S[(V_j(t-\tau_{ij}(r,\bar{r}),\bar{r})-h_{|j})]d\bar{r} \\ \qquad +I_i^{ext}(r,t), \qquad t \ge 0, 1 \le i \le p, \\ V_i(t,r)=\phi_i(t,r) \qquad t \in [-T,0] \end{cases} \tag{1}$$

We give an interpretation of the various parameters and functions that appear in (1), Ω is finite piece of nodes and/or feature space and is represented as an open bounded set of R^d . The vector r and $\bar{r}$ represent points in Ω . The function $S: R \to (0,1)$ is the normalized sigmoid function:

$$S(z) = \frac{1}{1+e^{-z}} \tag{2}$$

It describes the relation between the input rate v_i of population i as a function of the packets potential, for example, $V_i = v_i = S[\sigma_i(V_i - h_i)].$ We note V the $p-$ dimensional vector $(V_1,...,V_p).$ The p function $\phi_i, i = 1,..., p,$ represent the initial conditions, see below. We note ϕ the $p-$ dimensional vector $(\phi_1,...,\phi_p).$ The p function $I_i^{ext}, i = 1,..., p,$ represent external factors from other network areas. We note I^{ext} the $p-$ dimensional vector $(I_1^{ext},...,I_p^{ext}).$ The $p \times p$ matrix of functions $J = \{J_{ij}\}_{i,j=1,...,p}$ represents the connectivity between populations i and $j,$ see below. The p real values $h_i, i = 1,..., p,$ determine the threshold of activity for each population, that is, the value of the nodes potential corresponding to 50% of the maximal activity. The p real positive values $\sigma_i, i = 1,..., p,$ determine the slopes of the sigmoids at the origin. Finally the p real positive values $l_i, i = 1,..., p,$ determine the speed at which each anycast node potential decreases exponentially toward its real value. We also introduce the function $S: R^p \to R^p,$ defined by $S(x) = [S(\sigma_1(x_1 - h_1)),..., S(\sigma_p - h_p))],$ and the diagonal $p \times p$ matrix $L_0 = diag(l_1,...,l_p).$ Is the intrinsic dynamics of the population given by the linear response of data transfer. $(\frac{d}{dt}+l_i)$ is replaced by $(\frac{d}{dt}+l_i)^2$ to use the alpha function response. We use $(\frac{d}{dt}+l_i)$ for simplicity although our analysis applies to more general intrinsic dynamics. For the sake, of generality, the propagation delays are not assumed to be identical for all populations, hence they are described by a matrix $\tau(r,\bar{r})$ whose element $\tau_{ij}(r,\bar{r})$ is the propagation delay between population j at $\bar{r}$ and population i at r. The reason for this assumption is that it is still unclear from anycast if propagation delays are independent of the populations. We assume for technical reasons that τ is continuous, that is $\tau \in C^0(\overline{\Omega}^2, R_+^{p\times p})$. Moreover packet data indicate that τ is not a symmetric function i.e., $\tau_{ij}(r,\bar{r}) \neq \tau_{ij}(\bar{r},r)$, thus no assumption is made about this symmetry unless otherwise stated. In order to compute the righthand side of (1), we need to know the node potential factor V on interval $[-T,0]$. The value of T is obtained by considering the maximal delay:

$$\tau_m = \max_{i,j(r,\bar{r}\in\bar{\Omega}\times\bar{\Omega})} \tau_{i,j}(r,\bar{r}) \qquad (3)$$

Hence we choose $T = \tau_m$

A. Mathematical Framework

A convenient functional setting for the non-delayed packet field equations is to use the space $F = L^2(\Omega, R^p)$ which is a Hilbert space endowed with the usual inner product:

$$\langle V, U\rangle_F = \sum_{i=1}^{p} \int_\Omega V_i(r)U_i(r)dr \qquad (1)$$

To give a meaning to (1), we defined the history space $C = C^0([-\tau_m,0],F)$ with $\|\phi\| = \sup_{t\in[-\tau_m,0]} \|\phi(t)\| F$, which is the Banach phase space associated with equation (3). Using the notation $V_t(\theta) = V(t+\theta), \theta \in [-\tau_m, 0]$, we write (1) as

$$\begin{cases} V(t) = -L_0 V(t) \overset{\cdot}{+} L_1 S(V_t) + I^{ext}(t), \\ \qquad V_0 = \phi \in C, \end{cases} \qquad (2)$$

Where

$$\begin{cases} \qquad L_1 : C \to F, \\ \phi \to \int_\Omega J(.,\bar{r})\phi(\bar{r}, -\tau(.,\bar{r}))d\bar{r} \end{cases}$$

Is the linear continuous operator satisfying $\|L_1\| \le \|J\|_{L^2(\Omega^2, R^{p\times p})}$. Notice that most of the papers on this subject assume Ω infinite, hence requiring $\tau_m = \infty$.

Proposition 1.0 If the following assumptions are satisfied.

1. $J \in L^2(\Omega^2, R^{p\times p})$,
2. The external current $I^{ext} \in C^0(R, F)$,
3. $\tau \in C^0(\overline{\Omega^2}, R_+^{p\times p}), \sup_{\overline{\Omega^2}} \tau \le \tau_m$.

Then for any $\phi \in C$, there exists a unique solution $V \in C^1([0,\infty), F) \cap C^0([-\tau_m, \infty, F)$ to (3)

Notice that this result gives existence on R_+, finite-time explosion is impossible for this delayed differential equation. Nevertheless, a particular solution could grow indefinitely, we now prove that this cannot happen.

B. Boundedness of Solutions

A valid model of neural networks should only feature bounded packet node potentials.

Theorem 1.0 All the trajectories are ultimately bounded by the same constant R if $I \equiv \max_{t\in R^+} \|I^{ext}(t)\|_F < \infty$.

Proof :Let us defined $f : R\times C \to R^+$ as

$$f(t,V_t) \overset{def}{=} \langle -L_0 V_t(0) + L_1 S(V_t) + I^{ext}(t), V(t)\rangle_F = \frac{1}{2}\frac{d\|V\|_F^2}{dt}$$

We note $l = \min_{i=1,\dots p} l_i$

$$f(t,V_t) \le -l\|V(t)\|_F^2 + (\sqrt{p|\Omega|}\,\|J\|_F + I)\|V(t)\|_F$$

Thus, if

$$\|V(t)\|_F \ge 2\frac{\sqrt{p|\Omega|}.\|J\|_F + I}{l} \overset{def}{=} R, f(t,V_t) \le -\frac{lR^2}{2} \overset{def}{=} -\delta < 0$$

Let us show that the open route of F of center 0 and radius R, B_R, is stable under the dynamics of equation. We know that $V(t)$ is defined for all $t \ge 0s$ and that $f < 0$ on ∂B_R, the boundary of B_R. We consider three cases for the initial condition V_0. If $\|V_0\|_C < R$ and set $T = \sup\{t \mid \forall s \in [0,t], V(s) \in \overline{B_R}\}$. Suppose that $T \in R$, then $V(T)$ is defined and belongs to $\overline{B_R}$, the closure of B_R, because $\overline{B_R}$ is closed, in effect to ∂B_R, we also have

$$\frac{d}{dt}\|V\|_F^2 \,|_{t=T} = f(T,V_T) \leq -\delta < 0$$

because $V(T) \in \partial B_R$. Thus we deduce that for $\varepsilon > 0$ and small enough, $V(T+\varepsilon) \in \overline{B_R}$ which contradicts the definition of T. Thus $T \notin R$ and $\overline{B_R}$ is stable. Because f<0 on $\partial B_R, V(0) \in \partial B_R$ implies that $\forall t > 0, V(t) \in B_R$. Finally we consider the case $V(0) \in C\overline{B_R}$. Suppose that $\forall t > 0, V(t) \notin \overline{B_R}$, then $\forall t > 0, \frac{d}{dt}\|V\|_F^2 \leq -2\delta$, thus $\|V(t)\|_F$ is monotonically decreasing and reaches the value of R in finite time when $V(t)$ reaches ∂B_R. This contradicts our assumption. Thus $\exists T > 0 \,|\, V(T) \in B_R$.

Proposition 1.1 : Let s and t be measured simple functions on X. for $E \varepsilon M$, define

$$\phi(E) = \int_E s\,d\mu \qquad (1)$$

Then ϕ is a measure on M.

$$\int_X (s+t)d\mu = \int_X s\,d\mu + \int_X t d\mu \qquad (2)$$

Proof : If s and if $E_1, E_2, \ldots$ are disjoint members of M whose union is E, the countable additivity of μ shows that

$$\phi(E) = \sum_{i=1}^{n} \alpha_i \mu(A_i \cap E) = \sum_{i=1}^{n} \alpha_i \sum_{r=1}^{\infty} \mu(A_i \cap E_r)$$
$$= \sum_{r=1}^{\infty}\sum_{i=1}^{n} \alpha_i \mu(A_i \cap E_r) = \sum_{r=1}^{\infty} \phi(E_r)$$

Also, $\varphi(\phi) = 0$, so that φ is not identically ∞. Next, let s be as before, let $\beta_1, \ldots, \beta_m$ be the distinct values of t,and let $B_j = \{x : t(x) = \beta_j\}$ If $E_{ij} = A_i \cap B_j$, the

$$\int_{E_{ij}} (s+t)d\mu = (\alpha_i + \beta_j)\mu(E_{ij})$$

and $\int_{E_{ij}} sd\mu + \int_{E_{ij}} td\mu = \alpha_i \mu(E_{ij}) + \beta_j \mu(E_{ij})$

Thus (2) holds with E_{ij} in place of X. Since X is the disjoint union of the sets E_{ij} $(1 \leq i \leq n, 1 \leq j \leq m)$, the first half of our proposition implies that (2) holds.

Theorem 1.1: If K is a compact set in the plane whose complement is connected, if f is a continuous complex function on K which is holomorphic in the interior of , and if $\varepsilon > 0$, then there exists a polynomial P such that $|f(z) = P(z)| < \varepsilon$ for all $z \varepsilon K$. If the interior of K is empty, then part of the hypothesis is vacuously satisfied, and the conclusion holds for every $f \varepsilon C(K)$. Note that K need to be connected.

Proof: By Tietze's theorem, f can be extended to a continuous function in the plane, with compact support. We fix one such extension and denote it again by f. For any $\delta > 0$, let $\omega(\delta)$ be the supremum of the numbers $|f(z_2) - f(z_1)|$ Where z_1 and z_2 are subject to the condition $|z_2 - z_1| \leq \delta$. Since f is uniformly continous, we have $\lim_{\delta \to 0} \omega(\delta) = 0$ (1) From now on, δ will be fixed. We shall prove that there is a polynomial P such that

$$|f(z) - P(z)| < 10{,}000\ \omega(\delta) \quad (z \varepsilon K) \qquad (2)$$

By (1), this proves the theorem. Our first objective is the construction of a function $\Phi \varepsilon C_c^{'}(R^2)$, such that for all z

$$|f(z) - \Phi(z)| \leq \omega(\delta), \qquad (3)$$

$$|(\partial\Phi)(z)| < \frac{2\omega(\delta)}{\delta}, \qquad (4)$$

And

$$\Phi(z) = -\frac{1}{\pi}\iint_X \frac{(\partial\Phi)(\zeta)}{\zeta - z} d\zeta\, d\eta \qquad (\zeta = \xi + i\eta), \qquad (5)$$

Where X is the set of all points in the support of Φ whose distance from the complement of K does not δ. (Thus X contains no point which is "far within" K.) We construct Φ as the convolution of f with a smoothing function A. Put $a(r) = 0$ if $r > \delta$, put

$$a(r) = \frac{3}{\pi\delta^2}(1 - \frac{r^2}{\delta^2})^2 \qquad (0 \leq r \leq \delta), \qquad (6)$$

And define

$$A(z) = a(|z|) \qquad (7)$$

For all complex z. It is clear that $A \varepsilon C_c^{'}(R^2)$. We claim that

$$\iint_{R^s} A = 1, \qquad (8)$$

$$\iint_{R^2} \partial A = 0, \qquad (9)$$

$$\iint_{R^3} |\partial A| = \frac{24}{15\delta} < \frac{2}{\delta}, \qquad (10)$$

The constants are so adjusted in (6) that (8) holds. (Compute the integral in polar coordinates), (9) holds simply because A has compact support. To compute (10), express ∂A in polar coordinates, and note that $\partial A / \partial\theta = 0$,

$$\partial A / \partial r = -a',$$

Now define

$$\Phi(z) = \iint_{R^2} f(z-\zeta) A d\xi d\eta = \iint_{R^2} A(z-\zeta) f(\zeta) d\xi d\eta \qquad (11)$$

Since f and A have compact support, so does Φ. Since

$$\Phi(z) - f(z)$$
$$= \iint_{R^2} [f(z-\zeta) - f(z)] A(\xi) d\xi d\eta \quad (12)$$

And $A(\zeta) = 0$ if $|\zeta| > \delta$, (3) follows from (8). The difference quotients of A converge boundedly to the corresponding partial derivatives, since $A \varepsilon C_c'(R^2)$. Hence the last expression in (11) may be differentiated under the integral sign, and we obtain

$$(\partial\Phi)(z) = \iint_{R^2} (\overline{\partial A})(z-\zeta) f(\zeta) d\xi d\eta$$
$$= \iint_{R^2} f(z-\zeta)(\partial A)(\zeta) d\xi d\eta$$
$$= \iint_{R^2} [f(z-\zeta) - f(z)](\partial A)(\zeta) d\xi d\eta \qquad (13)$$

The last equality depends on (9). Now (10) and (13) give (4). If we write (13) with Φ_x and Φ_y in place of $\partial\Phi$, we see that Φ has continuous partial derivatives, if we can show that $\partial\Phi = 0$ in G, where G is the set of all $z\varepsilon K$ whose distance from the complement of K exceeds δ. We shall do this by showing that

$$\Phi(z) = f(z) \qquad (z\varepsilon G); \qquad (14)$$

Note that $\partial f = 0$ in G, since f is holomorphic there. Now if $z\varepsilon G$, then $z - \zeta$ is in the interior of K for all ζ with $|\zeta| < \delta$. The mean value property for harmonic functions therefore gives, by the first equation in (11),

$$\Phi(z) = \int_0^{\delta} a(r) r dr \int_0^{2\pi} f(z - re^{i\theta}) d\theta$$
$$= 2\pi f(z) \int_0^{\delta} a(r) r dr = f(z) \iint_{R^2} A = f(z) \qquad (15)$$

For all $z \ \varepsilon \ G$, we have now proved (3), (4), and (5) The definition of X shows that X is compact and that X can be covered by finitely many open discs $D_1, \ldots, D_n$, of radius 2δ, whose centers are not in K. Since $S^2 - K$ is connected, the center of each D_j can be joined to ∞ by a polygonal path in $S^2 - K$. It follows that each D_j contains a compact connected set E_j, of diameter at least 2δ, so that $S^2 - E_j$ is connected and so that $K \cap E_j = \phi$. with $r = 2\delta$. There are functions $g_j \varepsilon H(S^2 - E_j)$ and constants b_j so that the inequalities.

$$|Q_j(\zeta, z)| < \frac{50}{\delta}, \qquad (16)$$

$$\left| Q_j(\zeta, z) - \frac{1}{z-\zeta} \right| < \frac{4{,}000\delta^2}{|z-\zeta|^2} \qquad (17)$$

Hold for $z \notin E_j$ and $\zeta \in D_j$, if

$$Q_j(\zeta, z) = g_j(z) + (\zeta - b_j) g_j^2(z) \qquad (18)$$

Let Ω be the complement of $E_1 \cup \ldots \cup E_n$. Then Ω is an open set which contains K. Put $X_1 = X \cap D_1$ and $X_j = (X \cap D_j) - (X_1 \cup \ldots \cup X_{j-1})$, for $2 \le j \le n$,

Define

$$R(\zeta, z) = Q_j(\zeta, z) \qquad (\zeta \varepsilon X_j, z \varepsilon \Omega) \qquad (19)$$

And

$$F(z) = \frac{1}{\pi} \iint_X (\partial\Phi)(\zeta) R(\zeta, z) d\zeta d\eta \qquad (20)$$
$$(z \ \varepsilon \ \Omega)$$

Since,

$$F(z) = \sum_{j=1} \frac{1}{\pi} \iint_{X_i} (\partial\Phi)(\zeta) Q_j(\zeta, z) d\xi d\eta, \qquad (21)$$

(18) shows that F is a finite linear combination of the functions g_j and g_j^2. Hence $F \varepsilon H(\Omega)$. By (20), (4), and (5) we have

$$\left|F(z)-\Phi(z)\right| < \frac{2\omega(\delta)}{\pi\delta}\iint_X | R(\zeta,z) - \frac{1}{z-\zeta}| \, d\xi d\eta \quad (z\ \varepsilon\ \Omega) \quad (22)$$

Observe that the inequalities (16) and (17) are valid with R in place of Q_j if $\zeta\ \varepsilon\ X$ and $z\ \varepsilon\ \Omega.$

Now fix $z\ \varepsilon\ \Omega.$, put $\zeta = z+\rho e^{i\theta}$, and estimate the integrand in (22) by (16) if $\rho < 4\delta,$ by (17) if $4\delta \le \rho.$ The integral in (22) is then seen to be less than the sum of

$$2\pi\int_0^{4\delta}\left(\frac{50}{\delta}+\frac{1}{\rho}\right)\rho d\rho = 808\pi\delta \qquad (23)$$

And

$$2\pi\int_{4\delta}^{\infty}\frac{4,000\delta^2}{\rho^2}\rho d\rho = 2,000\pi\delta. \qquad (24)$$

Hence (22) yields

$$\left|F(z)-\Phi(z)\right| < 6,000\omega(\delta) \qquad (z\ \varepsilon\ \Omega) \quad (25)$$

Since $F\ \varepsilon\ H(\Omega),\ K\subset\Omega,$ and S^2-K is connected, Runge's theorem shows that F can be uniformly approximated on K by polynomials. Hence (3) and (25) show that (2) can be satisfied. This completes the proof.

Lemma 1.0 : Suppose $f\varepsilon C_c^{'}(R^2),$ the space of all continuously differentiable functions in the plane, with compact support. Put

$$\partial = \frac{1}{2}\left(\frac{\partial}{\partial x}+i\frac{\partial}{\partial y}\right) \qquad (1)$$

Then the following "Cauchy formula" holds:

$$f(z) = -\frac{1}{\pi}\iint_{R^2}\frac{(\partial f)(\zeta)}{\zeta - z}d\xi d\eta$$

$$(\zeta = \xi + i\eta) \qquad (2)$$

Proof: This may be deduced from Green's theorem. However, here is a simple direct proof:

Put $\varphi(r,\theta) = f(z+re^{i\theta}),\ r>0,\ \theta$ real

If $\zeta = z+re^{i\theta}$, the chain rule gives

$$(\partial f)(\zeta) = \frac{1}{2}e^{i\theta}\left[\frac{\partial}{\partial r}+\frac{i}{r}\frac{\partial}{\partial\theta}\right]\varphi(r,\theta) \qquad (3)$$

The right side of (2) is therefore equal to the limit, as $\varepsilon\to 0,$ of

$$-\frac{1}{2}\int_\varepsilon^\infty\int_0^{2\pi}\left(\frac{\partial\varphi}{\partial r}+\frac{i}{r}\frac{\partial\varphi}{\partial\theta}\right)d\theta dr \qquad (4)$$

For each $r>0,\varphi$ is periodic in θ, with period 2π. The integral of $\partial\varphi/\partial\theta$ is therefore 0, and (4) becomes

$$-\frac{1}{2\pi}\int_0^{2\pi}d\theta\int_\varepsilon^\infty\frac{\partial\varphi}{\partial r}dr = \frac{1}{2\pi}\int_0^{2\pi}\varphi(\varepsilon,\theta)d\theta \qquad (5)$$

As $\varepsilon\to 0,\ \varphi(\varepsilon,\theta)\to f(z)$ uniformly. This gives (2)

If $X^\alpha\in a$ and $X^\beta\in k\left[X_1,\ldots X_n\right]$, then $X^\alpha X^\beta = X^{\alpha+\beta}\in a$, and so A satisfies the condition $(*)$. Conversely,

$$(\sum_{\alpha\in A}c_\alpha X^\alpha)(\sum_{\beta\in□^n}d_\beta X^\beta) = \sum_{\alpha,\beta}c_\alpha d_\beta X^{\alpha+\beta} \qquad (finite\,sums),$$

and so if A satisfies $(*)$, then the subspace generated by the monomials $X^\alpha,\alpha\in a$, is an ideal. The proposition gives a classification of the monomial ideals in $k\left[X_1,\ldots X_n\right]$: they are in one to one correspondence with the subsets A of $□^n$ satisfying $(*)$. For example, the monomial ideals in $k\left[X\right]$ are exactly the ideals $(X^n),\ n\ge 1$, and the zero ideal (corresponding to the empty set A). We write $\left\langle X^\alpha \,|\, \alpha\in A\right\rangle$ for the ideal corresponding to A (subspace generated by the $X^\alpha,\alpha\in a$).

LEMMA 1.1. Let S be a subset of $□^n$. The the ideal a generated by $X^\alpha,\alpha\in S$ is the monomial ideal corresponding to

$$A \overset{df}{=} \left\{\beta\in□^n \,|\, \beta-\alpha\in□^n,\quad some\ \alpha\in S\right\}$$

Thus, a monomial is in a if and only if it is divisible by one of the $X^\alpha,\alpha\in| S$

PROOF. Clearly A satisfies $(*)$, and $a\subset\left\langle X^\beta \,|\, \beta\in A\right\rangle$. Conversely, if $\beta\in A$, then $\beta-\alpha\in□^n$ for some $\alpha\in S$, and $X^\beta = X^\alpha X^{\beta-\alpha}\in a$. The last statement follows from the fact that $X^\alpha \,|\, X^\beta \Leftrightarrow \beta-\alpha\in□^n$. Let $A\subset□^n$ satisfy$(*)$. From the geometry of A, it is clear that there is a finite set of elements $S=\left\{\alpha_1,\ldots\alpha_s\right\}$ of A such that $A=\left\{\beta\in□^n \,|\, \beta-\alpha_i\in□^2,\ some\ \alpha_i\in S\right\}$ (The $\alpha_i's$ are the corners of A) Moreover,

$a \overset{df}{=} \langle X^{\alpha} \mid \alpha \in A \rangle$ is generated by the monomials $X^{\alpha_i}, \alpha_i \in S$.

DEFINITION 1.0. For a nonzero ideal a in $k[X_1,...,X_n]$, we let $(LT(a))$ be the ideal generated by
$\{LT(f) \mid f \in a\}$

LEMMA 1.2 Let a be a nonzero ideal in $k[X_1,...,X_n]$; then $(LT(a))$ is a monomial ideal, and it equals $(LT(g_1),...,LT(g_n))$ for some $g_1,...,g_n \in a$.

PROOF. Since $(LT(a))$ can also be described as the ideal generated by the leading monomials (rather than the leading terms) of elements of a .

THEOREM 1.2. Every *ideal* a in $k[X_1,...,X_n]$ is finitely generated; more precisely, $a=(g_1,...,g_s)$ where $g_1,...,g_s$ are any elements of a whose leading terms generate $LT(a)$

PROOF. Let $f \in a$. On applying the division algorithm, we find $f = a_1g_1+...+a_sg_s+r, \quad a_i, r \in k[X_1,...,X_n]$, where either $r=0$ or no monomial occurring in it is divisible by any $LT(g_i)$. But $r = f - \sum a_i g_i \in a$, and therefore $LT(r) \in LT(a) = (LT(g_1),...,LT(g_s))$, implies that every monomial occurring in r is divisible by one in $LT(g_i)$. Thus $r=0$, and $g \in (g_1,...,g_s)$.

DEFINITION 1.1. A finite subset $S=\{g_1,|...,g_s\}$ of an ideal a is a standard ($(Gr\ddot{o}bner)$ bases for a if $(LT(g_1),...,LT(g_s)) = LT(a)$. In other words, S is a standard basis if the leading term of every element of a is divisible by at least one of the leading terms of the g_i.

THEOREM 1.3 *The ring* $k[X_1,...,X_n]$ *is Noetherian i.e., every ideal is finitely generated.*

PROOF. For $n=1$, $k[X]$ is a principal ideal domain, which means that every ideal is generated by single element. We shall prove the theorem by induction on n . Note that the obvious map $k[X_1,...X_{n-1}][X_n] \to k[X_1,...X_n]$ is an isomorphism – this simply says that every polynomial f in n variables $X_1,...X_n$ can be expressed uniquely as a polynomial in X_n with coefficients in $k[X_1,...,X_n]$:

$$f(X_1,...X_n) = a_0(X_1,...X_{n-1})X_n^r + ... + a_r(X_1,...X_{n-1})$$

Thus the next lemma will complete the proof

LEMMA 1.3. If A is Noetherian, then so also is $A[X]$

PROOF. For a polynomial

$$f(X) = a_0X^r + a_1X^{r-1} + ... + a_r, \quad a_i \in A, \quad a_0 \neq 0,$$

r is called the degree of f , and a_0 is its leading coefficient. We call 0 the leading coefficient of the polynomial 0. Let a be an ideal in $A[X]$. The leading coefficients of the polynomials in a form an ideal a' in A, and since A is Noetherian, a' will be finitely generated. Let $g_1,...,g_m$ be elements of a whose leading coefficients generate a' , and let r be the maximum degree of g_i. Now let $f \in a$, and suppose f has degree $s > r$, say, $f = aX^s + ...$ Then $a \in a'$, and so we can write $a = \sum b_i a_i, \quad b_i \in A,$

$a_i = leading\ coefficient\ of\ g_i$

Now

$f - \sum b_i g_i X^{s-r_i}, \quad r_i = \deg(g_i),$ has degree $< \deg(f)$. By continuing in this way, we find that $f \equiv f_t \quad \mod(g_1,...g_m)$ With f_t a polynomial of degree $t<r$. For each $d<r$, let a_d be the subset of A consisting of 0 and the leading coefficients of all polynomials in a of degree d; it is again an ideal in A . Let $g_{d,1},...,g_{d,m_d}$ be polynomials of degree d whose leading coefficients generate a_d . Then the same argument as above shows that any polynomial f_d in a of degree d can be written $f_d \equiv f_{d-1} \quad \mod(g_{d,1},...g_{d,m_d})$ With f_{d-1}

of degree $\leq d-1$. On applying this remark repeatedly we find that $f_t \in (g_{r-1,1},\dots g_{r-1,m_{r-1}},\dots g_{0,1},\dots g_{0,m_0})$ Hence

$$f_t \in (g_1,\dots g_m g_{r-1,1},\dots g_{r-1,m_{r-1}},\dots, g_{0,1},\dots, g_{0,m_0})$$

and so the polynomials $g_1,\dots, g_{0,m_0}$ generate a

One of the great successes of category theory in computer science has been the development of a "unified theory" of the constructions underlying denotational semantics. In the untyped λ-calculus, any term may appear in the function position of an application. This means that a model D of the λ-calculus must have the property that given a term t whose interpretation is $d \in D$, Also, the interpretation of a functional abstraction like $\lambda x\,.\,x$ is most conveniently defined as a function from $D to D$, which must then be regarded as an element of *D*. Let $\psi : [D \to D] \to D$ be the function that picks out elements of *D* to represent elements of $[D \to D]$ and $\phi : D \to [D \to D]$ be the function that maps elements of *D* to functions of *D*. Since $\psi(f)$ is intended to represent the function f as an element of *D*, it makes sense to require that $\phi(\psi(f)) = f$, that is, $\psi o \psi = id_{[D\to D]}$ Furthermore, we often want to view every element of *D* as representing some function from *D to D* and require that elements representing the same function be equal – that is

$$\psi(\varphi(d)) = d$$

$$or$$

$$\psi\, o\, \phi = id_D$$

The latter condition is called extensionality. These conditions together imply that $\phi\, and\, \psi$ are inverses--- that is, *D* is isomorphic to the space of functions from *D to D* that can be the interpretations of functional abstractions: $D \cong [D \to D]$ .Let us suppose we are working with the untyped $\lambda - calculus$, we need a solution ot the equation $D \cong A + [D \to D]$, where A is some predetermined domain containing interpretations for elements of *C*. Each element of *D* corresponds to either an element of *A* or an element of $[D \to D]$, with a tag. This equation can be solved by finding least fixed points of the function $F(X) = A + [X \to X]$ from domains to domains --- that is, finding domains *X* such that $X \cong A + [X \to X]$, and such that for any domain *Y* also satisfying this equation, there is an embedding of *X* to *Y* --- a pair of maps

$$X \overset{f}{\underset{f^R}{\square}} Y$$

Such that

$$f^R\, o\, f = id_X$$

$$f\, o\, f^R \subseteq id_Y$$

Where $f \subseteq g$ means that $f\ approximates\ g$ in some ordering representing their information content. The key shift of perspective from the domain-theoretic to the more general category-theoretic approach lies in considering *F* not as a function on domains, but as a *functor* on a category of domains. Instead of a least fixed point of the function, *F.*

Definition 1.3: Let ***K*** be a category and $F : K \to K$ as a functor. A fixed point of *F* is a pair (A,a), where A is a ***K-object*** and $a : F(A) \to A$ is an isomorphism. A prefixed point of F is a pair (A,a), where A is a ***K-object*** and a is any arrow from F(A) to A

Definition 1.4 : An $\omega - chain$ in a category ***K*** is a diagram of the following form:

$$\Delta = D_o \xrightarrow{f_o} D_1 \xrightarrow{f_1} D_2 \xrightarrow{f_2} \dots\dots$$

Recall that a cocone μ of an $\omega - chain\ \Delta$ is a *K*-object *X* and a collection of K *–arrows* $\{\mu_i : D_i \to X \mid i \geq 0\}$ such that $\mu_i = \mu_{i+1} o\, f_i$ for all $i \geq 0$. We sometimes write $\mu : \Delta \to X$ as a reminder of the arrangement of $\mu's$ components Similarly, a colimit $\mu : \Delta \to X$ is a cocone with the property that if $\nu : \Delta \to X'$ is also a cocone then there exists a unique mediating arrow $k : X \to X'$ such that for all $i \geq 0,$, $\nu_i = k\, o\, \mu_i$. Colimits of $\omega - chains$ are sometimes referred to as $\omega - co\lim its$. Dually, an $\omega^{op} - chain$ in ***K*** is a diagram of the following form:

$$\Delta = D_o \xleftarrow{f_o} D_1 \xleftarrow{f_1} D_2 \xleftarrow{f_2} \dots\dots$$

A cone $\mu : X \to \Delta$ of an $\omega^{op} - chain\ \Delta$ is a **K**-object X and a collection of **K**-arrows $\{\mu_i : D_i \mid i \geq 0\}$ such that for all $i \geq 0,\ \mu_i = f_i\, o\, \mu_{i+1}$. An ω^{op}-limit of an $\omega^{op} - chain\ \ \Delta$ is a cone $\mu : X \to \Delta$ with the property that if $\nu : X' \to \Delta$ is also a cone, then there exists a unique mediating arrow $k : X^{'} \to X$ such that for all $i \geq 0, \mu_i \, o \, k = v_i$. We write $\perp_k$ (or just $\perp$) for the distinguish initial object of ***K,*** when it has one, and $\perp \to A$ for the unique arrow from $\perp$ to each ***K***-object A. It is also convenient to write $\Delta^{-} = D_1 \xrightarrow{f_1} D_2 \xrightarrow{f_2} .....$ to denote all of Δ except D_o and f_0. By analogy, μ^{-} is $\{\mu_i \mid i \geq 1\}$. For the images of Δ and μ under ***F*** we write

$$F(\Delta) = F(D_o) \xrightarrow{F(f_o)} F(D_1) \xrightarrow{F(f_1)} F(D_2) \xrightarrow{F(f_2)} .....$$

and $F(\mu) = \{F(\mu_i) \mid i \geq 0\}$

We write F^i for the ***i***-fold iterated composition of ***F*** – that is, $F^o(f) = f, F^1(f) = F(f), F^2(f) = F(F(f))$,etc. With these definitions we can state that every monitonic function on a complete lattice has a least fixed point:

Lemma 1.4. Let **K** be a category with initial object $\perp$ and let $F : K \to K$ be a functor. Define the $\omega - chain \, \Delta$ by

$$\Delta = \perp \xrightarrow{!\perp \to F(\perp)} F(\perp) \xrightarrow{F(!\perp \to F(\perp))} F^2(\perp) \xrightarrow{F^2(!\perp \to F(\perp))} ....$$

If both $\mu : \Delta \to D$ and $F(\mu) : F(\Delta) \to F(D)$ are colimits, then (D,d) is an intial F-algebra, where $d : F(D) \to D$ is the mediating arrow from $F(\mu)$ to the cocone μ^{-}

Theorem 1.4 Let a DAG G given in which each node is a random variable, and let a discrete conditional probability distribution of each node given values of its parents in G be specified. Then the product of these conditional distributions yields a joint probability distribution P of the variables, and (G,P) satisfies the Markov condition.

Proof. Order the nodes according to an ancestral ordering. Let $X_1, X_2, ... X_n$ be the resultant ordering. Next define.

$$P(x_1, x_2, ...x_n) = P(x_n \mid pa_n) P(x_{n-1} \mid Pa_{n-1}) ...$$
$$..P(x_2 \mid pa_2) P(x_1 \mid pa_1),$$

Where PA_i is the set of parents of X_i of in G and $P(x_i \mid pa_i)$ is the specified conditional probability distribution. First we show this does indeed yield a joint probability distribution. Clearly, $0 \leq P(x_1, x_2, ...x_n) \leq 1$ for all values of the variables. Therefore, to show we have a joint distribution, as the variables range through all their possible values, is equal to one. To that end, Specified conditional distributions are the conditional distributions they notationally represent in the joint distribution. Finally, we show the Markov condition is satisfied. To do this, we need show for $1 \leq k \leq n$ that
whenever

$$P(pa_k) \neq 0, if \;\; P(nd_k \mid pa_k) \neq 0$$
$$and \;\; P(x_k \mid pa_k) \neq 0$$
$$then \; P(x_k \mid nd_k, pa_k) = P(x_k \mid pa_k),$$

Where ND_k is the set of nondescendents of X_k of in G. Since $PA_k \subseteq ND_k$, we need only show $P(x_k \mid nd_k) = P(x_k \mid pa_k)$. First for a given k, order the nodes so that all and only nondescendents of X_k precede X_k in the ordering. Note that this ordering depends on k, whereas the ordering in the first part of the proof does not. Clearly then

$$ND_k = \{X_1, X_2, ...X_{k-1}\}$$

Let

$$D_k = \{X_{k+1}, X_{k+2}, ...X_n\}$$

follows $\sum_{d_k}$

We define the m^{th} *cyclotomic field to be the field* $Q[x]/(\Phi_m(x))$ Where $\Phi_m(x)$ is the m^{th} cyclotomic polynomial. $Q[x]/(\Phi_m(x))$ $\Phi_m(x)$ *has degree* $\varphi(m)$ *over* Q *since* $\Phi_m(x)$ has degree $\varphi(m)$. *The roots of* $\Phi_m(x)$ are just the primitive m^{th} roots of unity, so the complex embeddings of $Q[x]/(\Phi_m(x))$ *are simply the* $\varphi(m)$ *maps*

$$\sigma_k : Q[x]/(\Phi_m(x)) \mapsto C,$$
$$1 \leq k \prec m, (k, m) = 1, \quad where$$
$$\sigma_k(x) = \xi_m^k,$$

ξ_m being our fixed choice of primitive m^{th} root of unity. Note that $\xi_m^k \in Q(\xi_m)$ for every k; it follows that $Q(\xi_m) = Q(\xi_m^k)$ for all k relatively prime to m . In particular, the images of the σ_i coincide, so $Q[x]/(\Phi_m(x))$ *is Galois over* Q. *This means that we can write* $Q(\xi_m)$ *for* $Q[x]/(\Phi_m(x))$ *without much fear of ambiguity; we will do so from now on, the identification being* $\xi_m \mapsto x$. *One advantage of* *this is that one can easily talk about cyclotomic fields being extensions of one another,or intersections or compositums; all of these things take place considering them as subfield of* C. We now investigate some basic properties of cyclotomic fields. The first issue is whether or not they are all distinct; to determine this, we need to know which roots of unity lie in $Q(\xi_m)$.Note, for example, that if m is odd, then $-\xi_m$ is a $2m^{th}$ root of unity. We will show that this is the only way in which one can obtain any non-m^{th} roots of unity.

LEMMA 1.5 If m divides n , then $Q(\xi_m)$ *is contained in* $Q(\xi_n)$

PROOF. *Since* $\xi^{n/m} = \xi_m$, *we have* $\xi_m \in Q(\xi_n)$, *so the result is clear*

LEMMA 1.6 *If* m and n are relatively prime, then

$$Q(\xi_m, \xi_n) = Q(\xi_{nm})$$

and

$$Q(\xi_m) \cap Q(\xi_n) = Q$$

(Recall the $Q(\xi_m, \xi_n)$ is the compositum of $Q(\xi_m)$ *and* $Q(\xi_n)$)

PROOF. One checks easily that $\xi_m \xi_n$ is a primitive mn^{th} root of unity, so that

$$Q(\xi_{mn}) \subseteq Q(\xi_m, \xi_n)$$

$$\left[Q(\xi_m, \xi_n) : Q\right] \le \left[Q(\xi_m) : Q\right]\left[Q(\xi_n : Q\right]$$

$$= \varphi(m)\varphi(n) = \varphi(mn);$$

Since $\left[Q(\xi_{mn}) : Q\right] = \varphi(mn);$ this implies that $Q(\xi_m, \xi_n) = Q(\xi_{nm})$ We know that $Q(\xi_m, \xi_n)$ has degree $\varphi(mn)$ over Q, so we must have

$$\left[Q(\xi_m, \xi_n) : Q(\xi_m)\right] = \varphi(n)$$

and

$$\left[Q(\xi_m, \xi_n) : Q(\xi_m)\right] = \varphi(m)$$

$$\left[Q(\xi_m) : Q(\xi_m) \cap Q(\xi_n)\right] \ge \varphi(m)$$

And thus that $Q(\xi_m) \cap Q(\xi_n) = Q$

PROPOSITION 1.2 For any m and n

$$Q(\xi_m, \xi_n) = Q(\xi_{[m,n]})$$

And

$$Q(\xi_m) \cap Q(\xi_n) = Q(\xi_{(m,n)});$$

here $[m,n]$ and (m,n) denote the least common multiple and the greatest common divisor of m and n, respectively.

PROOF. Write $m = p_1^{e_1} \ldots\ldots p_k^{e_k}$ *and* $p_1^{f_1} \ldots. p_k^{f_k}$ where the p_i are distinct primes. (We allow e_i *or* f_i to be zero)

$$Q(\xi_m) = Q(\xi_{p_1^{e_1}})Q(\xi_{p_2^{e_2}})\ldots Q(\xi_{p_k^{e_k}})$$

and

$$Q(\xi_n) = Q(\xi_{p_1^{f_1}})Q(\xi_{p_2^{f_2}})\ldots Q(\xi_{p_k^{f_k}})$$

Thus

$$Q(\xi_m, \xi_n) = Q(\xi_{p_1^{e_1}})\ldots\ldots\ldots Q(\xi_{p_2^{e_k}})Q(\xi_{p_1^{f_1}})\ldots Q(\xi_{p_k^{f_k}})$$

$$= Q(\xi_{p_1^{e_1}})Q(\xi_{p_1^{f_1}})\ldots Q(\xi_{p_k^{e_k}})Q(\xi_{p_k^{f_k}})$$

$$= Q(\xi_{p_1^{\max(e_1, f_1)}})\ldots\ldots. Q(\xi_{p_1^{\max(e_k, f_k)}})$$

$$= Q(\xi_{p_1^{\max(e_1, f_1)} \ldots\ldots\ldots p_1^{\max(e_k, f_k)}})$$

$$= Q(\xi_{[m,n]});$$

An entirely similar computation shows that $Q(\xi_m) \cap Q(\xi_n) = Q(\xi_{(m,n)})$

Mutual information measures the information transferred when x_i is sent and y_i is received, and is defined as

$$I(x_i, y_i) = \log_2 \frac{P(x_i/y_i)}{P(x_i)} \ bits \qquad (1)$$

In a noise-free channel, **each** y_i is uniquely connected to the corresponding x_i , and so they constitute an input –output pair (x_i, y_i) for which

$$P(x_i/y_j) = 1 \ and \ I(x_i, y_j) = \log_2 \frac{1}{P(x_i)} \quad \text{bits;}$$

that is, the transferred information is equal to the self-information that corresponds to the input x_i In a very noisy channel, the output y_i and input x_i would be completely uncorrelated, and so $P(x_i/y_j) = P(x_i)$ and also $I(x_i, y_j) = 0;$ that is, there is no transference of information. In general, a given channel will operate between these two extremes. The mutual information is defined between the input and the output of a given channel. An average of the calculation of the mutual information for all input-output pairs of a given channel is the average mutual information:

$$I(X,Y)=\sum_{i.j}P(x_i,y_j)I(x_i,y_j)=\sum_{i.j}P(x_i,y_j)\log_2\left[\frac{P({}^{x_i}\!/_{y_j})}{P(x_i)}\right]$$

bits per symbol . This calculation is done over the input and output alphabets. The average mutual information. The following expressions are useful for modifying the mutual information expression:

$$P(x_i,y_j)=P({}^{x_i}\!/_{y_j})P(y_j)=P({}^{y_j}\!/_{x_i})P(x_i)$$

$$P(y_j)=\sum_i P({}^{y_j}\!/_{x_i})P(x_i)$$

$$P(x_i)=\sum_i P({}^{x_i}\!/_{y_j})P(y_j)$$

Then

$$I(X,Y)=\sum_{i.j}P(x_i,y_j)$$

$$=\sum_{i.j}P(x_i,y_j)\log_2\left[\frac{1}{P(x_i)}\right]$$

$$-\sum_{i.j}P(x_i,y_j)\log_2\left[\frac{1}{P({}^{x_i}\!/_{y_j})}\right]$$

$$\sum_{i.j}P(x_i,y_j)\log_2\left[\frac{1}{P(x_i)}\right]$$

$$=\sum_i\left[P({}^{x_i}\!/_{y_j})P(y_j)\right]\log_2\frac{1}{P(x_i)}$$

$$\sum_i P(x_i)\log_2\frac{1}{P(x_i)}=H(X)$$

$$I(X,Y)=H(X)-H({}^{X}\!/_{Y})$$

Where $H({}^{X}\!/_{Y})=\sum_{i,j}P(x_i,y_j)\log_2\dfrac{1}{P({}^{x_i}\!/_{y_j})}$

is usually called the equivocation. In a sense, the equivocation can be seen as the information lost in the noisy channel, and is a function of the backward conditional probability. The observation of an output symbol y_j provides $H(X)-H({}^{X}\!/_{Y})$ bits of information. This difference is the mutual information of the channel. *Mutual Information: Properties* Since

$$P({}^{x_i}\!/_{y_j})P(y_j)=P({}^{y_j}\!/_{x_i})P(x_i)$$

The mutual information fits the condition

$$I(X,Y)=I(Y,X)$$

And by interchanging input and output it is also true that

$$I(X,Y)=H(Y)-H({}^{Y}\!/_{X})$$

Where

$$H(Y)=\sum_j P(y_j)\log_2\frac{1}{P(y_j)}$$

This last entropy is usually called the noise entropy. Thus, the information transferred through the channel is the difference between the output entropy and the noise entropy. Alternatively, it can be said that the channel mutual information is the difference between the number of bits needed for determining a given input symbol before knowing the corresponding output symbol, and the number of bits needed for determining a given input symbol after knowing the corresponding output symbol

$$I(X,Y)=H(X)-H({}^{X}\!/_{Y})$$

As the channel mutual information expression is a difference between two quantities, it seems that this parameter can adopt negative values. However, and is spite of the fact that for some $y_j, H(X/y_j)$ can be larger than $H(X)$, this is not possible for the average value calculated over all the outputs:

$$\sum_{i,j}P(x_i,y_j)\log_2\frac{P({}^{x_i}\!/_{y_j})}{P(x_i)}=\sum_{i,j}P(x_i,y_j)\log_2\frac{P(x_i,y_j)}{P(x_i)P(y_j)}$$

Then

$$-I(X,Y)=\sum_{i,j}P(x_i,y_j)\frac{P(x_i)P(y_j)}{P(x_i,y_j)}\leq 0$$

Because this expression is of the form

$$\sum_{i=1}^{M}P_i\log_2\left(\frac{Q_i}{P_i}\right)\leq 0$$

The above expression can be applied due to the factor $P(x_i)P(y_j)$, which is the product of two probabilities, so that it behaves as the quantity Q_i, which in this expression is a dummy variable that fits the condition $\sum_i Q_i \leq 1$. It can be concluded that the average mutual information is a non-negative number. It can also be equal to zero, when the input and the output are independent of each other. A related entropy called the joint entropy is defined as

$$H(X,Y)=\sum_{i,j}P(x_i,y_j)\log_2\frac{1}{P(x_i,y_j)}$$
$$=\sum_{i,j}P(x_i,y_j)\log_2\frac{P(x_i)P(y_j)}{P(x_i,y_j)}$$
$$+\sum_{i,j}P(x_i,y_j)\log_2\frac{1}{P(x_i)P(y_j)}$$

Theorem 1.5: Entropies of the binary erasure channel (BEC) The BEC is defined with an alphabet of two inputs and three outputs, with symbol probabilities.

$P(x_1)=\alpha$ *and* $P(x_2)=1-\alpha,$ and transition probabilities

$$P({}^{y_3}\!/\!{}_{x_2})=1-p \;\; and \;\; P({}^{y_2}\!/\!{}_{x_1})=0,$$
$$and \;\; P({}^{y_3}\!/\!{}_{x_1})=0$$
$$and \;\; P({}^{y_1}\!/\!{}_{x_2})=p$$
$$and \;\; P({}^{y_3}\!/\!{}_{x_2})=1-p$$

Lemma 1.7. Given an arbitrary restricted time-discrete, amplitude-continuous channel whose restrictions are determined by sets F_n and whose density functions exhibit no dependence on the state s, let n be a fixed positive integer, and $p(x)$ an arbitrary probability density function on Euclidean *n*-space. $p(y|x)$ for the density $p_n(y_1,...,y_n|x_1,...x_n)$ and $F\;\; for\;\; F_n$. For any real number a, let

$$A=\left\{(x,y):\log\frac{p(y|x)}{p(y)}>a\right\} \qquad (1)$$

Then for each positive integer u , there is a code (u,n,λ) such that

$$\lambda\le ue^{-a}+P\{(X,Y)\notin A\}+P\{X\notin F\} \qquad (2)$$

Where

Proof: A sequence $x^{(1)}\in F$ *such that*

$$P\{Y\in A_{x^1}\mid X=x^{(1)}\}\ge 1-\varepsilon$$

where $A_x=\{y:(x,y)\varepsilon A\};$

Choose the decoding set B_1 to be $A_{x^{(1)}}$. Having chosen $x^{(1)},........,x^{(k-1)}$ and $B_1,...,B_{k-1}$, select $x^k\in F$ such that

$$P\left\{Y\in A_{x^{(k)}}-\bigcup_{i=1}^{k-1}B_i\mid X=x^{(k)}\right\}\ge 1-\varepsilon;$$

Set $B_k=A_{x^{(k)}}-\bigcup_{i=1}^{k-1}B_i$, If the process does not terminate in a finite number of steps, then the sequences $x^{(i)}$ and decoding sets $B_i,\ i=1,2,...,u,$ form the desired code. Thus assume that the process terminates after t steps. (Conceivably $t=0$). We will show $t\ge u$ by showing that $\varepsilon\le te^{-a}+P\{(X,Y)\notin A\}+P\{X\notin F\}$. We proceed as follows.

Let

$$B=\bigcup_{j=1}^{t}B_j.\quad (If\;\; t=0,\;\; take\;\; B=\phi).\;\; Then$$

$$P\{(X,Y)\in A\}=\int_{(x,y)\in A}p(x,y)dx\,dy$$
$$=\int_x p(x)\int_{y\in A_x}p(y|x)dy\,dx$$
$$=\int_x p(x)\int_{y\in B\cap A_x}p(y|x)dy\,dx+\int_x p(x)$$

Algorithms

Ideals. Let A be a ring. Recall that an *ideal a* in A is a subset such that a is subgroup of A regarded as a group under addition;

$$a\in a, r\in A\Rightarrow ra\in A$$

The ideal generated by a subset S of A is the intersection of all ideals A containing a ----- it is easy to verify that this is in fact an ideal, and that it consist of all finite sums of the form $\sum r_i s_i$ with $r_i\in A, s_i\in S$. When $S=\{s_1,.....,s_m\}$, we shall write $(s_1,.....,s_m)$ for the ideal it generates.

Let a and b be ideals in A. The set $\{a+b\mid a\in a, b\in b\}$ is an ideal, denoted by $a+b$. The ideal generated by $\{ab\mid a\in a, b\in b\}$ is denoted by ab . Note that $ab\subset a\cap b$. Clearly ab consists of all finite sums $\sum a_i b_i$ with $a_i\in a$

$$P\{(X,Y)\in A\}=\int_A\cdots\int p(x,y)dxdy, \qquad p(x,y)=p(x)p(y|x)$$

and

$$P\{X\in F\}=\int_F\cdots\int p(x)dx$$

and $b_i\in b$, and if $a=(a_1,...,a_m)$ and $b=(b_1,...,b_n)$, then $ab=(a_1b_1,...,a_ib_j,...,a_mb_n)$ .Let a be an ideal of A. The set of cosets of a in A forms a ring A/a , and $a\mapsto a+a$ is a homomorphism $\phi: A\mapsto A/a$. The map $b\mapsto\phi^{-1}(b)$ is a one to one correspondence between the ideals of A/a and the ideals of A containing a An ideal p if *prime* if $p\neq A$ and $ab\in p\Rightarrow a\in p$ or $b\in p$. Thus p is prime if and only if A/p is nonzero and has the property that $ab=0,\quad b\neq 0\Rightarrow a=0,$ i.e.,

A/p is an integral domain. An ideal m is *maximal* if $m \neq| A$ and there does not exist an ideal n contained strictly between m and A. Thus m is maximal if and only if A/m has no proper nonzero ideals, and so is a field. Note that m maximal $\Rightarrow$ m prime. The ideals of $A\times B$ are all of the form $a\times b$, with a and b ideals in A and B. To see this, note that if c is an ideal in $A\times B$ and $(a,b)\in c$, then $(a,0)=(a,b)(1,0)\in c$ and $(0,b)=(a,b)(0,1)\in c$. This shows that $c=a\times b$ with

$$a=\{a\,|\,(a,b)\in c \;\; some \;\; b\in b\}$$

and

$$b=\{b\,|\,(a,b)\in c \;\; some \;\; a\in a\}$$

Let A be a ring. An A-algebra is a ring B together with a homomorphism $i_B : A\to B$. A *homomorphism of* A -algebra $B\to C$ is a homomorphism of rings $\varphi : B\to C$ such that $\varphi(i_B(a))=i_C(a)$ for all $a\in A$. An A-algebra B is said to be *finitely generated* (or of *finite-type* over A) if there exist elements $x_1,...,x_n\in B$ such that every element of B can be expressed as a polynomial in the x_i with coefficients in $i(A)$, i.e., such that the homomorphism $A[X_1,...,X_n]\to B$ sending X_i to x_i is surjective. A ring homomorphism $A\to B$ is *finite*, and B is finitely generated as an A-module. Let k be a field, and let A be a k-algebra. If $1\neq 0$ in A, then the map $k\to A$ is injective, we can identify k with its image, i.e., we can regard k as a subring of A. If 1=0 in a ring R, the R is the zero ring, i.e., $R=\{0\}$.

Polynomial rings. Let k be a field. A *monomial* in $X_1,...,X_n$ is an expression of the form $X_1^{a_1}...X_n^{a_n}$, $a_j\in N$. The *total degree* of the monomial is $\sum a_i$. We sometimes abbreviate it by X^{α}, $\alpha=(a_1,...,a_n)\in\square^n$. The elements of the polynomial ring $k[X_1,...,X_n]$ are finite sums $\sum c_{a_1....a_n}X_1^{a_1}...X_n^{a_n}$, $c_{a_1....a_n}\in k$, $a_j\in\square$ With the obvious notions of equality, addition and multiplication. Thus the monomials from basis for $k[X_1,...,X_n]$ as a k-vector space. The ring $k[X_1,...,X_n]$ is an integral domain, and the only units in it are the nonzero constant polynomials. A polynomial $f(X_1,...,X_n)$ is *irreducible* if it is nonconstant and has only the obvious factorizations, i.e., $f=gh\Rightarrow g$ or h is constant. **Division in** $k[X]$. The division algorithm allows us to divide a nonzero polynomial into another: let f and g be polynomials in $k[X]$ with $g\neq 0$; then there exist unique polynomials $q,r\in k[X]$ such that $f=qg+r$ with either $r=0$ or deg r < deg g. Moreover, there is an algorithm for deciding whether $f\in(g)$, namely, find r and check whether it is zero. Moreover, the Euclidean algorithm allows to pass from finite set of generators for an ideal in $k[X]$ to a single generator by successively replacing each pair of generators with their greatest common divisor.

(Pure) lexicographic ordering (lex). Here monomials are ordered by lexicographic(dictionary) order. More precisely, let $\alpha=(a_1,...a_n)$ and $\beta=(b_1,...b_n)$ be two elements of $\square^n$; then $\alpha>\beta$ *and* $X^{\alpha}>X^{\beta}$ (lexicographic ordering) if, in the vector difference $\alpha-\beta\in\square$, the left most nonzero entry is positive. For example, $XY^2>Y^3Z^4$; $X^3Y^2Z^4>X^3Y^2Z$. Note that this isn't quite how the dictionary would order them: it would put *XXXYYZZZZ* after *XXXYYZ*. *Graded reverse lexicographic order (grevlex).* Here monomials are ordered by total degree, with ties broken by reverse lexicographic ordering. Thus, $\alpha>\beta$ if $\sum a_i>\sum b_i$, or $\sum a_i=\sum b_i$ and in $\alpha-\beta$ the right most nonzero entry is negative. For example:

$X^4Y^4Z^7>X^5Y^5Z^4$ *(total degree greater)*

$XY^5Z^2>X^4YZ^3$, $X^5YZ>X^4YZ^2$.

Orderings on $k[X_1,...X_n]$**.** Fix an ordering on the monomials in $k[X_1,...X_n]$. Then we can write an element f of $k[X_1,...X_n]$ in a canonical fashion, by re-ordering its elements in decreasing order. For example, we would write

$$f=4XY^2Z+4Z^2-5X^3+7X^2Z^2$$

as

$$f=-5X^3+7X^2Z^2+4XY^2Z+4Z^2 \quad (lex)$$

or

$f = 4XY^2Z + 7X^2Z^2 - 5X^3 + 4Z^2 \ \ (grevlex)$

Let $\sum a_\alpha X^\alpha \in k[X_1,..., X_n]$, in decreasing order:

$f = a_{\alpha_0} X^{\alpha_0} +_{\alpha_1} X^{\alpha_1} + ..., \qquad \alpha_0 > \alpha_1 > ..., \quad \alpha_0 \neq 0$

Then we define.

- The *multidegree* of f to be multdeg(f)= α_0 ;
- The *leading coefficient of* f to be *LC(* f *)*= a_{α_0} ;
- The *leading monomial of* f to be *LM(* f *)* = X^{α_0} ;
- The *leading term of* f to be *LT(* f *)* = $a_{\alpha_0} X^{\alpha_0}$

For the polynomial $f = 4XY^2Z + ...,$ the multidegree is (1,2,1), the leading coefficient is 4, the leading monomial is XY^2Z , and the leading term is $4XY^2Z$. **The division algorithm in** $k[X_1,...X_n]$. Fix a monomial ordering in □ ^{2} . Suppose given a polynomial f and an ordered set $(g_1,...g_s)$ of polynomials; the division algorithm then constructs polynomials $a_1,...a_s$ and r such that $f = a_1g_1 + ... + a_sg_s + r$ Where either $r = 0$ or no monomial in r is divisible by any of $LT(g_1),..., LT(g_s)$ **Step 1:** If $LT(g_1) | LT(f)$, divide g_1 into f to get $f = a_1g_1 + h, \quad a_1 = \frac{LT(f)}{LT(g_1)} \in k[X_1,..., X_n]$

If $LT(g_1) | LT(h)$, repeat the process until $f = a_1g_1 + f_1$ (different a_1) with $LT(f_1)$ not divisible by $LT(g_1)$. Now divide g_2 into f_1, and so on, until $f = a_1g_1 + ... + a_sg_s + r_1$ With $LT(r_1)$ not divisible by any $LT(g_1),...LT(g_s)$
Step 2: Rewrite $r_1 = LT(r_1) + r_2$, and repeat Step 1 with r_2 for f : $f = a_1g_1 + ... + a_sg_s + LT(r_1) + r_3$ (different a_i 's) **Monomial ideals.** In general, an ideal a will contain a polynomial without containing the individual terms of the polynomial; for example, the ideal $a = (Y^2 - X^3)$ contains $Y^2 - X^3$ but not Y^2 or X^3 .

DEFINITION 1.5. An ideal a is *monomial* if $\sum c_\alpha X^\alpha \in a \Rightarrow X^\alpha \in a$ all α with $c_\alpha \neq 0$.

PROPOSITION 1.3. Let a be a *monomial ideal,* and let $A = \{\alpha \mid X^\alpha \in a\}$. Then A satisfies the condition $\alpha \in A, \ \ \beta \in$ □ $^n \Rightarrow \alpha + \beta \in$ $(*)$ And a is the k -subspace of $k[X_1,..., X_n]$ generated by the $X^\alpha, \alpha \in A$. Conversely, of A is a subset of □ ^{n} satisfying $(*)$, then the k-subspace a of $k[X_1,..., X_n]$ generated by $\{X^\alpha \mid \alpha \in A\}$ is a monomial ideal.

PROOF. It is clear from its definition that a monomial ideal a is the k -subspace of $k[X_1,..., X_n]$ generated by the set of monomials it contains. If $X^\alpha \in a$ and $X^\beta \in k[X_1,..., X_n]$.

If a permutation is chosen uniformly and at random from the $n!$ possible permutations in S_n, then the counts $C_j^{(n)}$ of cycles of length j are dependent random variables. The joint distribution of $C^{(n)} = (C_1^{(n)},..., C_n^{(n)})$ follows from Cauchy's formula, and is given by

$$P[C^{(n)} = c] = \frac{1}{n!} N(n,c) = 1\left\{\sum_{j=1}^{n} jc_j = n\right\} \prod_{j=1}^{n} (\frac{1}{j})^{c_j} \frac{1}{c_j!}, \qquad (1.1)$$

for $c \in$ □ $^n_+$.

Lemma1.7 For nonnegative integers $m_{1,...,} m_n$,

$$E\left(\prod_{j=1}^{n} (C_j^{(n)})^{[m_j]}\right) = \left(\prod_{j=1}^{n} \left(\frac{1}{j}\right)^{m_j}\right) 1\left\{\sum_{j=1}^{n} jm_j \leq n\right\} \qquad (1.4)$$

Proof. This can be established directly by exploiting cancellation of the form $c_j^{[m_j]} / c_j^! = 1/(c_j - m_j)!$ when $c_j \geq m_j$, which occurs between the ingredients in Cauchy's formula and the falling factorials in the moments. Write $m = \sum jm_j$. Then, with the first sum indexed by $c = (c_1,...c_n) \in$ □ $^n_+$ and the last sum indexed by

$d=(d_1,\dots,d_n)\in\square^n_+$ via the correspondence $d_j=c_j-m_j$, we have

$$E\left(\prod_{j=1}^{n}(C_j^{(n)})^{[m_j]}\right)=\sum_c P[C^{(n)}=c]\prod_{j=1}^{n}(c_j)^{[m_j]}$$

$$=\sum_{c:c_j\ge m_j\ \ for\ all\ j}1\left\{\sum_{j=1}^{n}jc_j=n\right\}\prod_{j=1}^{n}\frac{(c_j)^{[m_j]}}{j^{c_j}c_j!}$$

$$=\prod_{j=1}^{n}\frac{1}{j^{m_j}}\sum_d 1\left\{\sum_{j=1}^{n}jd_j=n-m\right\}\prod_{j=1}^{n}\frac{1}{j^{d_j}(d_j)!}$$

This last sum simplifies to the indicator $1(m\le n)$, corresponding to the fact that if $n-m\ge 0$, then $d_j=0$ for $j>n-m$, and a random permutation in S_{n-m} must have some cycle structure $(d_1,\dots,d_{n-m})$. The moments of $C_j^{(n)}$ follow immediately as

$$E(C_j^{(n)})^{[r]}=j^{-r}1\{jr\le n\} \qquad (1.2)$$

We note for future reference that (1.4) can also be written in the form

$$E\left(\prod_{j=1}^{n}(C_j^{(n)})^{[m_j]}\right)=E\left(\prod_{j=1}^{n}Z_j^{[m_j]}\right)1\left\{\sum_{j=1}^{n}jm_j\le n\right\}, \qquad (1.3)$$

Where the Z_j are independent Poisson-distribution random variables that satisfy $E(Z_j)=1/j$

The marginal distribution of cycle counts provides a formula for the joint distribution of the cycle counts C_j^n, we find the distribution of C_j^n using a combinatorial approach combined with the inclusion-exclusion formula.

Lemma 1.8. For $1\le j\le n$,

$$P[C_j^{(n)}=k]=\frac{j^{-k}}{k!}\sum_{l=0}^{[n/j]-k}(-1)^l\frac{j^{-l}}{l!} \qquad (1.1)$$

Proof. Consider the set I of all possible cycles of length j, formed with elements chosen from $\{1,2,\dots n\}$, so that $|I|=n^{[j]/j}$. For each $\alpha\in I$, consider the "property" G_α of having α; that is, G_α is the set of permutations $\pi\in S_n$ such that α is one of the cycles of π. We then have $|G_\alpha|=(n-j)!$, since the elements of $\{1,2,\dots,n\}$ not in α must be permuted among themselves. To use the inclusion-exclusion formula we need to calculate the term S_r, which is the sum of the probabilities of the r-fold intersection of properties, summing over all sets of r distinct properties. There are two cases to consider. If the r properties are indexed by r cycles having no elements in common, then the intersection specifies how rj elements are moved by the permutation, and there are $(n-rj)!1(rj\le n)$ permutations in the intersection. There are $n^{[rj]}/(j^r r!)$ such intersections. For the other case, some two distinct properties name some element in common, so no permutation can have both these properties, and the r-fold intersection is empty. Thus

$$S_r=(n-rj)!1(rj\le n)$$

$$\times\frac{n^{[rj]}}{j^r r!}\frac{1}{n!}=1(rj\le n)\frac{1}{j^r r!}$$

Finally, the inclusion-exclusion series for the number of permutations having exactly k properties is

$$\sum_{l\ge 0}(-1)^l\binom{k+l}{l}S_{k+l},$$

Which simplifies to (1.1) Returning to the original hat-check problem, we substitute j=1 in (1.1) to obtain the distribution of the number of fixed points of a random permutation. For $k=0,1,\dots,n,$

$$P[C_1^{(n)}=k]=\frac{1}{k!}\sum_{l=0}^{n-k}(-1)^l\frac{1}{l!}, \qquad (1.2)$$

and the moments of $C_1^{(n)}$ follow from (1.2) with $j=1$. In particular, for $n\ge 2$, the mean and variance of $C_1^{(n)}$ are both equal to 1. The joint distribution of $(C_1^{(n)},\dots,C_b^{(n)})$ for any $1\le b\le n$ has an expression similar to (1.7); this too can be derived by inclusion-exclusion. For any $c=(c_1,\dots,c_b)\in\square^b_+$ with $m=\sum ic_i$,

$$P[(C_1^{(n)},\dots,C_b^{(n)})=c]$$

$$=\left\{\prod_{i=1}^{b}\left(\frac{1}{i}\right)^{c_i}\frac{1}{c_i!}\right\}\sum_{\substack{l\ge 0\ with\\ \sum il_i\le n-m}}(-1)^{l_1+\dots+l_b}\prod_{i=1}^{b}\left(\frac{1}{i}\right)^{l_i}\frac{1}{l_i!} \qquad (1.3)$$

The joint moments of the first b counts $C_1^{(n)},\dots,C_b^{(n)}$ can be obtained directly from (1.2) and (1.3) by setting $m_{b+1}=\dots=m_n=0$

The limit distribution of cycle counts

It follows immediately from Lemma 1.2 that for each fixed j, as $n\to\infty$,

$$P[C_j^{(n)}=k]\to\frac{j^{-k}}{k!}e^{-1/j},\quad k=0,1,2,\dots,$$

So that $C_j^{(n)}$ converges in distribution to a random variable Z_j having a Poisson distribution with mean $1/j;$ we use the notation $C_j^{(n)} \rightarrow_d Z_j$ where $Z_j \,\square\, P_o(1/j)$ to describe this. Infact, the limit random variables are independent.

Theorem 1.6 The process of cycle counts converges in distribution to a Poisson process of □ with intensity j^{-1}. That is, as $n \to \infty$,

$$(C_1^{(n)}, C_2^{(n)}, \ldots) \rightarrow_d (Z_1, Z_2, \ldots) \qquad (1.1)$$

Where the $Z_j, j = 1, 2, \ldots,$ are independent Poisson-distributed random variables with $E(Z_j) = \frac{1}{j}$

Proof. To establish the converges in distribution one shows that for each fixed $b \ge 1$, as $n \to \infty$,

$$P[(C_1^{(n)}, \ldots, C_b^{(n)}) = c] \to P[(Z_1, \ldots, Z_b) = c]$$

Error rates

The proof of Theorem says nothing about the rate of convergence. Elementary analysis can be used to estimate this rate when $b = 1$. Using properties of alternating series with decreasing terms, for $k = 0, 1, \ldots, n,$

$$\frac{1}{k!}\left(\frac{1}{(n-k+1)!} - \frac{1}{(n-k+2)!}\right) \le \left|P[C_1^{(n)} = k] - P[Z_1 = k]\right|$$

$$\le \frac{1}{k!(n-k+1)!}$$

It follows that

$$\frac{2^{n+1}}{(n+1)!}\frac{n}{n+2} \le \sum_{k=0}^{n} \left|P[C_1^{(n)} = k] - P[Z_1 = k]\right| \le \frac{2^{n+1}-1}{(n+1)!} \qquad (1.11)$$

Since

$$P[Z_1 > n] = \frac{e^{-1}}{(n+1)!}\left(1 + \frac{1}{n+2} + \frac{1}{(n+2)(n+3)} + \ldots\right) < \frac{1}{(n+1)!},$$

We see from (1.11) that the total variation distance between the distribution $L(C_1^{(n)})$ of $C_1^{(n)}$ and the distribution $L(Z_1)$ of Z_1

Establish the asymptotics of $\mathrm{P}\left[A_n(C^{(n)})\right]$ under conditions (A_0) and (B_{01}), where

$$A_n(C^{(n)}) = \bigcap_{1 \le i \le n} \ \bigcap_{r_i' + 1 \le j \le r_i} \left\{C_{ij}^{(n)} = 0\right\},$$

and $\zeta_i = (r_i' / r_{id}) - 1 = O(i^{-g'})$ as $i \to \infty$, for some $g' > 0.$ We start with the expression

$$P[A_n(C^{(n)})] = \frac{P[T_{0m}(Z') = n]}{P[T_{0m}(Z) = n]}$$

$$\prod_{\substack{1 \le i \le n \\ r_i' + 1 \le j \le r_i}} \left\{1 - \frac{\theta}{ir_i}(1 + E_{i0})\right\} \qquad (1.1)$$

$$P[T_{0n}(Z') = n]$$

$$= \frac{\theta d}{n} \exp\left\{\sum_{i \ge 1} [\log(1 + i^{-1}\theta d) - i^{-1}\theta d]\right\}$$

$$\left\{1 + O(n^{-1}\varphi'_{\{1,2,7\}}(n))\right\} \qquad (1.2)$$

and

$$P[T_{0n}(Z') = n]$$

$$= \frac{\theta d}{n} \exp\left\{\sum_{i \ge 1} [\log(1 + i^{-1}\theta d) - i^{-1}\theta d]\right\}$$

$$\left\{1 + O(n^{-1}\varphi_{\{1,2,7\}}(n))\right\} \qquad (1.3)$$

Where $\varphi'_{\{1,2,7\}}(n)$ refers to the quantity derived from Z'. It thus follows that $P[A_n(C^{(n)})] \,\square\, Kn^{-\theta(1-d)}$ for a constant K, depending on Z and the r_i' and computable explicitly from (1.1) – (1.3), if Conditions (A_0) and (B_{01}) are satisfied and if $\zeta_i^* = O(i^{-g'})$ from some $g' > 0,$ since, under these circumstances, both $n^{-1}\varphi'_{\{1,2,7\}}(n)$ and $n^{-1}\varphi_{\{1,2,7\}}(n)$ tend to zero as $n \to \infty$. In particular, for polynomials and square free polynomials, the relative error in this asymptotic approximation is of order n^{-1} if $g' > 1.$

For $0 \le b \le n/8$ and $n \ge n_0,$ with n_0

$$d_{TV}(L(C[1,b]), L(Z[1,b]))$$

$$\le d_{TV}(L(\overset{\square}{C}[1,b]), L(\overset{\square}{Z}[1,b]))$$

$$\le \varepsilon_{\{7,7\}}(n,b),$$

Where $\varepsilon_{\{7,7\}}(n,b) = O(b/n)$ under Conditions $(A_0), (D_1)$ and (B_{11}) Since, by the Conditioning Relation,

$$L(\overset{\square}{C}[1,b] \,|\, T_{0b}(C) = l) = L(\overset{\square}{Z}[1,b] \,|\, T_{0b}(Z) = l),$$

It follows by direct calculation that

$$d_{TV}(L(\overset{\square}{C}[1,b]),L(\overset{\square}{Z}[1,b]))$$
$$=d_{TV}(L(T_{0b}(C)),L(T_{0b}(Z)))$$
$$=\max_{A}\sum_{r\in A}P[T_{0b}(Z)=r]$$
$$\left\{1-\frac{P[T_{bn}(Z)=n-r]}{P[T_{0n}(Z)=n]}\right\} \qquad (1.4)$$

Suppressing the argument Z from now on, we thus obtain

$$d_{TV}(L(\overset{\square}{C}[1,b]),L(\overset{\square}{Z}[1,b]))$$
$$=\sum_{r\geq 0}P[T_{0b}=r]\left\{1-\frac{P[T_{bn}=n-r]}{P[T_{0n}=n]}\right\}_{+}$$
$$\leq\sum_{r>n/2}P[T_{0b}=r]+\sum_{r=0}^{[n/2]}\frac{P[T_{0b}=r]}{P[T_{0b}=n]}$$
$$\times\left\{\sum_{s=0}^{n}P[T_{0b}=s](P[T_{bn}=n-s]-P[T_{bn}=n-r]\right\}_{+}$$

$$\leq\sum_{r>n/2}P[T_{0b}=r]+\sum_{r=0}^{[n/2]}P[T_{0b}=r]$$
$$\times\sum_{s=0}^{[n/2]}P[T_{0b}=s]\frac{\{P[T_{bn}=n-s]-P[T_{bn}=n-r]\}}{P[T_{0n}=n]}$$
$$+\sum_{s=0}^{[n/2]}P[T_{0b}=r]\sum_{s=[n/2]+1}^{n}P[T=s]P[T_{bn}=n-s]/P[T_{0n}=n]$$

The first sum is at most $2n^{-1}ET_{0b}$; the third is bound by

$$(\max_{n/2<s\leq n}P[T_{0b}=s])/P[T_{0n}=n]$$
$$\leq\frac{2\varepsilon_{\{10.5(1)\}}(n/2,b)}{n}\frac{3n}{\theta P_{\theta}[0,1]},$$
$$\frac{3n}{\theta P_{\theta}[0,1]}4n^{-2}\phi^{*}_{\{10.8\}}(n)\sum_{r=0}^{[n/2]}P[T_{0b}=r]\sum_{s=0}^{[n/2]}P[T_{0b}=s]\frac{1}{2}|r-s|$$
$$\leq\frac{12\phi^{*}_{\{10.8\}}(n)}{\theta P_{\theta}[0,1]}\frac{ET_{0b}}{n}$$

Hence we may take

$$\varepsilon_{\{7,7\}}(n,b)=2n^{-1}ET_{0b}(Z)\left\{1+\frac{6\phi^{*}_{\{10.8\}}(n)}{\theta P_{\theta}[0,1]}\right\}P$$

$$+\frac{6}{\theta P_{\theta}[0,1]}\varepsilon_{\{10.5(1)\}}(n/2,b) \qquad (1.5)$$

Required order under Conditions $(A_0),(D_1)$ and (B_{11}), if $S(\infty)<\infty$. If not, $\phi^{*}_{\{10.8\}}(n)$ can be replaced by $\phi^{*}_{\{10.11\}}(n)$ in the above, which has the required order, without the restriction on the r_i implied by $S(\infty)<\infty$. Examining the Conditions $(A_0),(D_1)$ and (B_{11}), it is perhaps surprising to find that (B_{11}) is required instead of just (B_{01}); that is, that we should need $\sum_{l\geq 2}l\varepsilon_{il}=O(i^{-a_1})$ to hold for some $a_1>1$. A first observation is that a similar problem arises with the rate of decay of ε_{i1} as well. For this reason, n_1 is replaced by $\overset{\square}{n}_1$. This makes it possible to replace condition (A_1) by the weaker pair of conditions (A_0) and (D_1) in the eventual assumptions needed for $\varepsilon_{\{7,7\}}(n,b)$ to be of order $O(b/n)$; the decay rate requirement of order $i^{-1-\gamma}$ is shifted from ε_{i1} itself to its first difference. This is needed to obtain the right approximation error for the random mappings example. However, since all the classical applications make far more stringent assumptions about the $\varepsilon_{i1},l\geq 2,$ than are made in (B_{11}). The critical point of the proof is seen where the initial estimate of the difference $P[T^{(m)}_{bn}=s]-P[T^{(m)}_{bn}=s+1]$. The factor $\varepsilon_{\{10.10\}}(n),$ which should be small, contains a far tail element from of the form $\phi^{\theta}_{1}(n)+u^{*}_{1}(n),$ which is only small if $a_1>1,$ being otherwise of order $O(n^{1-a_1+\delta})$ for any $\delta>0,$ since $a_2>1$ is in any case assumed. For $s\geq n/2,$ this gives rise to a contribution of order $O(n^{-1-a_1+\delta})$ in the estimate of the difference $P[T_{bn}=s]-P[T_{bn}=s+1],$ which, in the remainder of the proof, is translated into a contribution of order $O(tn^{-1-a_1+\delta})$ for differences of the form $P[T_{bn}=s]-P[T_{bn}=s+1],$ finally leading to a contribution of order $bn^{-a_1+\delta}$ for any $\delta>0$ in $\varepsilon_{\{7.7\}}(n,b).$ Some improvement would seem to be possible, defining the function g by $g(w)=1_{\{w=s\}}-1_{\{w=s+t\}},$ differences that are of the form $P[T_{bn}=s]-P[T_{bn}=s+t]$ can be directly estimated, at a cost of only a single contribution of the form $\phi^{\theta}_{1}(n)+u^{*}_{1}(n).$ Then, iterating the cycle, in which one estimate of a difference in point probabilities is improved to an estimate of smaller order, a bound of the form

$$\left|P[T_{bn}=s]-P[T_{bn}=s+t]\right|=O(n^{-2}t+n^{-1-a_1+\delta})$$

for any $\delta>0$ could perhaps be attained, leading to a final error estimate in order $O(bn^{-1}+n^{-a_1+\delta})$ for any $\delta>0$, to replace $\varepsilon_{\{7.7\}}(n,b)$. This would be of the ideal order $O(b/n)$ for large enough b, but would still be coarser for small b.

With b and n as in the previous section, we wish to show that

$$\left|d_{TV}(L(C[1,b]),L(Z[1,b]))-\frac{1}{2}(n+1)^{-1}\left|1-\theta\right|E\left|T_{0b}-ET_{0b}\right|\right|$$

$$\le\varepsilon_{\{7,8\}}(n,b),$$

Where

$\varepsilon_{\{7.8\}}(n,b)=O(n^{-1}b[n^{-1}b+n^{-\beta_{12}+\delta}])$ for any $\delta>0$ under Conditions $(A_0),(D_1)$ and (B_{12}), with β_{12}. The proof uses sharper estimates. As before, we begin with the formula

$$d_{TV}(L(\overset{\square}{C}[1,b]),L(\overset{\square}{Z}[1,b]))$$

$$=\sum_{r\ge0}P[T_{0b}=r]\left\{1-\frac{P[T_{bn}=n-r]}{P[T_{0n}=n]}\right\}_+$$

Now we observe that

$$\left|\sum_{r\ge0}P[T_{0b}=r]\left\{1-\frac{P[T_{bn}=n-r]}{P[T_{0n}=n]}\right\}_+-\sum_{r=0}^{[n/2]}\frac{P[T_{0b}=r]}{P[T_{0n}=n]}\right|$$

$$\times\left|\sum_{s=[n/2]+1}^{n}P[T_{0b}=s](P[T_{bn}=n-s]-P[T_{bn}=n-r])\right|$$

$$\le4n^{-2}ET_{0b}^2+(\max_{n/2<s\le n}P[T_{0b}=s])/P[T_{0n}=n]$$

$$+P[T_{0b}>n/2]$$

$$\le8n^{-2}ET_{0b}^2+\frac{3\varepsilon_{\{10.5(2)\}}(n/2,b)}{\theta P_\theta[0,1]},\qquad(1.1)$$

We have

$$\Big|\ \sum_{r=0}^{[n/2]}\frac{P[T_{0b}=r]}{P[T_{0n}=n]}$$

$$\times\ (\left\{\sum_{s=0}^{[n/2]}P[T_{0b}=s](P[T_{bn}=n-s]-P[T_{bn}=n-r]\right\}_+$$

$$-\left\{\sum_{s=0}^{[n/2]}P[T_{0b}=s]\frac{(s-r)(1-\theta)}{n+1}P[T_{0n}=n]\right\}_+\ )\ \Big|$$

$$\le\frac{1}{n^2P[T_{0n}=n]}\sum_{r\ge0}P[T_{0b}=r]\sum_{s\ge0}P[T_{0b}=s]\left|s-r\right|$$

$$\times\left\{\varepsilon_{\{10.14\}}(n,b)+2(r\vee s)\left|1-\theta\right|n^{-1}\left\{K_0\theta+4\phi^*_{\{10.8\}}(n)\right\}\right\}$$

$$\le\frac{6}{\theta nP_\theta[0,1]}ET_{0b}\varepsilon_{\{10.14\}}(n,b)$$

$$+4\left|1-\theta\right|n^{-2}ET_{0b}^2\left\{K_0\theta+4\phi^*_{\{10.8\}}(n)\right\}$$

$$(\frac{3}{\theta nP_\theta[0,1]})\ \Big\},\qquad(1.2)$$

The approximation in (1.2) is further simplified by noting that

$$\sum_{r=0}^{[n/2]}P[T_{0b}=r]\left|\left\{\sum_{s=0}^{[n/2]}P[T_{0b}=s]\frac{(s-r)(1-\theta)}{n+1}\right\}_+\right.$$

$$\left.-\left\{\sum_{s=0}P[T_{0b}=s]\frac{(s-r)(1-\theta)}{n+1}\right\}_+\ \right|$$

$$\le\sum_{r=0}^{[n/2]}P[T_{0b}=r]\sum_{s>[n/2]}P[T_{0b}=s]\frac{(s-r)\left|1-\theta\right|}{n+1}$$

$$\le\left|1-\theta\right|n^{-1}E(T_{0b}1\{T_{0b}>n/2\})\le2\left|1-\theta\right|n^{-2}ET_{0b}^2,\qquad(1.3)$$

and then by observing that

$$\sum_{r>[n/2]}P[T_{0b}=r]\left\{\sum_{s\ge0}P[T_{0b}=s]\frac{(s-r)(1-\theta)}{n+1}\right\}$$

$$\le n^{-1}\left|1-\theta\right|(ET_{0b}P[T_{0b}>n/2]+E(T_{0b}1\{T_{0b}>n/2\}))$$

$$\le4\left|1-\theta\right|n^{-2}ET_{0b}^2\qquad(1.4)$$

Combining the contributions of (1.2) –(1.3), we thus find tha

$$\Big| \; d_{TV}(L(\overset{\square}{C}[1,b]), L(\overset{\square}{Z}[1,b]))$$

$$-(n+1)^{-1}\sum_{r\ge 0} P[T_{0b}=r]\left\{\sum_{s\ge 0} P[T_{0b}=s](s-r)(1-\theta)\right\}_{+} \; \Big|$$

$$\le \varepsilon_{\{7.8\}}(n,b)$$

$$=\frac{3}{\theta P_{\theta}[0,1]}\left\{\varepsilon_{\{10.5(2)\}}(n/2,b)+2n^{-1}ET_{0b}\varepsilon_{\{10.14\}}(n,b)\right\}$$

$$+2n^{-2}ET_{0b}^{2}\left\{4+3|1-\theta|+\frac{24|1-\theta|\phi^{*}_{\{10.8\}}(n)}{\theta P_{\theta}[0,1]}\right\} \qquad (1.5)$$

The quantity $\varepsilon_{\{7.8\}}(n,b)$ is seen to be of the order claimed under Conditions $(A_0),(D_1)$ and (B_{12}) , provided that $S(\infty)<\infty$; this supplementary condition can be removed if $\phi^{*}_{\{10.8\}}(n)$ is replaced by $\phi^{*}_{\{10.11\}}(n)$ in the definition of $\varepsilon_{\{7.8\}}(n,b)$, has the required order without the restriction on the r_i implied by assuming that $S(\infty)<\infty$. Finally, a direct calculation now shows that

$$\sum_{r\ge 0} P[T_{0b}=r]\left\{\sum_{s\ge 0} P[T_{0b}=s](s-r)(1-\theta)\right\}_{+}$$

$$=\frac{1}{2}|1-\theta|\,E\left|T_{0b}-ET_{0b}\right|$$

Example 1.0. Consider the point $O=(0,\ldots,0)\in\square^{n}$. For an arbitrary vector r , the coordinates of the point $x=O+r$ are equal to the respective coordinates of the vector $r: x=(x^1,\ldots x^n)$ and $r=(x^1,\ldots,x^n)$. The vector r such as in the example is called the position vector or the radius vector of the point x . (Or, in greater detail: r is the radius-vector of x w.r.t an origin O). Points are frequently specified by their radius-vectors. This presupposes the choice of O as the "standard origin". Let us summarize. We have considered $\square^{n}$ and interpreted its elements in two ways: as points and as vectors. Hence we may say that we leading with the two copies of $\square^{n}$: $\square^{n}$ = {points}, $\square^{n}$ = {vectors}

Operations with vectors: multiplication by a number, addition. Operations with points and vectors: adding a vector to a point (giving a point), subtracting two points (giving a vector). $\square^{n}$ treated in this way is called an *n-dimensional affine space. (*An "abstract" affine space is a pair of sets , the set of points and the set of vectors so that the operations as above are defined axiomatically). Notice that vectors in an affine space are also known as "free vectors". Intuitively, they are not fixed at points and "float freely" in space. From $\square^{n}$ considered as an affine space we can precede in two opposite directions: $\square^{n}$ as an Euclidean space $\Leftarrow$ $\square^{n}$ as an affine space $\Rightarrow$ $\square^{n}$ as a manifold.Going to the left means introducing some extra structure which will make the geometry richer. Going to the right means forgetting about part of the affine structure; going further in this direction will lead us to the so-called "smooth (or differentiable) manifolds". The theory of differential forms does not require any extra geometry. So our natural direction is to the right. The Euclidean structure, however, is useful for examples and applications. So let us say a few words about it:

Remark 1.0. *Euclidean geometry.* In $\square^{n}$ considered as an affine space we can already do a good deal of geometry. For example, we can consider lines and planes, and quadric surfaces like an ellipsoid. However, we cannot discuss such things as "lengths", "angles" or "areas" and "volumes". To be able to do so, we have to introduce some more definitions, making $\square^{n}$ a Euclidean space. Namely, we define the length of a vector $a=(a^1,\ldots,a^n)$ to be

$$|a|:=\sqrt{(a^1)^2+\ldots+(a^n)^2} \qquad (1)$$

After that we can also define distances between points as follows:

$$d(A,B):=\left|\overrightarrow{AB}\right| \qquad (2)$$

One can check that the distance so defined possesses natural properties that we expect: is it always non-negative and equals zero only for coinciding points; the distance from A to B is the same as that from B to A (symmetry); also, for three points, A, B and C, we have $d(A,B)\le d(A,C)+d(C,B)$ (the "triangle inequality"). To define angles, we first introduce the scalar product of two vectors

$$(a,b):=a^1b^1+\ldots+a^nb^n \qquad (3)$$

Thus $|a|=\sqrt{(a,a)}$. The scalar product is also denote by dot: $a.b=(a,b)$, and hence is often referred to as the "dot product" . Now, for nonzero vectors, we define the angle between them by the equality

$$\cos\alpha:=\frac{(a,b)}{|a||b|} \qquad (4)$$

The angle itself is defined up to an integral multiple of 2π . For this definition to be consistent we have to ensure that the r.h.s. of (4) does not exceed 1 by the absolute value. This follows from the inequality

$$(a,b)^2 \le |a|^2 |b|^2 \qquad (5)$$

known as the Cauchy–Bunyakovsky–Schwarz inequality (various combinations of these three names are applied in different books). One of the ways of proving (5) is to consider the scalar square of the linear combination $a+tb$, where $t \in R$. As $(a+tb, a+tb) \ge 0$ is a quadratic polynomial in t which is never negative, its discriminant must be less or equal zero. Writing this explicitly yields (5). The triangle inequality for distances also follows from the inequality (5).

Example 1.1. Consider the function $f(x) = x^i$ (the i-th coordinate). The linear function dx^i (the differential of x^i) applied to an arbitrary vector h is simply h^i.From these examples follows that we can rewrite df as

$$df = \frac{\partial f}{\partial x^1} dx^1 + \ldots + \frac{\partial f}{\partial x^n} dx^n, \qquad (1)$$

which is the standard form. Once again: the partial derivatives in (1) are just the coefficients (depending on x); $dx^1, dx^2, \ldots$ are linear functions giving on an arbitrary vector h its coordinates $h^1, h^2, \ldots,$ respectively. Hence

$$df(x)(h) = \partial_{hf(x)} = \frac{\partial f}{\partial x^1} h^1 + \ldots + \frac{\partial f}{\partial x^n} h^n, \qquad (2)$$

Theorem 1.7. Suppose we have a parametrized curve $t \mapsto x(t)$ passing through $x_0 \in \mathbb{R}^n$ at $t = t_0$ and with the velocity vector $x(t_0) = \upsilon$ Then

$$\frac{df(x(t))}{dt}(t_0) = \partial_\upsilon f(x_0) = df(x_0)(\upsilon) \qquad (1)$$

Proof. Indeed, consider a small increment of the parameter $t : t_0 \mapsto t_0 + \Delta t$, Where $\Delta t \mapsto 0$. On the other hand, we have $f(x_0+h) - f(x_0) = df(x_0)(h) + \beta(h)|h|$ for an arbitrary vector h , where $\beta(h) \to 0$ when $h \to 0$. Combining it together, for the increment of $f(x(t))$ we obtain

$$\begin{aligned}
&f(x(t_0+\Delta t) - f(x_0) \\
&= df(x_0)(\upsilon.\Delta t + \alpha(\Delta t)\Delta t) \\
&+ \beta(\upsilon.\Delta t + \alpha(\Delta t)\Delta t).|\upsilon\Delta t + \alpha(\Delta t)\Delta t| \\
&= df(x_0)(\upsilon).\Delta t + \gamma(\Delta t)\Delta t
\end{aligned}$$

For a certain $\gamma(\Delta t)$ such that $\gamma(\Delta t) \to 0$ when $\Delta t \to 0$ (we used the linearity of $df(x_0)$). By the definition, this means that the derivative of $f(x(t))$ at $t = t_0$ is exactly $df(x_0)(\upsilon)$. The statement of the theorem can be expressed by a simple formula:

$$\frac{df(x(t))}{dt} = \frac{\partial f}{\partial x^1} x^1 + \ldots + \frac{\partial f}{\partial x^n} x^n \qquad (2)$$

To calculate the value Of df at a point x_0 on a given vector υ one can take an arbitrary curve passing Through x_0 at t_0 with υ as the velocity vector at t_0 and calculate the usual derivative of $f(x(t))$ at $t = t_0$.

Theorem 1.8. For functions $f, g : U \to \mathbb{R}$, $U \subset \mathbb{R}^n$,

$$d(f+g) = df + dg \qquad (1)$$

$$d(fg) = df.g + f.dg \qquad (2)$$

Proof. Consider an arbitrary point x_0 and an arbitrary vector υ stretching from it. Let a curve $x(t)$ be such that $x(t_0) = x_0$ and $x(t_0) = \upsilon$. Hence

$$d(f+g)(x_0)(\upsilon) = \frac{d}{dt}(f(x(t)) + g(x(t)))$$

at $t = t_0$ and

$$d(fg)(x_0)(\upsilon) = \frac{d}{dt}(f(x(t))g(x(t)))$$

at $t = t_0$ Formulae (1) and (2) then immediately follow from the corresponding formulae for the usual derivative Now, almost without change the theory generalizes to functions taking values in $\mathbb{R}^m$ instead of $\mathbb{R}$. The only difference is that now the differential of a map $F : U \to \mathbb{R}^m$ at a point x will be a linear function taking vectors in $\mathbb{R}^n$ to vectors in $\mathbb{R}^m$ (instead of $\mathbb{R}$) . For an arbitrary vector $h \in| \mathbb{R}^n$,

$$F(x+h) = F(x) + dF(x)(h)$$

$$+\beta(h)\left|h\right| \qquad (3)$$

Where $\beta(h)\to 0$ when $h\to 0$. We have $dF=(dF^1,\ldots,dF^m)$ and

$$dF=\frac{\partial F}{\partial x^1}dx^1+\ldots+\frac{\partial F}{\partial x^n}dx^n$$

$$=\begin{pmatrix}\frac{\partial F^1}{\partial x^1}\cdots\frac{\partial F^1}{\partial x^n}\\ \ldots\quad\ldots\quad\ldots\\ \frac{\partial F^m}{\partial x^1}\cdots\frac{\partial F^m}{\partial x^n}\end{pmatrix}\begin{pmatrix}dx^1\\ \ldots\\ dx^n\end{pmatrix} \qquad (4)$$

In this matrix notation we have to write vectors as vector-columns.

Theorem 1.9. For an arbitrary parametrized curve $x(t)$ in $\mathbb{R}^n$, the differential of a map $F:U\to\mathbb{R}^m$ (where $U\subset\mathbb{R}^n$) maps the velocity vector $x(t)$ to the velocity vector of the curve $F(x(t))$ in $\mathbb{R}^m$:

$$\frac{dF(x(t))}{dt}=dF(x(t))(\dot{x}(t)) \qquad (1)$$

Proof. By the definition of the velocity vector,

$$x(t+\Delta t)=x(t)+\dot{x}(t).\Delta t+\alpha(\Delta t)\Delta t \qquad (2)$$

Where $\alpha(\Delta t)\to 0$ when $\Delta t\to 0$. By the definition of the differential,

$$F(x+h)=F(x)+dF(x)(h)+\beta(h)\left|h\right. \qquad (3)|$$

Where $\beta(h)\to 0$ when $h\to 0$. we obtain

$$F(x(t+\Delta t))=F(x+\underbrace{x(t).\Delta t+\dot{\alpha}(\Delta t)\Delta t)}_{h}$$

$$=F(x)+dF(x)(\dot{x}(t)\Delta t+\alpha(\Delta t)\Delta t)+$$

$$\beta(\dot{x}(t)\Delta t+\alpha(\Delta t)\Delta t).\left|x(t)\Delta t+\dot{\alpha}(\Delta t)\Delta t\right|$$

$$=F(x)+dF(x)(\dot{x}(t)\Delta t+\gamma(\Delta t)\Delta t$$

For some $\gamma(\Delta t)\to 0$ when $\Delta t\to 0$. This recisely means that $dF(x)\dot{x}(t)$ is the velocity vector of $F(x)$. As every vector attached to a point can be viewed as the velocity vector of some curve passing through this point, this theorem gives a clear geometric picture of dF as a linear map on vectors.

Theorem 1.10 Suppose we have two maps $F:U\to V$ and $G:V\to W$, where $U\subset\mathbb{R}^n, V\subset\mathbb{R}^m, W\subset\mathbb{R}^p$ (open domains). Let $F:x\mapsto y=F(x)$. Then the differential of the composite map $GoF:U\to W$ is the composition of the differentials of F and G:

$$d(GoF)(x)=dG(y)odF(x) \qquad (4)$$

Proof. We can use the description of the differential .Consider a curve $x(t)$ in $\mathbb{R}^n$ with the velocity vector $\dot{x}$. Basically, we need to know to which vector in $\mathbb{R}^p$ it is taken by $d(GoF)$. the curve $(GoF)(x(t)=G(F(x(t))$. By the same theorem, it equals the image under dG of the Anycast Flow vector to the curve $F(x(t))$ in $\mathbb{R}^m$. Applying the theorem once again, we see that the velocity vector to the curve $F(x(t))$ is the image under dF of the vector $\dot{x}(t)$. Hence $d(GoF)(\dot{x})=dG(dF(\dot{x}))$ for an arbitrary vector $\dot{x}$.

Corollary 1.0. If we denote coordinates in $\mathbb{R}^n$ by $(x^1,\ldots,x^n)$ and in $\mathbb{R}^m$ by $(y^1,\ldots,y^m)$, and write

$$dF=\frac{\partial F}{\partial x^1}dx^1+\ldots+\frac{\partial F}{\partial x^n}dx^n \qquad (1)$$

$$dG=\frac{\partial G}{\partial y^1}dy^1+\ldots+\frac{\partial G}{\partial y^n}dy^n, \qquad (2)$$

Then the chain rule can be expressed as follows:

$$d(GoF)=\frac{\partial G}{\partial y^1}dF^1+\ldots+\frac{\partial G}{\partial y^m}dF^m, \qquad (3)$$

Where dF^i are taken from (1). In other words, to get $d(GoF)$ we have to substitute into (2) the expression for $dy^i=dF^i$ from (3). This can also be expressed by the following matrix formula:

$$d(GoF)=\begin{pmatrix}\frac{\partial G^1}{\partial y^1}\cdots\frac{\partial G^1}{\partial y^m}\\ \ldots\quad\ldots\quad\ldots\\ \frac{\partial G^p}{\partial y^1}\cdots\frac{\partial G^p}{\partial y^m}\end{pmatrix}\begin{pmatrix}\frac{\partial F^1}{\partial x^1}\cdots\frac{\partial F^1}{\partial x^n}\\ \ldots\quad\ldots\quad\ldots\\ \frac{\partial F^m}{\partial x^1}\cdots\frac{\partial F^m}{\partial x^n}\end{pmatrix}\begin{pmatrix}dx^1\\ \ldots\\ dx^n\end{pmatrix} \qquad (4)$$

i.e., if dG and dF are expressed by matrices of partial derivatives, then $d(GoF)$ is expressed by the product of these matrices. This is often written as

$$\begin{pmatrix} \frac{\partial z^1}{\partial x^1} \cdots\cdot \frac{\partial z^1}{\partial x^n} \\ \cdots \quad \cdots \quad \cdots \\ \frac{\partial z^p}{\partial x^1} \cdots \frac{\partial z^p}{\partial x^n} \end{pmatrix} = \begin{pmatrix} \frac{\partial z^1}{\partial y^1} \cdots\cdot \frac{\partial z^1}{\partial y^m} \\ \cdots \quad \cdots \quad \cdots \\ \frac{\partial z^p}{\partial y^1} \cdots \frac{\partial z^p}{\partial y^m} \end{pmatrix} \begin{pmatrix} \frac{\partial y^1}{\partial x^1} \cdots\cdot \frac{\partial y^1}{\partial x^n} \\ \cdots \quad \cdots \quad \cdots \\ \frac{\partial y^m}{\partial x^1} \cdots \frac{\partial y^m}{\partial x^n} \end{pmatrix}, \qquad (5)$$

Or

$$\frac{\partial z^{\mu}}{\partial x^{a}} = \sum_{i=1}^{m} \frac{\partial z^{\mu}}{\partial y^{i}} \frac{\partial y^{i}}{\partial x^{a}}, \qquad (6)$$

Where it is assumed that the dependence of $y \in \mathbb{R}^m$ on $x \in \mathbb{R}^n$ is given by the map F, the dependence of $z \in \mathbb{R}^p$ on $y \in \mathbb{R}^m$ is given by the map G, and the dependence of $z \in \mathbb{R}^p$ on $x \in \mathbb{R}^n$ is given by the composition GoF.

Definition 1.6. Consider an open domain $U \subset \mathbb{R}^n$. Consider also another copy of $\mathbb{R}^n$, denoted for distinction $\mathbb{R}^n_y$, with the standard coordinates $(y^1 \ldots y^n)$. A system of coordinates in the open domain U is given by a map $F: V \to U$, where $V \subset \mathbb{R}^n_y$ is an open domain of $\mathbb{R}^n_y$, such that the following three conditions are satisfied :

(1) F is smooth;

(2) F is invertible;

(3) $F^{-1}: U \to V$ is also smooth

The coordinates of a point $x \in U$ in this system are the standard coordinates of $F^{-1}(x) \in \mathbb{R}^n_y$

In other words,

$$F: (y^1 \ldots, y^n) \mapsto x = x(y^1 \ldots, y^n) \qquad (1)$$

Here the variables $(y^1 \ldots, y^n)$ are the "new" coordinates of the point x

Example 1.2. Consider a curve in $\mathbb{R}^2$ specified in polar coordinates as

$$x(t): r = r(t), \varphi = \varphi(t) \qquad (1)$$

We can simply use the chain rule. The map $t \mapsto x(t)$ can be considered as the composition of the maps $t \mapsto (r(t), \varphi(t)), (r, \varphi) \mapsto x(r, \varphi)$. Then, by the chain rule, we have

$$\dot{x} = \frac{dx}{dt} = \frac{\partial x}{\partial r}\frac{dr}{dt} + \frac{\partial x}{\partial \varphi}\frac{d\varphi}{dt} = \frac{\partial x}{\partial r}\dot{r} + \frac{\partial x}{\partial \varphi}\dot{\varphi} \qquad (2)$$

Here $\dot{r}$ and $\dot{\varphi}$ are scalar coefficients depending on t, whence the partial derivatives $\partial x/\partial r, \partial x/\partial \varphi$ are vectors depending on point in $\mathbb{R}^2$. We can compare this with the formula in the "standard" coordinates: $\dot{x} = e_1\dot{x} + e_2\dot{y}$. Consider the vectors $\partial x/\partial r, \partial x/\partial \varphi$. Explicitly we have

$$\frac{\partial x}{\partial r} = (\cos\varphi, \sin\varphi) \qquad (3)$$

$$\frac{\partial x}{\partial \varphi} = (-r\sin\varphi, r\cos\varphi) \qquad (4)$$

From where it follows that these vectors make a basis at all points except for the origin (where $r = 0$). It is instructive to sketch a picture, drawing vectors corresponding to a point as starting from that point. Notice that $\partial x/\partial r, \partial x/\partial \varphi$ are, respectively, the velocity vectors for the curves $r \mapsto x(r, \varphi)$ $(\varphi = \varphi_0\ fixed)$ and $\varphi \mapsto x(r, \varphi)$ $(r = r_0\ fixed)$. We can conclude that for an arbitrary curve given in polar coordinates the velocity vector will have components $(\dot{r}, \dot{\varphi})$ if as a basis we take $e_r := \partial x/\partial r, e_{\varphi} := \partial x/\partial \varphi$:

$$\dot{x} = e_r\dot{r} + e_{\varphi}\dot{\varphi} \qquad (5)$$

A characteristic feature of the basis e_r, e_{φ} is that it is not "constant" but depends on point. Vectors "stuck to points" when we consider curvilinear coordinates.

Proposition 1.3. The velocity vector has the same appearance in all coordinate systems.

Proof. Follows directly from the chain rule and the transformation law for the basis e_i. In particular, the elements of the basis $e_i = \partial x/\partial x^i$ (originally, a formal notation) can be understood directly as the velocity vectors of the coordinate lines $x^i \mapsto x(x^1, \ldots, x^n)$ (all coordinates but x^i are fixed). Since we now know how to handle velocities in arbitrary coordinates, the best way to treat the differential of a map $F:\square^n \to \square^m$ is by its action on the velocity vectors. By definition, we set

$$dF(x_0):\frac{dx(t)}{dt}(t_0)\mapsto\frac{dF(x(t))}{dt}(t_0) \qquad (1)$$

Now $dF(x_0)$ is a linear map that takes vectors attached to a point $x_0\in\square^n$ to vectors attached to the point $F(x)\in\square^m$

$$dF=\frac{\partial F}{\partial x^1}dx^1+...+\frac{\partial F}{\partial x^n}dx^n$$

$$(e_1,...,e_m)\begin{pmatrix}\frac{\partial F^1}{\partial x^1}\cdots\frac{\partial F^1}{\partial x^n}\\ ...\ \ ...\ \ ...\\ \frac{\partial F^m}{\partial x^1}\cdots\frac{\partial F^m}{\partial x^n}\end{pmatrix}\begin{pmatrix}dx^1\\ ...\\ dx^n\end{pmatrix}, \qquad (2)$$

In particular, for the differential of a function we always have

$$df=\frac{\partial f}{\partial x^1}dx^1+...+\frac{\partial f}{\partial x^n}dx^n, \qquad (3)$$

Where x^i are arbitrary coordinates. The form of the differential does not change when we perform a change of coordinates.

Example 1.3 Consider a 1-form in $\square^2$ given in the standard coordinates:
$A=-ydx+xdy$ In the polar coordinates we will have $x=r\cos\varphi, y=r\sin\varphi$, hence

$$dx=\cos\varphi dr-r\sin\varphi d\varphi$$

$$dy=\sin\varphi dr+r\cos\varphi d\varphi$$

Substituting into A, we get

$$A=-r\sin\varphi(\cos\varphi dr-r\sin\varphi d\varphi)$$

$$+r\cos\varphi(\sin\varphi dr+r\cos\varphi d\varphi)$$

$$=r^2(\sin^2\varphi+\cos^2\varphi)d\varphi=r^2d\varphi$$

Hence $A=r^2d\varphi$ is the formula for A in the polar coordinates. In particular, we see that this is again a 1-form, a linear combination of the differentials of coordinates with functions as coefficients. Secondly, in a more conceptual way, we can define a 1-form in a domain U as a linear function on vectors at every point of U :

$$\omega(\upsilon)=\omega_1\upsilon^1+...+\omega_n\upsilon^n, \qquad (1)$$

If $\upsilon=\sum e_i\upsilon^i$, where $e_i=\partial x/\partial x^i$. Recall that the differentials of functions were defined as linear functions on vectors (at every point), and

$$dx^i(e_j)=dx^i\left(\frac{\partial x}{\partial x^j}\right)=\delta^i_j \qquad (2)$$

at every point x.

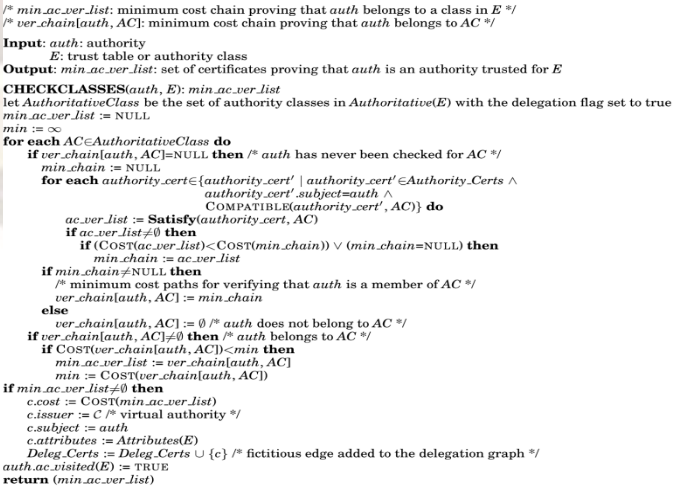
Theorem 1.9. For arbitrary 1-form ω and path γ, the integral $\int_\gamma \omega$ does not change if we change parametrization of γ provide the orientation remains the same.

Proof: Consider $\left\langle\omega(x(t)),\frac{dx}{dt'}\right\rangle$ and $\left\langle\omega(x(t(t'))),\frac{dx}{dt'}\right\rangle$ As

$$\left\langle\omega(x(t(t'))),\frac{dx}{dt'}\right\rangle=\left|\left\langle\omega(x(t(t'))),\frac{dx}{dt'}\right\rangle.\frac{dt}{dt'}\right.,$$

C. *Functions Invoked by Function Satisfy*

We discuss in detail the functions invoked by function Satisfy, briefly described in Section 4, and provide an example of their execution. CheckClasses. Function CheckClasses (see Figure 12) receives as input the authority auth to be checked, a trust table or authority class E, and verifies whether auth is a member of an authority class in the authoritative clause of E (i.e., if auth has an authority certificate that directly or indirectly proves that it belongs to a class trusted for E). If this is the case, CheckClasses returns the delegation chains proving such a membership; an empty set, otherwise. Variable min ac ver list contains the delegation chains with minimum cost, proving that auth is a member of a class trusted for E; variable min corresponds to the cost of min ac ver list; and variable ver chain[auth, AC] contains the delegation chains proving that auth is a member of class AC. Initially, variables min ac ver list and min are set to NULL and ∞, respectively.



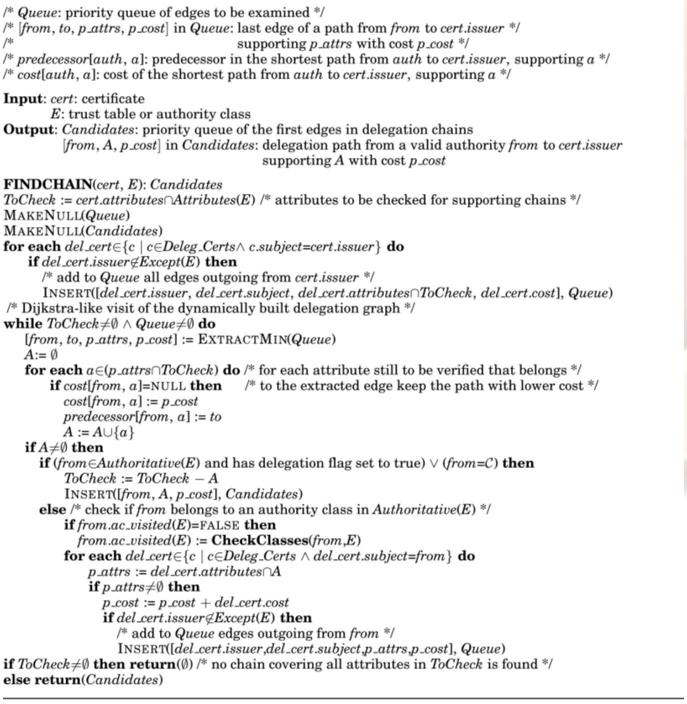
```
/* min_ac_ver_list: minimum cost chain proving that auth belongs to a class in E */
/* ver_chain[auth, AC]: minimum cost chain proving that auth belongs to AC */

Input: auth: authority
       E: trust table or authority class
Output: min_ac_ver_list: set of certificates proving that auth is an authority trusted for E

CHECKCLASSES(auth, E): min_ac_ver_list
let AuthoritativeClass be the set of authority classes in Authoritative(E) with the delegation flag set to true
min_ac_ver_list := NULL
min := ∞
for each AC∈AuthoritativeClass do
    if ver_chain[auth, AC]=NULL then /* auth has never been checked for AC */
      min_chain := NULL
      for each authority_cert∈{authority_cert′ | authority_cert′∈Authority_Certs ∧
                               authority_cert′.subject=auth ∧
                               COMPATIBLE(authority_cert′, AC)} do
          ac_ver_list := Satisfy(authority_cert, AC)
          if ac_ver_list≠∅ then
            if (COST(ac_ver_list)<COST(min_chain)) ∨ (min_chain=NULL) then
              min_chain := ac_ver_list
      if min_chain≠NULL then
        /* minimum cost paths for verifying that auth is a member of AC */
        ver_chain[auth, AC] := min_chain
      else
        ver_chain[auth, AC] := ∅ /* auth does not belong to AC */
    if ver_chain[auth, AC]≠∅ then /* auth belongs to AC */
      if COST(ver_chain[auth, AC])<min then
        min_ac_ver_list := ver_chain[auth, AC]
        min := COST(ver_chain[auth, AC])
if min_ac_ver_list≠∅ then
    c.cost := COST(min_ac_ver_list)
    c.issuer := C /* virtual authority */
    c.subject := auth
    c.attributes := Attributes(E)
    Deleg_Certs := Deleg_Certs ∪ {c} /* fictitious edge added to the delegation graph */
auth.ac_visited(E) := TRUE
return (min_ac_ver_list)
```

Fig. 12. Function checking if auth belongs to a class in Authoritative(E).

Let AuthoritativeClass be the set of authority classes in Authoritative(E) with the delegation option. For each authority class AC in AuthoritativeClass, if ver chain[auth, AC] is NULL, CheckClasses checks whether auth is a member of AC. Variable min chain is set to NULL and is used to store the chains with minimum cost, supporting the membership of auth to AC. For each authority certificate authority cert issued for auth and compatible with AC, CheckClasses recursively calls function Satisfy. At the end of the recursive call, variable ac ver list contains a set (possibly empty) of certificates forming the delegation chains for auth. If ac ver list is not empty, auth is a member of class AC, and therefore if the cost of ac ver list is less than the cost of the current best solution (min chain), ac ver list becomes the new best solution and min chain is set to ac ver list. When all authority certificates compatible with AC have been verified, if min chain is different from NULL, variable ver chain[auth, AC] is set to min chain; otherwise, auth is not a member of AC, and therefore ver chain[auth, AC] is set to ∅. If auth is a member of AC (i.e., ver chain[auth, AC] is not NULL), CheckClasses compares the cost of the delegation chain in ver chain[auth, AC] with the cost (variable min) of the delegation chain that is currently the chain with minimum cost and that proves that auth is a member of an authority class in the authoritative clause of E. If the cost of the delegation chain in ver chain[auth, AC] is less than min, ver chain[auth, AC] becomes the new best solution. Therefore, min ac ver list is set to ver chain[auth, AC] and min is set to the cost of ver chain[auth, AC]. When all classes in AuthoritativeClass have been processed, if min ac ver list is not NULL, CheckClasses adds a virtual delegation certificate c to Deleg Certs, where the issuer is a virtual authority C, the subject is auth, the cost of the certificate is COST(min ac ver list), and the set of certified attributes is Attributes(E). Finally, flag auth.ac visited(E) is set to true, meaning that it has already been analyzed as to whether auth is a member of an authority class listed in Authoritative(E). The function then returns the verification chain min ac ver list. FindChain. Function FindChain (see Figure 13) receives as input a certificate cert, and a trust table or an authority class E. It returns a (possibly empty) set of supporting chains for the attributes in variable ToCheck, initially set to cert.attributes∩Attributes(E). Finding a supporting chain for an attribute a means finding a path in the delegation graph ending in cert.issuer and starting at one of the authorities in Authoritative(E) or at an authority that belongs to one of the authority classes listed in Authoritative(E), such that the labels of all edges in the path include attribute a. The process for finding supporting chains is performed via a Dijkstra-like visit of the delegation graph, possibly extended with the virtual certificates. The while loop iterates until either a chain has been retrieved for all attributes (ToCheck is empty) or there are no more edges to examine (Queue is empty). When a path (chain) ends in C or in an authority in the authoritative clause of E, an element of the form [from, A, p cost] is added to the Candidates queue, where from represents the authority from which there is a path reaching cert.issuer; A is the set of supported attributes; and p cost is the cost of the path. Otherwise, if the authority from reached by the path is not a valid authority for E and flag from.ac visited(E) is false, function FindChain calls function CheckClasses on authority from to check if it

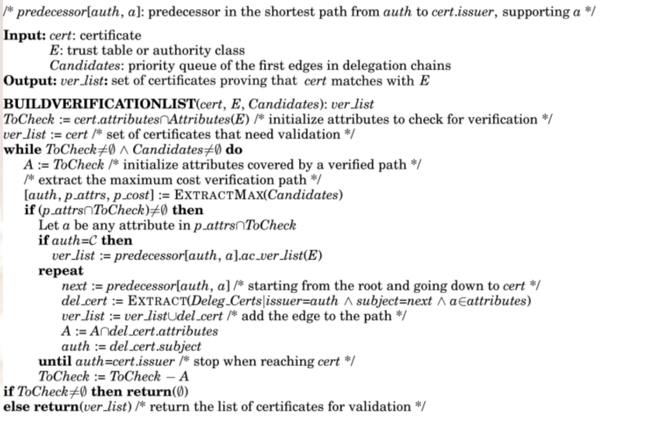


```
/* Queue: priority queue of edges to be examined */
/* [from, to, p_attrs, p_cost] in Queue: last edge of a path from from to cert.issuer */
/*                                supporting p_attrs with cost p_cost */
/* predecessor[auth, a]: predecessor in the shortest path from auth to cert.issuer, supporting a */
/* cost[auth, a]: cost of the shortest path from auth to cert.issuer, supporting a */

Input: cert: certificate
       E: trust table or authority class
Output: Candidates: priority queue of the first edges in delegation chains
        [from, A, p_cost] in Candidates: delegation path from a valid authority from to cert.issuer
                                supporting A with cost p_cost

FINDCHAIN(cert, E): Candidates
ToCheck := cert.attributes∩Attributes(E) /* attributes to be checked for supporting chains */
MAKENULL(Queue)
MAKENULL(Candidates)
for each del_cert∈{c | c∈Deleg_Certs∧ c.subject=cert.issuer} do
   if del_cert.issuer∉Except(E) then
     /* add to Queue all edges outgoing from cert.issuer */
     INSERT([del_cert.issuer, del_cert.subject, del_cert.attributes∩ToCheck, del_cert.cost], Queue)
 /* Dijkstra-like visit of the dynamically built delegation graph */
while ToCheck≠∅ ∧ Queue≠∅ do
   [from, to, p_attrs, p_cost] := EXTRACTMIN(Queue)
   A:= ∅
   for each a∈(p_attrs∩ToCheck) do /* for each attribute still to be verified that belongs */
      if cost[from, a]=NULL then    /* to the extracted edge keep the path with lower cost */
        cost[from, a] := p_cost
        predecessor[from, a] := to
        A := A∪{a}
   if A≠∅ then
     if (from∈Authoritative(E) and has delegation flag set to true) ∨ (from=C) then
       ToCheck := ToCheck − A
       INSERT([from, A, p_cost], Candidates)
     else /* check if from belongs to an authority class in Authoritative(E) */
       if from.ac_visited(E)=FALSE then
         from.ac_visited(E) := CheckClasses(from,E)
       for each del_cert∈{c | c∈Deleg_Certs ∧ del_cert.subject=from} do
          p_attrs := del_cert.attributes∩A
          if p_attrs≠∅ then
            p_cost := p_cost + del_cert.cost
            if del_cert.issuer∉Except(E) then
              /* add to Queue edges outgoing from from */
              INSERT([del_cert.issuer,del_cert.subject,p_attrs,p_cost], Queue)
if ToCheck≠∅ then return(∅) /* no chain covering all attributes in ToCheck is found */
else return(Candidates)
```

Fig. 13. Function determining supporting chains.



```
/* predecessor[auth, a]: predecessor in the shortest path from auth to cert.issuer, supporting a */

Input: cert: certificate
       E: trust table or authority class
       Candidates: priority queue of the first edges in delegation chains
Output: ver_list: set of certificates proving that cert matches with E

BUILDVERIFICATIONLIST(cert, E, Candidates): ver_list
ToCheck := cert.attributes∩Attributes(E) /* initialize attributes to check for verification */
ver_list := cert /* set of certificates that need validation */
while ToCheck≠∅ ∧ Candidates≠∅ do
   A := ToCheck /* initialize attributes covered by a verified path */
   /* extract the maximum cost verification path */
   [auth, p_attrs, p_cost] := EXTRACTMAX(Candidates)
   if (p_attrs∩ToCheck)≠∅ then
     Let a be any attribute in p_attrs∩ToCheck
     if auth=C then
       ver_list := predecessor[auth, a].ac_ver_list(E)
     repeat
        next := predecessor[auth, a] /* starting from the root and going down to cert */
        del_cert := EXTRACT(Deleg_Certs|issuer=auth ∧ subject=next ∧ a∈attributes)
        ver_list := ver_list∪del_cert /* add the edge to the path */
        A := A∩del_cert.attributes
        auth := del_cert.subject
     until auth=cert.issuer /* stop when reaching cert */
     ToCheck := ToCheck − A
if ToCheck≠∅ then return(∅)
else return(ver_list) /* return the list of certificates for validation */
```

Fig. 14. Function building the set of certificates to be validated.

belongs to an authority class in the authoritative clause of E. At the end of the while loop, if ToCheck is not empty (i.e., no chain has been found for some attributes), FindChain returns an empty set (and consequently function Satisfy terminates, returning an empty verification list); it returns priority queue Candidates, representing the delegation chains for the attributes of interest, otherwise. BuildVerificationList. Function BuildVerificationList (see Figure 14) receives as input a certificate cert, a trust table or an authority class E, and a priority queue Candidates of delegation chains, and returns the list of certificates that need to be validated, by removing redundant chains from Candidates. Variable ToCheck is initialized to the set of attributes to be verified (i.e., cert.attributes∩Attributes(E)). To minimize

We introduce three lemmas used in the proof of Theorem 4.6 on the termination and correctness of our delegation chain verification algorithm.

LEMMA A.3. Given a finite set of delegation certificates, Deleg Certs, a finite set of authority certificates Authority Certs and a certificate cert compatible with an entity E, each element [from, to, p attrs, p cost] in the Queue used by function FindChain represents a path starting from from, ending to cert.issuer, and supporting attributes p attrs ⊆ {cert.attributes∩Attributes(E)} with cost p cost.

PROOF. We prove this property by induction.
Base Case. This property is satisfied before entering the while loop, since Queue contains all edges (delegation certificates) entering cert.issuer and supporting attributes in del cert.attributes∩ToCheck, where ToCheck is initialized to cert.attributes ∩Attributes(E). Recursive Case. Suppose that this property is valid for all elements in Queue. We now prove that the new elements added to Queue always satisfy the above property. Let [from, to, p attrs, p cost] be the extracted element from Queue. By induction, we know that it represents a path from from to cert.issuer with cost p cost, and supports a set of attributes p attrs ⊆ {cert.attributes∩Attributes(E)}. If there is at least one attribute in {p attrs∩ToCheck} for which the extracted path is the current supporting path with minimum cost (i.e., cost[from, a]=NULL), and from is neither an authority directly listed in the authoritative clause of E nor the virtual authority C, the attribute is added to A, where A ⊆p attrs, and the edges entering from (i.e., the delegation certificates del cert with subject from) are added to Queue. For each edge, function FindChain then adds to Queue an element of the form [del cert.issuer, from, A∩del cert.attributes, p cost+del cert.cost] which therefore represents a path from del cert.issuer to cert.issuer, supporting a set of attributes included in cert.attributes∩Attributes(E).

LEMMA A.4. Given a finite set of delegation certificates Deleg Certs, a finite set of authority certificates Authority Certs, and a certificate cert compatible with an entity E, function FindChain terminates and determines correct delegation chains.
PROOF. We prove the termination of function FindChain by induction. Base Case. Function FindChain includes a for each loop followed by a while loop. The for each loop iterates on the delegation certificates in Deleg Certs with subject cert.issuer. Since Deleg Certs is finite, the for each loop terminates. The while loop terminates when either ToCheck becomes empty or Queue becomes empty. Initially, ToCheck contains all attributes in cert.attributes∩Attributes(E) and Queue contains all edges (delegation certificates) entering cert.issuer. At each iteration, the number of attributes in ToCheck may only decrease, one element is extracted from Queue, and new elements may be added to Queue. In particular, since the attributes in ToCheck are removed when FindChain has found a supporting path for them, two cases may occur. In the first case, FindChain finds a supporting path for all attributes in ToCheck, and therefore it becomes empty. In the second case, there is at least one attribute such that the supporting path does not exist. The while loop therefore terminates only if Queue becomes empty, and we then need to prove that the number of elements inserted into Queue is finite. At each iteration of the while loop, an element [from, to, p attrs, p cost] with minimum cost is extracted from Queue. If for all attributes p attrs, function FindChain has already found a delegation chain (i.e., p attrs∩ToCheck is empty), no element is added to Queue. Otherwise, p attrs∩ToCheck is not empty. Now let a be an attribute in {p attrs∩ToCheck}. If the extracted element represents an edge in the first path supporting a and passing through from (Lemma A.3), attribute a is added to A and the predecessor of from becomes authority to (i.e., predecessor[from, a] := to), thus correctly building the delegation path. Otherwise, if function FindChain has already analyzed another path supporting a and passing through from, p cost must be greater than or equal to the cost of the other path, since, at each iteration of the while loop, the path with minimum cost is extracted. The edges entering from are possibly added to Queue only if A is not empty (in the worst case this may happen as many times as the number of attributes in cert) and from is neither an authority directly listed in the authoritative clause of E nor the virtual authority C. If A is not empty and from is an authority directly listed in the authoritative clause of E or corresponds to C, the set ToCheck is modified by removing the set of attributes A and no element is added to Queue. Since however a finite number of

elements are inserted into Queue and at each iteration of the while loop an element is extracted from Queue, the while loop terminates. If after the termination of the while loop ToCheck is not empty, function FindChain returns an empty set.

Recursive Case.We first note that function FindChain terminates if CheckClasses terminates. Function CheckClasses is composed of two nested for each loops, which iterate on a finite number of authority classes and authority certificates, and therefore the operations inside these two loops are executed a finite number of times. In particular, we are interested in the number of calls to function Satisfy. Given an authority auth, and an authority class AC, the membership of auth to AC is checked only if ver chain[auth, AC] is NULL and there exists at least one authority certificate issued for auth and compatible with AC. In particular, the number of times that function Satisfy is called for checking whether auth is a member of AC is equal to the number of authority certificates issued for auth and compatible with AC. Since the delegation graph is acyclic, the recursive call to Satisfy cannot reach auth again. Furthermore, since the number of authority classes and delegation certificates is finite and since each call to function Satisfy visits a different portion of such a graph, the recursive call to Satisfy terminates (provided function Satisfy terminates, as proved by Theorem 4.6). When the inner for each loop also terminates, ver chain[auth, AC] becomes different from NULL and therefore the membership of auth to AC will not be checked any more.

II. PROOF OF CORRECTNESS AND COMPLEXITY THEOREMS

LEMMA A.5. Given a finite set of delegation certificates Deleg Certs, a finite set of authority certificates Authority Certs, and a certificate cert compatible with an entity E, function BuildVerificationList terminates.

PROOF. Function BuildVerificationList is composed of a while loop and an innermost repeat-until loop. The while loop terminates when either ToCheck becomes empty or Candidates becomes empty. Initially, ToCheck contains all attributes in cert.attributes∩Attributes(E) and Candidates contains a finite set of elements of the form [auth, p attrs, p cost] that represents a path staring from auth and ending in cert.issuer, with cost p cost, and supporting attributes p attrs (Lemma A.4). Since at each iteration of the while loop, an element [auth, p attrs, p cost] with maximum cost is extracted from Candidates and no element is added, Candidates will become empty. The innermost repeat-until loop is executed whenever the extracted element represents a path supporting at least one attribute a for which the corresponding delegation chain has not been visited yet, that is, a ∈ p attrs∩ToCheck. The repeat-until loop follows the corresponding path by exploiting the predecessor global variable, and the corresponding delegation certificates are added to ver list. Since function FindChain correctly builds the delegation chains through variable predecessor (Lemma A.4) and the graph is acyclic, at each iteration the repeat-until loop visits an edge of the delegation chain until cert.issuer is reached. Consequently, the repeat-until loop terminates, and if ToCheck is empty, the function returns the set ver list of certificates supporting attributes in cert.attributes∩Attributes(E).

THEOREM 4.6 (TERMINATION AND CORRECTNESS). Given a finite set of delegation certificates Deleg Certs, a finite set of authority certificates Authority Certs, and a certificate cert compatible with an entity E, function Satisfy terminates and determines a correct set of delegation chains for cert, if such delegation chains exist.

PROOF. We first prove that the function terminates and then that if it returns a nonempty set of certificates, they correspond to delegation chains that support all attributes in cert.attributes∩Attributes(E). The preliminary operations of function Satisfy check whether or not cert.issuer appears in Except(E) and Authoritative(E). Since such sets are finite, the preliminary operations terminate. Analogously, since both function FindChain and function BuildVerificationList terminate (Lemma A.4 and Lemma A.5); both Phase 1 and Phase 2 of function Satisfy also terminate. We now prove by contradiction that if there are delegation chains supporting all attributes in cert.attributes∩Attributes(E), function Satisfy returns them. Suppose that all attributes in cert.attributes∩Attributes(E) are supported by a unique delegation chain and let s be the set of delegation certificates forming the delegation chains for such attributes. Suppose also that function Satisfy returns an empty set. In this case, solution s cannot coincide with cert, since Satisfy checks if cert represents a solution to the problem. Then set s has to include more than one certificate. Since function Satisfy returns the empty set, function FindChain returned an empty Candidates list. Therefore, the while loop in function FindChain terminates because Queue becomes empty, while ToCheck includes at least one attribute a. By assumption, we know that there is a supporting path p for attribute a also, going from a valid authority for E to cert.issuer. By Lemma A.3, the edge in p entering cert.issuer and represented by element [from, cert.issuer, p attrs, p cost] is inserted into Queue before the while loop. Since, at the end of the while loop, Queue is empty, within an iteration of the while loop this edge is extracted from Queue, and therefore any edge

entering from, including the edge in path p, is inserted into Queue. By recursively applying this observation, we can conclude that all edges in p are evaluated and added to Queue until the function reaches an authority directly listed in the authoritative clause of E or the virtual authority C. Attribute a is then removed from ToCheck, thus contradicting our assumption that a remains in ToCheck. Function Satisfy can also return an empty set if function BuildVerificationList returns an empty ver list; that is, if the function cannot reconstruct a delegation
chain for all the attributes in cert.attributes∩Attributes(E). Suppose that a is the attribute for which it cannot reconstruct the delegation chain. Since a belongs to cert.attributes∩Attributes(E), function FindChain has removed a from ToCheck and added an element to Candidates containing a. Therefore, such an element is extracted from Candidates by function BuildVerificationList because, by assumption, there is only a unique path supporting a, and this path is reconstructed, since at each iteration of the repeat-until loop, variable next is assigned to the predecessor associated with auth for a. For Lemma A.3, cert.issuer is reached by the repeat-until loop and ToCheck is updated removing a.

THEOREM 4.7 (COMPLEXITY). Function Satisfy finds delegation chains for a certificate cert compatible with an entity E in O(Nd · NE · NAC · Nauth · log(Nd · NE)).

PROOF. The complexity of function Satisfy is obtained by evaluating the complexity of the preliminary checks and of the two phases composing it. We first evaluate the complexity of Satisfy in the base case, thus assuming that function CheckClasses is never called.

Preliminary Checks. The preliminary checks in Satisfy require (in total) time proportional to |Authoritative(E)| + |Except(E)|.

Phase 1. The cost of this phase is the cost of function FindChain. The for each loop of the function requires time proportional to Nd, and the cost of the while loop of the function requires time proportional to the number of iterations of such a loop. In the worst case, the while loop terminates when Queue is empty. As already noted, any certificate (edge) in Deleg Certs is inserted into Queue at most as many times as the number of attributes NE. This implies that the maximum number of elements in Queue is Nd · NE. All operations performed within the while loop have a constant cost, but the INSERT operation on priority queue Queue, whose cost is O(log(Nd ·NE)).We can then conclude that the cost of this first phase is proportional to Nd · NE · log(Nd · NE).

Phase 2. The cost of this phase is the cost of function BuildVerificationList that visits (a subset of) the paths found in the previous phase. The cost of this second phase is then proportional to Nd. Overall, the time complexity is proportional to |Authoritative(E)| + |Except(E)| + Nd · NE · log(Nd · NE)+ Nd. If we assume that all operations performed by function Satisfy have a constant cost and cmax is the maximum cost, the time complexity is in O(cmax · Nd · NE · log(Nd · NE)) = O(Nd · NE · log(Nd · NE)). Consider now the recursive execution of function Satisfy. As already noted, in the worst case, the function is recursively called NAC · Nauth times. Since the time complexity of each call is O(Nd ·NE ·log(Nd ·NE)) and in the worst case the whole delegation graph is visited, the overall time complexity is then O(Nd · NE · NAC · Nauth · log(Nd · NE)).

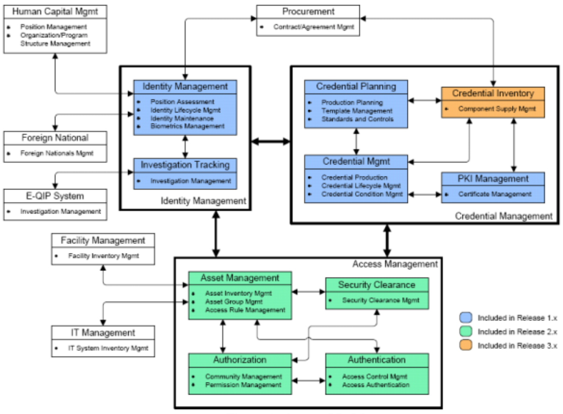


Figure 4: System Model

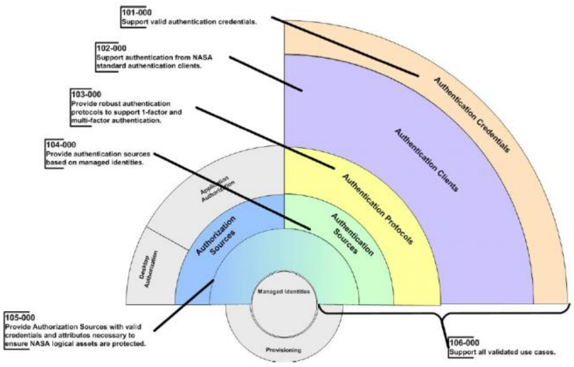

101-000
Support valid authentication credentials.
102-000
Support authentication from NASA standard authentication clients.
103-000
Provide robust authentication protocols to support 1-factor and multi-factor authentication.
104-000
Provide authentication sources based on managed identities.
105-000
Provide Authorization Sources with valid credentials and attributes necessary to ensure NASA logical assets are protected.
106-000
Support all validated use cases.
Authentication Credentials
Authentication Clients
Authentication Protocols
Authentication Sources
Authorization Sources
Application Authorization
Desktop Authorization
Managed Identities
Provisioning

Figure 5: Logical Access Control Framework

Let p be a rational prime and let $K = \square(\zeta_p)$. We write ζ for ζ_p or this section. Recall that K has degree $\varphi(p) = p-1$ over $\square$. We wish to show that $O_K = \square[\zeta]$. Note that ζ is a root of $x^p - 1$, and thus is an algebraic integer; since O_K is a ring we have that $\square[\zeta] \subseteq O_K$. We give a proof without assuming unique factorization of ideals. We begin with some norm and trace computations. Let j be an integer. If j is not divisible by p, then ζ^j is a primitive p^{th} root of unity, and thus its conjugates are $\zeta,\zeta^2,...,\zeta^{p-1}$. Therefore

$$Tr_{K/\Box}(\zeta^j)=\zeta+\zeta^2+...+\zeta^{p-1}=\Phi_p(\zeta)-1=-1$$

If p does divide j, then $\zeta^j=1$, so it has only the one conjugate 1, and $Tr_{K/\Box}(\zeta^j)=p-1$ By linearity of the trace, we find that

$$Tr_{K/\Box}(1-\zeta)=Tr_{K/\Box}(1-\zeta^2)=...$$

$$=Tr_{K/\Box}(1-\zeta^{p-1})=p$$

We also need to compute the norm of $1-\zeta$. For this, we use the factorization

$$x^{p-1}+x^{p-2}+...+1=\Phi_p(x)$$

$$=(x-\zeta)(x-\zeta^2)...(x-\zeta^{p-1});$$

Plugging in $x=1$ shows that

$$p=(1-\zeta)(1-\zeta^2)...(1-\zeta^{p-1})$$

Since the $(1-\zeta^j)$ are the conjugates of $(1-\zeta)$, this shows that $N_{K/\Box}(1-\zeta)=p$ The key result for determining the ring of integers O_K is the following.

LEMMA 1.9

$$(1-\zeta)O_K\cap\Box=p\Box$$

Proof. We saw above that p is a multiple of $(1-\zeta)$ in O_K, so the inclusion $(1-\zeta)O_K\cap\Box\supseteq p\Box$ is immediate. Suppose now that the inclusion is strict. Since $(1-\zeta)O_K\cap\Box$ is an ideal of $\Box$ containing $p\Box$ and $p\Box$ is a maximal ideal of $\Box$, we must have $(1-\zeta)O_K\cap\Box=\Box$ Thus we can write

$$1=\alpha(1-\zeta)$$

For some $\alpha\in O_K$. That is, $1-\zeta$ is a unit in O_K.

COROLLARY 1.1 For any $\alpha\in O_K$, $Tr_{K/\Box}((1-\zeta)\alpha)\in p.\Box$

PROOF. We have

$$Tr_{K/\Box}((1-\zeta)\alpha)=\sigma_1((1-\zeta)\alpha)+...+\sigma_{p-1}((1-\zeta)\alpha)$$

$$=\sigma_1(1-\zeta)\sigma_1(\alpha)+...+\sigma_{p-1}(1-\zeta)\sigma_{p-1}(\alpha)$$

$$=(1-\zeta)\sigma_1(\alpha)+...+(1-\zeta^{p-1})\sigma_{p-1}(\alpha)$$

Where the σ_i are the complex embeddings of K (which we are really viewing as automorphisms of K) with the usual ordering. Furthermore, $1-\zeta^j$ is a multiple of $1-\zeta$ in O_K for every $j\neq 0$. Thus $Tr_{K/\Box}(\alpha(1-\zeta))\in(1-\zeta)O_K$ Since the trace is also a rational integer.

PROPOSITION 1.4 Let p be a prime number and let $K=|\Box(\zeta_p)$ be the p^{th} cyclotomic field. Then $O_K=\Box[\zeta_p]\cong\Box[x]/(\Phi_p(x));$ Thus $1,\zeta_p,...,\zeta_p^{p-2}$ is an integral basis for O_K.

PROOF. Let $\alpha\in O_K$ and write $\alpha=a_0+a_1\zeta+...+a_{p-2}\zeta^{p-2}$ With $a_i\in\Box$. Then

$$\alpha(1-\zeta)=a_0(1-\zeta)+a_1(\zeta-\zeta^2)+...$$

$$+a_{p-2}(\zeta^{p-2}-\zeta^{p-1})$$

By the linearity of the trace and our above calculations we find that $Tr_{K/\Box}(\alpha(1-\zeta))=pa_0$ We also have $Tr_{K/\Box}(\alpha(1-\zeta))\in p\Box$, so $a_0\in\Box$ Next consider the algebraic integer $(\alpha-a_0)\zeta^{-1}=a_1+a_2\zeta+...+a_{p-2}\zeta^{p-3};$ This is an algebraic integer since $\zeta^{-1}=\zeta^{p-1}$ is. The same argument as above shows that $a_1\in\Box$, and continuing in this way we find that all of the a_i are in $\Box$. This completes the proof.

Example 1.4 Let $K=\Box$, then the local ring $\Box_{(p)}$ is simply the subring of $\Box$ of rational numbers with denominator relatively prime to p. Note that this ring $\Box_{(p)}$ is not the ring $\Box_p$ of p-adic integers; to get $\Box_p$ one must complete $\Box_{(p)}$. The usefulness of $O_{K,p}$ comes from the fact that it has a particularly simple ideal structure. Let a be any proper ideal of $O_{K,p}$ and consider the ideal $a\cap O_K$ of O_K. We claim that $a=(a\cap O_K)O_{K,p};$ That is, that a is generated by the elements of a in $a\cap O_K$. It is clear from the definition of an ideal that $a\supseteq(a\cap O_K)O_{K,p}$. To prove the other inclusion, let α be any element of a. Then we can write $\alpha=\beta/\gamma$ where

$\beta \in O_K$ and $\gamma \notin p$. In particular, $\beta \in a$ (since $\beta / \gamma \in a$ and a is an ideal), so $\beta \in O_K$ and $\gamma \notin p$. so $\beta \in a \cap O_K$. Since $1/\gamma \in O_{K,p}$, this implies that $\alpha = \beta / \gamma \in (a \cap O_K) O_{K,p}$, as claimed.We can use this fact to determine all of the ideals of $O_{K,p}$. Let a be any ideal of $O_{K,p}$ and consider the ideal factorization of $a \cap O_K$ in O_K. write it as $a \cap O_K = p^n b$ For some n and some ideal b, relatively prime to p. we claim first that $bO_{K,p} = O_{K,p}$. We now find that

$$a = (a \cap O_K) O_{K,p} = p^n b O_{K,p} = p^n O_{K,p}$$

Since $bO_{K,p}$. Thus every ideal of $O_{K,p}$ has the form $p^n O_{K,p}$ for some n; it follows immediately that $O_{K,p}$ is noetherian. It is also now clear that $p^n O_{K,p}$ is the unique non-zero prime ideal in $O_{K,p}$. Furthermore, the inclusion $O_K \mapsto O_{K,p} / pO_{K,p}$ Since $pO_{K,p} \cap O_K = p$, this map is also surjection, since the residue class of $\alpha / \beta \in O_{K,p}$ (with $\alpha \in O_K$ and $\beta \notin p$) is the image of $\alpha\beta^{-1}$ in $O_{K/p}$, which makes sense since β is invertible in $O_{K/p}$. Thus the map is an isomorphism. In particular, it is now abundantly clear that every non-zero prime ideal of $O_{K,p}$ is maximal. To show that $O_{K,p}$ is a Dedekind domain, it remains to show that it is integrally closed in K. So let $\gamma \in K$ be a root of a polynomial with coefficients in $O_{K,p}$; write this polynomial as $x^m + \frac{\alpha_{m-1}}{\beta_{m-1}} x^{m-1} + \ldots + \frac{\alpha_0}{\beta_0}$ With $\alpha_i \in O_K$ and $\beta_i \in O_{K-p}$. Set $\beta = \beta_0 \beta_1 \ldots \beta_{m-1}$. Multiplying by β^m we find that $\beta\gamma$ is the root of a monic polynomial with coefficients in O_K. Thus $\beta\gamma \in O_K$; since $\beta \notin p$, we have $\beta\gamma / \beta = \gamma \in O_{K,p}$. Thus $O_{K,p}$ is integrally close in K.

COROLLARY 1.2. Let K be a number field of degree n and let α be in O_K then $N'_{K/\square}(\alpha O_K) = \left| N_{K/\square}(\alpha) \right|$

PROOF. We assume a bit more Galois theory than usual for this proof. Assume first that $K/\square$ is Galois. Let σ be an element of $Gal(K/\square)$. It is clear that $\sigma(O_K)/\sigma(\alpha) \cong O_{K/\alpha}$; since $\sigma(O_K) = O_K$, this shows that $N'_{K/\square}(\sigma(\alpha) O_K) = N'_{K/\square}(\alpha O_K)$. Taking the product over all $\sigma \in Gal(K/\square)$, we have $N'_{K/\square}(N_{K/\square}(\alpha) O_K) = N'_{K/\square}(\alpha O_K)^n$ Since $N_{K/\square}(\alpha)$ is a rational integer and O_K is a free$\square$ -module of rank n, $O_K / N_{K/\square}(\alpha) O_K$ Will have order $N_{K/\square}(\alpha)^n$; therefore

$$N'_{K/\square}(N_{K/\square}(\alpha) O_K) = N_{K/\square}(\alpha O_K)^n$$

This completes the proof. In the general case, let L be the Galois closure of K and set $[L:K] = m$.

A. *Authors and Affiliations*

Dr Akash Singh is working with IBM Corporation as an IT Architect and has been designing Mission Critical System and Service Solutions; He has published papers in IEEE and other International Conferences and Journals.

He joined IBM in Jul 2003 as a IT Architect which conducts research and design of High Performance Smart Grid Services and Systems and design mission critical architecture for High Performance Computing Platform and Computational Intelligence and High Speed Communication systems. He is a member of IEEE (Institute for Electrical and Electronics Engineers), the AAAI (Association for the Advancement of Artificial Intelligence) and the AACR (American Association for Cancer Research). He is the recipient of numerous awards from World Congress in Computer Science, Computer Engineering and Applied Computing 2010, 2011, and IP Multimedia System 2008 and Billing and Roaming 2008. He is active research in the field of Artificial Intelligence and advancement in Medical Systems. He is in Industry for 18 Years where he performed various role to provide the Leadership in Information Technology and Cutting edge Technology.

REFERENCES

[1] Dynamics and Control of Large Electric Power Systems. Ilic, M. and Zaborszky, J. John Wiley & Sons, Inc. © 2000, p. 756.

[2] Modeling and Evaluation of Intrusion Tolerant Systems Based on Dynamic Diversity Backups. Meng, K. et al. Proceedings of the 2009 International Symposium on Information Processing

(ISIP'09). Huangshan, P. R. China, August 21-23, 2009, pp. 101–104

[3] Characterizing Intrusion Tolerant Systems Using A State Transition Model. Gong, F. et al., April 24, 2010.

[4] Energy Assurance Daily, September 27, 2007. U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability, Infrastructure Security and Energy Restoration Division. April 25, 2010.

[5] CENTIBOTS Large Scale Robot Teams. Konoledge, Kurt et al. Artificial Intelligence Center, SRI International, Menlo Park, CA 2003.

[6] Handling Communication Restrictions and Team Formation in Congestion Games, Agogino, A. and Tumer, K. Journal of Autonomous Agents and Multi Agent Systems, 13(1):97–115, 2006.

[7] Robotics and Autonomous Systems Research, School of Mechanical, Industrial and Manufacturing Engineering, College of Engineering, Oregon State University

[8] D. Dietrich, D. Bruckner, G. Zucker, and P. Palensky, "Communication and computation in buildings: A short introduction and overview," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3577–3584, Nov. 2010.

[9] V. C. Gungor and F. C. Lambert, "A survey on communication networks for electric system automation," *Comput. Networks*, vol. 50, pp. 877–897, May 2006.

[10] S. Paudyal, C. Canizares, and K. Bhattacharya, "Optimal operation of distribution feeders in smart grids," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4495–4503, Oct. 2011.

[11] D. M. Laverty, D. J. Morrow, R. Best, and P. A. Crossley, "Telecommunications for smart grid: Backhaul solutions for the distribution network," in *Proc. IEEE Power and Energy Society General Meeting*, Jul. 25–29, 2010, pp. 1–6.

[12] L. Wenpeng, D. Sharp, and S. Lancashire, "Smart grid communication network capacity planning for power utilities," in *Proc. IEEE PES, Transmission Distrib. Conf. Expo.*, Apr. 19–22, 2010, pp. 1–4.

[13] Y. Peizhong, A. Iwayemi, and C. Zhou, "Developing ZigBee deployment guideline under WiFi interference for smart grid applications," *IEEE Trans. Smart Grid*, vol. 2, no. 1, pp. 110–120, Mar. 2011.

[14] C. Gezer and C. Buratti, "A ZigBee smart energy implementation for energy efficient buildings," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC Spring)*, May 15–18, 2011, pp. 1–5.

[15] R. P. Lewis, P. Igic, and Z. Zhongfu, "Assessment of communication methods for smart electricity metering in the U.K.," in *Proc. IEEE PES/IAS Conf. Sustainable Alternative Energy (SAE)*, Sep. 2009, pp. 1–4.

[16] A. Yarali, "Wireless mesh networking technology for commercial and industrial customers," in *Proc. Elect. Comput. Eng., CCECE*,May 1–4, 2008, pp. 000047–000052.

[17] M. Y. Zhai, "Transmission characteristics of low-voltage distribution networks in China under the smart grids environment," *IEEE Trans. Power Delivery*, vol. 26, no. 1, pp. 173–180, Jan. 2011.

[18] V. Paruchuri, A. Durresi, and M. Ramesh, "Securing powerline communications," in *Proc. IEEE Int. Symp. Power Line Commun. Appl., (ISPLC)*, Apr. 2–4, 2008, pp. 64–69.

[19] Q.Yang, J. A. Barria, and T. C. Green, "Communication infrastructures for distributed control of power distribution networks," *IEEE Trans. Ind. Inform.*, vol. 7, no. 2, pp. 316–327, May 2011.

[20] T. Sauter and M. Lobashov, "End-to-end communication architecture for smart grids," *IEEE Trans. Ind. Electron.*, vol. 58, no. 4, pp. 1218–1228, Apr. 2011.

[21] K. Moslehi and R. Kumar, "Smart grid—A reliability perspective," *Innovative Smart Grid Technologies (ISGT)*, pp. 1–8, Jan. 19–21, 2010.

[22] Southern Company Services, Inc., "Comments request for information on smart grid communications requirements," Jul. 2010

[23] R. Bo and F. Li, "Probabilistic LMP forecasting considering load uncertainty," *IEEE Trans. Power Syst.*, vol. 24, pp. 1279–1289, Aug. 2009.

[24] *Power Line Communications*, H. Ferreira, L. Lampe, J. Newbury, and T. Swart (Editors), Eds. New York: Wiley, 2010.

[25] G. Bumiller, "Single frequency network technology for fast ad hoc communication networks over power lines," WiKu-Wissenschaftsverlag Dr. Stein 2010.

[31] G. Bumiller, L. Lampe, and H. Hrasnica, "Power line communications for large-scale control and automation systems," *IEEE Commun. Mag.*, vol. 48, no. 4, pp. 106–113, Apr. 2010.

[32] M. Biagi and L. Lampe, "Location assisted routing techniques for power line communication in smart grids," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 274–278.

[33] J. Sanchez, P. Ruiz, and R. Marin-Perez, “Beacon-less geographic routing made partical: Challenges, design guidelines and protocols,” *IEEE Commun. Mag.*, vol. 47, no. 8, pp. 85–91, Aug. 2009.

[34] N. Bressan, L. Bazzaco, N. Bui, P. Casari, L. Vangelista, and M. Zorzi, “The deployment of a smart monitoring system using wireless sensors and actuators networks,” in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2010, pp. 49–54.

[35] S. Dawson-Haggerty, A. Tavakoli, and D. Culler, “Hydro: A hybrid routing protocol for low-power and lossy networks,” in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2010, pp. 268–273.

[36] S. Goldfisher and S. J. Tanabe, “IEEE 1901 access system: An overview of its uniqueness and motivation,” *IEEE Commun. Mag.*, vol. 48, no. 10, pp. 150–157, Oct. 2010.

[37] V. C. Gungor, D. Sahin, T. Kocak, and S. Ergüt, “Smart grid communications and networking,” Türk Telekom, Tech. Rep. 11316-01, Apr 2011.