Probabilistic Attack on Neural Cryptography

Luís Francisco Seoane Iglesias
DIPLOMARBEIT/MASTER THESIS

Probabilistic Attack on Neural Cryptography

Ausgeführt am Bernstein Center for Computational Neuroscience der
Technische Universität Berlin
Submitted to the Bernstein Center for Computational Neuroscience of the
Technical University of Berlin

Unter der Anleitung von/Under the supervision of

Dr. Andreas Ruttor

Luís Francisco Seoane Iglesias
Matrikelnummer 329576

Berlin, Granada, Barcelona 2009/12

Barcelona, 23 März/May 2012
Affidavit

Eidesstattliche Versicherung  Statutory Declaration
Die selbständige und eigenhändige Aus- I declare in lieu of oath that I have
fertigung versichert an Eides statt written this thesis myself and have not

____________________________

(Datum/Date)

____________________________

(Ort/Place)

____________________________

(Unterschrift/Signature)
forewords

I

At the beginning, Neuron sat on the top of the Teufelsberg and he masturbated while contemplating the virgin world below. His orgasm was like the tremble of a Guitar\(^1\) which stroke the earth so hard that its strings broke up free releasing the chords which fell over the earth during the next 2666 years; and the chords were sacred sperm. And so Neuron created Berlin and the flattened land in its surround. And so he created the Spree, and he created the length that the hand can reach. So was born the Road and the roads, the Night and the nights. The sun was born and the moon, and while the chords would fall, then would appear the (wo)man as well. And all (wo)men stood drinking that holy milk, never growing old.

Well after the chords had posed underneath Berlin, they remained as a tidal wave for ever bending the floor of the city, for ever enlightening whoever would drink the Spree, whoever would listen to its string.

And then in a final shake, huge as a spike which traverses the spinal cord of the universe, Neuron released the remains of the Guitar from his trembling, stiff and ecstatic hands; and those remains rose up and became unsung chords and fell over the young earth for many centuries more\(^2\).

So the universe was born and made: with its structure and its maths, and everything would behave accordingly. But if we were to look at it, its very essence would not be revealed. A dislocation of the being had to take place so we could see.

\(^1\)http://vimeo.com/39597234

\(^2\)The telling might be obscure in what regards the time elapsed, but references in the many sources make clear that the origin of the earth can be traced back to exactly 2666 years before the first man appears. Also, a deeper reading of Neuron’s scrolls makes clear that the genesis took place exactly 2666 months before the first man appears. And that it happened exactly 2666 weeks before the first man appears, and also 2666 days before the first man appears the genesis was to occur again.
II

I had to go down to the Spree (now the corrupted Spree to which Europe dropped an unmoral succession of shot bodies) and drink its seminal waters. I had to summon the holy Road to envisage the straight path which runs firm through an ethereal realm. At either side of this, the mists of History were fighting its never-ending war. I crossed the sacred river where the youth sacrifice itself for glory, for the vaterland, the tsar and the emperor. I crossed it and I was still alive and I turned left and there I founded the clearing were I could lie and rest while a web was growing around me.

Meanwhile my senses had become clearer. I could already hear some of the remote, poetical chords which ruffled the reality since the beginning of time. When I opened my eyes, my eyes had become bigger. When I touched my skin, my skin could sense new shapes. And when I opened my mouth... when I opened my mouth and that obscene bird of the night sung its dark old chants, then my tongue melted away and out my trembling lips.

The web around me had crystallized in a quite dense woodland but I could still peer into the distance and discern the shadows and lights in it. And so I set off and directed myself to the deep darkness.

Were they, the people in the woods, scared? Tired maybe, sleepy, and could not bear the sharp light my feet emanated? They receded at every step I took forth. They covered their faces with their forearms or their hands, and they seemed so nebulous notwithstanding my straight sight. They would vanish in a crackling under my steps if I came too close, and the crackling would be followed by a splintered cloud which climbed swirling, playing with the branches up the trees. Had they tried before the path I sought? And had they grown so tired, so scared? What if science also gets me trapped in a random walk? What’d be its dimension? What its substrate... what is it? Such a random walk. So random and so. Random walk.

I had to hold my self together. Literally. My legs and arms were spread all over the scene, some of my fingers detached as well. My head had landed some
meters away from the trunk, touching with the nose my genitalia –deep breath, wet and sweet. So I had to stand up and pick up my parts. Stuff them in my backpack, dry and rigid as they were. And then I dropped the backpack. I was way too heavy to carry my own weight. The reality was so dense: it had to hold all the light and shadows the woods contained. Way too much. It would just break apart.

Things happen in a space we can’t reach. From there, they yield to our world instances we recognize and put names to; so, many events of the history might be projections from the same underlying entity. One example would be what we named god. Once transcended, it’s easy to see how some genial and powerful event took place in this hidden world; and its instances in our physical world, we called them god. This same event would take place later many times. But the word god had been mystified, amplified. It had taken rust and weight, acquired harsh layers of reality that usage had put upon. Humans couldn’t recognize anymore as godly the instances yielded by the very same event. And so these, as powerful and genial as the original one, were given other names: Cervantes, Leon Tolstoi, Roberto Bolaño, Richard Feynman; blurred with time: Siddhartha, Pitagoras, Euclides... until they are so remote that they melt together in an abstract Unknown from which god drinks.

Each instance yielded to our world hasn’t got the same expression as any other of the same kind. They happen at different moments and they interact with other instances (whether of the same type or different). Each one attains its own, singular realization. Some isolated ripples from different entities can interplay giving rise to phantom items that we identify as independent and that shape our world as well, although no onely transcendental event exists which would project such.

The reality we see is nothing but a light, subtle fabric which pervades everything; which has been lied over a more essential reality where more essential events take place. And it’s their realizations in our world that we can sense and name. We are threads which move freely in this fiber and cross it with the hope of finding something at the other side; but it’s the same old fabric we perceive, no matter the dimension we’ve chosen to traverse it across.

It’s so that it emerges the goddess of History: the eternal tailor. Our free will as a naive struggle against her firm and expert hands. We play childishly; but when the moment arrives, she pulls our threads and the fabric gets wrinkled. And so the rugged and ramified shapes our lives take, and so the universe complete collapses into a moment which joins the tension points that our stitches made.

We are the projection of a standard event from this hidden realm. This standard event yields a human in our physical world each time it takes place, and this event is always one and the same. It was this same event which yielded god. We are the closest to god it ever existed. We have our lives to reverberate, make the fabric shiver, reach with the hands and push the world. Later, later enough so we won’t ever see it, the coarse-graining of generations will preserve our name or will scribble it in a huge gravestone the size of the earth; so humanity can regret-less forget us, melt all of us together so the gods get fed, so the gods moved the world.
When I woke up I was somewhere in Poland, near the German frontier, just past the Oder river. I saw that the sky over me was the ceiling of my friend’s van: a thin panel crisscrossed with random scars. Had I seen the world through those interwoven scars? Could that precarious net hold the insights I had been shown long enough before I would forget?

My eyes hold there, my brain, the lines... a revelation: they are not the lines what I am reading, nor the universe at all; it is my mind through them. Through them cause they are here, as pits or bumps which corrupt the shadows of the lights and the lights of the shadows.

I still remember the pale-blue-oranged sky which was breathing and almost made me cry with emotion. I was wheezing, my heart hurt, and a numbness held my left arm stiff.

Once understood what’s beneath the delicate fabric (the ever-shaped, ever-shaping universal fabric), we are just trying to unveil what’s underneath the patch of it in which our mind manifests. Does our own mind enclose a complexity enough to unveil the mystery?

The definitive cave, the uppermost myth of the cave so far. We run a cryptographic race against ourselves.
acknowledgments

Writing these acknowledgments is by far the hardest part of this Master Thesis. I face two paramount problems. It terrifies me to death the possibility of forgetting someone: so many people I owe to! Even with the recent help of Facebook and other technologies I feel I cannot remember all those who helped me so much. On the other hand I know I can not find the words that each one deserves. The vague acknowledgments I will write here are in most of the cases just the tip of the iceberg: so much I owe!

Sine qua non

First of all I would like to name those without whose direct help it would not have been possible at all to complete this work. Of course, I think that every helping hand I have found was completely necessary to shape my way but if the people and institutions listed right below had not existed this work would not only be radically different, but provably it would not exist at all either.

I am and remain forever thankful to the Fundación Pedro Barrié de la Maza\(^3\). They funded my studies in Berlin through their outstanding scholarship program that, I believe, is of great importance not only for me and my career but also to keep the Galician engine running now and in the future – and for this I shall thank them too. I can only hope the best for this institution and also I hope to repay someday the great investment that they have made in me with my little contribution to the society – and to the Galician society explicitly. As responsible of the scholarship program I must specially thank Mrs. Montserrat Orta Isasi whose kind help solved incredibly many problems.

Without Dr. Andreas Ruttor this thesis would not be a reality at all. To him I thank the confidence he placed in me and the freedom he gave me to research a challenging question on my own. It is not so usual that students can profit of such independence when entering the research world and I understand that making this bet can be riskier for younger researchers such as Dr. Ruttor, therefore I have in a higher esteem the chance he took when he relied on me. From Dr. Ruttor’s partnership with me was born the scientific article which constitutes the core of the present Master Thesis. His experience was determinant to give to that scientific manuscript the quality that it needed to be published and this must be specially acknowledged, also because this text is literally reproduced in chapter III of the present work. Finally I thank him for being the official main supervisor of my Masterarbeit. In this sense I must also

\(^3\)http://www.fundacionbarrie.org/
thank Prof. Dr. Klaus Obermayer for being the second supervisor of this work. I recognize in him a competent researcher and I trust that his evaluation of my scientific work will be rigorous and of very high value.

The Bernstein Center for Computational Neuroscience was my host institution during the thrilling time that I spent in Berlin. I must thank many lecturers for their very interesting contributions and the many enlightments they brought upon me. I would like to expressly thank Prof. Dr. Richard Kempter for his magnificent lectures and also for his very useful guidance and advise and for officially supporting my work at the University of Granada. I shall never thank him enough and would be honored if I ever get the chance to collaborate with him. Prof. Dr. Benjamin Blankertz encouraged me to test my ideas on Brain-Computer Interfaces and his guidance and expertise on this field are of an invaluable worth in the exciting and close-to-science-fiction research that we currently conduct. Very very specially I must and wish thank to Dr. Vanessa Casagrande her infinite patience, help and management through bureaucratic and academic problems. This appreciation must also be extended to Mrs. Margret Franke and Mrs. Julia Schaeffer, but once more Dr. Casagrande’s help must be acknowledged above usual standards.

As part of my studies during the Master Program in Computational Neuroscience that the Bernstein Center hosts in Berlin I made a lab-rotation at the Statistical Physics Group of the University of Granada. I am deeply thankful to the University of Granada, to the mythological and overwhelming city of Granada to where I hope I will have always the opportunity of going back and to the Statistical Physics Group itself for hosting me during the very special time that I spent there. People from there will be listed in the adequate section but now it is the moment for Prof. Dr. Miguel Ángel Muñoz who leaded the fascinating research of which I had the luck of being a partner and whose intelligence and experience I have in great esteem. Mr. Jordi Hidalgo was my companion during that scientific travel and his bright mind and analytic rigor were essential for the success of the work and a pleasure for me.

I can not forget here the enormous contribution of some of my student col-
league from the Bernstein Center to many of the works that I have done during these last years. Therefore I must and would like to thank Mr. Stephan Gabler for encouraging our research in Brain-Computer Interfaces and for the important part of this work of which he is responsible. I also would like to thank Mr. Tiziano D’Albis, Mrs. Lovísa Helgadóttir, Mr. Saikat Ray, Mr. Seyed Hadi Roohani and Mrs. Helene Schmidt – cited in alphabetical order but to which I remain equivalently grateful – for bringing the ReGo project to a kind of reality and for the priceless work and time that they devoted to this and which resulted so lovely to me.

Back in the Womb

This Master Thesis is the bureaucratic culmination of my work during the last two years and a half. This work was developed mainly in Berlin but also in Granada, A Coruña, and Barcelona – or while in a flight, train or bus on my way between arbitrary European cities, indeed. But before such a cosmopolitan life I was settled and had a permanent family; and nothing that came later on my life would had ever happened if it were not for them.

My grandfather Francisco Seoane remains the most direct influence I ever had to become a scientist. As I stop being a teenager – something I am still busy with – I further appreciate his deep values and thoughts. Now his importance grows to me beyond his intelligence or beyond the clairvoyance with which he handled my family’s rhythms notwithstanding his blindness. I acknowledge with deference his unyielding defense of the Enlightenment values and its robust criticism of the scientific method.

I thank my mom for reading in English and German on the phone. Carmen Seoane is not any mother: she is my mother. She deserves a chapter in the human history and the most intense happiness possible in this world and in the worlds behind this one. She was and still is a dam holding a home together. She made me to whom I am through unpayable sacrifices and this little words can never be enough for her, but nothing can after all. I must and wish to acknowledge her coming to Berlin to spend almost a month during the effervescent spring and summer seasons of 2010. One of my most painful experiences is discovering the unraveling worlds that this world contains and not being able to show my mother and the rest of my family the wonders I see, and therefore I specially thank that she came over to see that tiny part of one of the most marvelous realms I ever had the chance to inhabit.

My grandmother Mercedes Iglesias is another paramount person in this life of mine. She is a quiet Galician joy or a sudden flood bringing life. I regret not being able to show her those worlds I was talking about, but she alone already contains a richness enough to satisfy a thirsty traveler. She will surely understand that I have to acknowledge her also because of the lunches and dinners she prepares for me, and to which I look forward to returning soon.

Unluckily I can only briefly mention the rest of my family, but not without saying how much it means to have a family back there when one is abroad. I

\footnote{Criticism must be understood here as the opposite of faith which, I believe, he would reject immediately even if the scientific method itself were the object of this faith.}
have seen many who did not have such a backup and I must say that I feel lucky notwithstanding that I can not share my experiences with them as much as I would like to.

I thank to my aunts and uncles Beli Seoane, Che Pan, Chiruca Seoane, and Marcelino Castro – listed after the alphabetical order of their first names wishing not to make any distinction between them – for being a part of my family, and one to which I like to return.

I owe very much to my cousins David Pan and Manuel Pan, to whom I admire more than they shall believe and more than I can explain. They possess a subtle secret of life that can only be innate and that the brightest genius would die for with little results. I also owe much of what I am – but in a different way – to my other cousins Pili Castro, Merchi Castro, and Lino Castro – this time in anti-alphabetical order.

To all of them, to my niece Cristina Tubío and his father Javier Tubío, and to my nephews Teo Castro and Lucas Castro I am thankful because of their happiness and I celebrate their lives. They are a pot boiling with life, keeping a holy Womb warm for me to return to. I hope I can ever convey to you the things that I have seen, to ever make you see the treasures hidden in this life that I had the enormous luck to discover. Wish you ever join me on my quest.

The Night and the nights

These last years have been for me very much more than scientific maturing. They were also a peak of personal realization. They were the longest and brightest Night\(^5\). I shone along with the nights and the many people I have found on my way. I had the enormous luck of enjoying their talent in the challenging science and art of living, which spans any other field. I admired many and I felt admired by many of them. To all of them I must say that everything, everybody that happened was necessary to me. A stone thrown away alone towards the void is not, as it never interacts with anything and nothing will ever give account of this stone. In the same way, a person thrown away to its enclosed being is not, as nobody can ever give an account of this being. A person can only be if it is along its companions. I would never had been without mines and to them I owe up to the slightest interesting feature they could ever see in me – if it is not because they made me so, which is usually the case, it is then at least because without a counterpart to mirror myself upon I would not be at all.

And the conspicuous place where my being was the most; it was, of course, at Night. In the following I would like to list those who populated my nights in Berlin, Granada, and many other European cities. The presence of these people is mythological, when not sacred.

Javier Molero Segovia, or Javiño, or Pano, or whatever I might name him the next time I see him. His only name. He is by far the person who impacted me the most during these last years. I do not think I can write a single word that would ever make justice to him and to everything that he meant (means) to me. I can just humbly thank him for everything – for everything here, and in Granada, and in Berlin, and in worlds to come. Facing the hopeless duty

---

\(^5\)Yeah! With uppercase letter and all the sacred it deserves.
of acknowledging him enough I just devote to him any kind word that follows and move on with the hope of finding a place between Camino de Ronda 180 and Pappellalle 5 where I can be touched by his grace again. I love you. I always will.

Marc Marin Webb brought magic. We would summon the Night together and She would happen in front of us and then She would crash both of us at once. Everything is on its place when Marc plays his magic. I am astonished by his brightness and intelligence and feel privileged because he shared it with me. I owe him loads of storms on my brain, rainbows that explode tangential to spaces emptied of void, filled with Almirant Vernon. We are together along the Path, may He take us quite far, may we step on Him and leave a trace, may our feet grow so worn-out. I love you. I always will.

Javier Infante Lourido is the uppermost mythological figure on the Road. He drove me back home throughout Europe striking the hackneyed network of eastern roads and made them shiver. He is a silhouette against the horizon, the dream of Europe, swarms raping a chessboard. This and much more he is, and I had the honor of getting to know him and therefore and for much more I thank him. I also thank him for the picture in the cover of this Master Thesis and so many other beautiful pictures that enclose his whole legend. I love you. I always will.

I must speed up this text because the deadline for my Master Thesis is approaching. The following people shall excuse me because of the space that I can devote to them – shorter that that of the previous guys. I shall list them in a sort of chronological order and wish not to make any sorting regarding importance.

I thank very very much Víctor Gil because of the never ending nights and enlightening discussions. La Ley Innata, Suicide Circus. I enjoy his companion and my ideas bear a lot of his quiet, emblematic teachings. Whoever seeks a trustworthy path must follow Víctor well beyond the Mexico way.

In my summer dreams I can not separate Borja Bethencourt, Conchita Cabrera, Ane Nicolás-Rodríguez, Francisco Fabelo, and Paula Ríos apart. They are a whole and so they will remain in my memory for ever. I thank them for teaching me so many different ways to live, so many shiny colors underneath the rough skin of this life. They also bear magic and know how to use it to tame the nights and shape them fancy, an arcane tarot lifting me from the most waisted corners. Dreams of a homeland, a motherland; well justified dreams of a Regenerationism that never seems to come true. A wish that they never stop thinking, writing. If a righteous Spain is ever to exist, she should listen to these prodigious daughters and sons. To talk, a voice must fly away from a body; to contribute to an idea, its guarantors must be detached from it. Shall these kids be the voice of a country that does not exist. You all became to me more of a homeland than I will ever be able to say of any nation or land. I thank you because of this and because of many more things. So many that they can only be enclosed within the wide space that you stretch with your thoughts, but not within this pages.

I must return to Granada to enlist some people who are still waiting for me there. I look back to those days as a beauty and joyful time, and see the people that follows as those juicy drops that can only be born in that sacred south. I am specially thankful to Samuel Johnson and Sebastiano de Franciscis because
they saved me from being torn apart by a pack of *perroflautas*. To them and also to Jordi Hidalgo, Virginia Domínguez, Leticia Rubio, Alejandro Pinto, and Jose María Martín must and wish I thank so many wonderful nights and tapas. I deeply miss the great discussions, arbitrary theories about everything, the joy of looking at the world with brand new eyes at dusk. I envy your intelligence, sense of humor, the way you just are in the world and how you own it to do with it as you please. I wish a blazing future for you and I wish to participate of it at least from abroad. I can not say these words without having a last one though for Samuel Johnson again, with whom I synchronized many times and once we were about to kiss each other in the mouth.

Also from Granada I must and wish with all my might acknowledge Jose Espín. I must make a stop to tell him that I love him and to celebrate the nights twisted as they are, full of dangers, a scent of sex and dirt, *extremoduro*, the strait of Gibraltar edgy as a razor blade in the hands of a patient with Parkinson’s. This guy is poetry. Not the tacky one, but the one I would be ready to kill and be slain for. The one the Sahara desert whispered to Andalucía many centuries ago, the one that comes with the shit of thousand camels within its dust. This guy is creation as it can only come from destruction, vivid genius as he steps on the pavement of his city. I wish this guy were my brother. I also wish that Ángel Ibáñez were my brother, and also to him I am thankful. I hope I ever have the chance to know him better because I guess the warmth in him; the zoom out of a movie over a land cursed with magic, over a land that is himself. I look forward to meeting you too again.

Back to Berlin I wish to specially acknowledge that other beast which is about to rock the world: Lluís España. I feel so fortunate to have met this guy! By now this can sound repetitive, but I do not care because it is the most honest of the truths. Thank you for filling up with inspiration and wizardry the Night and the Road, for chanting spells to them and have them moving gently as your hand spiders with the chords of your guitar – that masterpiece. I owe you so much that I can just promise to continue these words if I ever get to write a Ph.D. Thesis one day. Also from that days I wish to recall the good memories with Julia Gómez, Clara Madueño, Marta Vidaurreta, and Víctor Mier, and many more! Sorry that I stop here, but I must go on.

Just before moving forward, I must and wish to acknowledge some abstract friends. I deeply thank now and forever the mere existence of Roberto Bolaño, Andrés Neuman, Leon Tolstoi, and Miguel de Cervantes. I thank that they created *2666*, *El Viajero del Siglo*, *War and Peace*, and *El Ingenioso Hidalgo Don Quijote de la Mancha* as only extreme devotees of some religions can thank their gods that they exist. I thank *Extremoduro* for writing *La Ley Innata* thus ripping my life off from me. I thank Pink Floyd for writing *Wish You Were Here* thus bringing me back to life. I also thank all this Masters for the many other genial works they have written and which I enjoy as often as I can.

Unluckily so many friends must be left aside. Time is finite. I would not like

---

6 Literally: *dogflutes*. This is a Spanish term to name wasted guys who pretend to live on the streets but who usually have their wealthy parents to cover up for them just in case and who seek for anything in life but pretending that they are above the rest of the world. Yeah, this is a *perroflauta*. But indeed it was a pack of their dogs what was about to eat me alive.

7 Although obviously they do not.
to yield, though, without quickly listing many of them. To Vera del Estal, to whom I can not make justice in such a brief space, I remain in debt. To Dr. M. de M. A. and Prof. Dr. L. S. D. for shedding such a new life over this world that was fading. To Manu Berlin – sorry that I do not know your name, such a mystery you came and were gone with.

To Tobias Müller! I definitely can not name him without a few more words. I thank him for his joy, of a strange kind to me. This is the cultural shock, I guess. Hope to meet you soon and share more time with you because it is always a pleasure. To his brother Felix and to his – Tobias’s – girlfriend Marlene, and also to their – Felix’s and Tobias’s – cousin Nadine.

I can not forget the Fuentetaja sisters: Natalia, with whom I would love to collaborate one day and from whom I admire her determination and intelligence. I can only wish you the best together with Xurxo, to whom I remain in debt as well. And Raquel, the other Fuentetaja sister, I think we have loads of pending conversations! I wish you the best in your life and thank you for those many good moments together!

To Ayleén, to whom I owe much more than a brief acknowledgment and a few bears.

To Marcita! and to Andrés Dean – how funny is it that I place you together in this text? To Christina Blum, with whom I would like to share more time in Granada or Coruña.

To Lucía García I remain always thankful. I always look forwards to her visits and enjoyed very much the days she spent with her boyfriend Fede and with me in Berlin. Also, I still remain thankful to her parents for saving my life in the center of Europe so long ago!

Of course, I can not finish without naming Jorge Infante, Jorge el Rubio, Fran Arranz, Carlos Lourido, Carol Tekilón Muñoz, and Jose Lourido – this time the sorting is quite arbitrary, but I do not wish to make any distinction between them either. I have left you for the end of this section of the acknowledgments to dedicate you some heartfelt words too. I thank you for the many good memories we have together – they were so necessary!! Along with crazy, old Javi we were able to squeeze that juice from the streets of Coruña, so proud and unconcerned as she is. Thank you for your magic trips to visit me in random places. Hope to relive those good times some day not far from now.

Miscellanea

The elaboration of this Master Thesis has been unacceptably stretched in time, mainly because of surrealistic bureaucratic problems. Whatever the case, I have left Berlin behind with this manuscript still open to modifications. As I met new people – luminous people – in Barcelona I acquired new duties and must further acknowledge the enormous influence that these people are exerting on me, on the way I see the world, and on my personal values. I do not wish to dwell on this point because I wish and will work to have the chance of making justice to them, again, if I get to write a Ph.D. Thesis some day. However, it is fair enough to at least dedicate a few words to them.

I already owe deeply to Ricard Solé. Not only for the great chance he has given to me, but also for bringing up for me some powerful and revolutionary
– from my modest point of view – ideas. I admire the rich and stimulant working environment that he has been able to create; and there I recognize the paramount role of Sergi Valverde, Carlos Rodríguez, Javier Macía, and Bernat Corominas – listed after the anti-alphabetical order of their surnames wishing not to make any distinctions between them. It is a pleasure to share a part of this huge fight in the realm of knowledge with you. I feel privileged well beyond any standard. I also feel privileged and am amazed to see that I will continue this adventure together with Max Carbonell, Nuria Conde, Salva Durán, and Ben Shirt-Ediss – named now in alphabetical order wishing not to make any distinctions as well. If I could only contribute to you with the slightest part of the daily thrill that it is for me to just have lunch in your companion!

With the sense of fulfillment I had because of these last years I could not ever believe that there were anything beyond. But life is full of surprises and gifts and in Barcelona I discovered a new kind of happiness. I must and wish thank to The House at Horta for its mere existence, and with this I come to the following brilliant people: Iván Riba is love, the essence of love! He owns the hearts of the people and can make them beat with the rhythm he considers appropriate. Iván embraces the world and leaves a heavy trace of satisfaction behind him. Marc Marin is old acquaintance of us and there is nothing left I can add here. To Lucía Saavedra I must and wish to thank many things, not only from this time of my life. To her I owe great part of my childhood, the most lucid one and the one I look back to with more satisfaction. Now I would like to thank her for the bravery with which she unveils her talent and I thank the chance to be present at such a discovery. Also, of course, for the thorough and useful corrections that she has made to part of this manuscript. Living with you three is happiness and love and a wish that life shall be immortal, and the certainty that this life of ours will already last for ever. Thank you very much for everything that can be thanked for.

I thank Gonzalo Heredia for suggesting me the use QR codes in this manuscript. Apart from this technical issue, he must already know how short my words will be. I can only thank him that he happened to me. That he decided to happen to me. But the throbbing of my blood makes me wonder what essential Ethos is to be thanked at all: our free-will gambling our lives in a chess match against destiny; Barcelona, a perfect chessboard therefore. What a masterstroke! What a delicate way of moving the threads from Budapest, Berlin to here. My flesh tumbling down and shocked. What words then now?

Again, I am thankful to a lot more of people and I just try to enlist few of them. It would be impossible otherwise. To Carles Martínez and Roi Campos for a few vivid nights that I am looking forward to repeating. Together with them, to the many people who came to Barcelona after Berlin: nice stretch to walk together.

I wish to mention Jorge Mira specially because I still remain in debt to him. He gave me many important opportunities during my career and I admire his discernment and intelligence. It is a pleasure to collaborate with him and I wish to have again the chance in the future.

I would like to acknowledge the great influence exerted onto me during this last year by the Think Tank Chibetín. I enjoy its political views and its robust arguments against the fearsome and dangerous catholic church and other
political institutions that promote values contrary to those from the age of Enlightenment just for the sake of power. I also would like to acknowledge the FROGS – FRont of Galician speaking Scientists as a great and interesting initiative.
Summary

The present Master Thesis deals with the implementation of a successful probabilistic attack to crack a Neural Cryptographic system. Neural Cryptography has been developed during the last decade with great success. Two systems implemented on artificial neural network have shown to be robust enough as key-exchange protocols when tested against different attacks.

Existing attacks usually attempt to break the cryptographic protocols by mimicking it and show low performance. A probabilistic attack attempts to track the probability that the key-exchange protocol is converging towards each possible secret key and is radically different from those attacks tried before on Neural Cryptography. This attack was suggested before, as acknowledged in the body of the present work, but here it is presented the first implementation – to the best of our knowledge, including exhaustive literature – of a probabilistic attack on the two most researched Neural Cryptographic systems.

In one of these systems – namely, the so-called Permutation Parity Machine (PPM) – the described algorithm shows an outstanding performance and allows us to conclude that the system is not safe enough for any cryptographic means. This result was published as a paper in the scientific journal Physical Review E and constitutes the core of the present Master Thesis. The results obtained so far for the probabilistic attack on the other relevant Neural Cryptographic system – the Tree Parity Machine (TPM) – are only of a speculative nature and do not allow to assess the security of this system under such an attack. These scarce results are included as a chapter of the present work with the hope that they can help to further improve any candidate probabilistic attack on TPM-based cryptography in future implementations.
Chapter 1

Introduction

Cryptography and cryptanalysis

Artificial neural networks have got many ways in into the global field of cryptography. They are quite capable systems with astonishing plasticity and sufficient power to solve hard tasks with only few information available. In a manner natural enough the first contact of artificial neural networks is with cryptanalysis: the branch of cryptography that deals with deciphering the secrets rather that building up a code to hide them. (This later application is the one we are concerned with during this work.)

Indeed, artificial neural networks were shown to be universal approximators to arbitrary continuous functions over compact sets [1]. This means that if we want to reconstruct an arbitrary function\(^1\) we can use artificial neural networks and they will, in principle, have a complexity enough as to reproduce the sought target. In reality a potentially infinite number of neurons can be needed to build the right artificial network, so there can be certain drawbacks.

But usually you can approximate a function as much as you wish teaching it to an artificial neural network. This means: presenting examples of the function that the neural network has to learn. This learning is implemented by some algorithm that adapts the network to the required task. Here comes another advantage of these systems: they learn by example with well standardized algorithms that find the right setup to reproduce any sought behavior\(^2\). Artificial neural networks behave as a black box capable of copying a procedure that is repeatedly presented to them, and we do not need to take special care about what is really going on inside the network.

This is very handy for cryptanalysis and reverse engineering in general. To break a password it is great to have some object capable of learning the secret code behind just by presenting examples, and this is exactly what artificial neural networks can do. A different issue is whether the neural network can successfully break this secret code only with the information provided by two agents interchanging ciphered messages, which should not be the case if the cryptographic system used is safe enough. Before arriving to that we can already see—because of the features of artificial neural networks—that the idea is at least worth to try it.

\(^1\)One that is continuous over a compact.

\(^2\)I.e. any sought continuous function over a compact set.
CHAPTER 1. INTRODUCTION

Back to 1995 Sebastien Dourlens [2] describes cryptographic features embedded in an artificial neural network framework – his work goes beyond cryptanalysis since Dourlens also considers some ciphering capabilities that neural systems should be able to implement as well. In [2] it is implemented an attack on the Data Encryption Standard (DES) that is based on neural networks and that learns by example, showing interesting performance. Since then literature accumulates showing the capabilities of artificial neural networks for either ciphering or deciphering codes.

In the present thesis there will only be described implementations of artificial neural networks that were actually used at any point of the work – namely Permutation Parity Machines and Tree Parity Machines. For a more complete treaty about the topic it is recommended to follow any standard course on artificial neural networks or machine intelligence [3, 4].

Before Cryptography: the importance of synchronization

Artificial neural networks result interesting from a cryptanalysis point of view, but our main concern now is that they can also implement key-exchange protocols over a public channel. For this aim it is needed a less trivial feature than the plasticity and power mentioned above: the ability to synchronize.

Synchronization happens all around us. It is a well studied and expected property of loads of systems from the physical to the animal realm and many more. Astonishing examples are bursts of fireflies [5] and similar phenomena in bacteria [6]. It can be easily seen how complex mathematical objects synchronize with the presence of just very weak couplings between them, as it can happen to pendulums hanging from opposite walls. Some chaotic systems can synchronize as well [7–11] and several neural systems can also do so as they shall be instances of chaotic systems [12–17]. Synchronization can be induced in different ways: by the presence of a common external stimulus which might be just noise or because of the interactions between parts of the synchronizing systems. In this last scenario we can differentiate between unidirectional an bidirectional synchronization. In the former case there is a communication flow and synchronizing parts are not symmetric. This leads to master-slave-like structures and some parts of the systems might be regarded as passive while others actively drive the dynamics. In the later case, the synchronizing parts are symmetric and the information flow is similar in all possible directions.

In many systems there is no difference between directional or unidirectional synchronization [13]. This means that the system is able to synchronize, and usually within the same average time, whether if all interacting agents actively generate information and use it to drive the dynamics of each other or if they just listen to the information that others emit. For some other synchronizing systems this is not true [15–24], and two agents which engage in active interaction can synchronize faster or much faster than a passive agent that attempts to track the reciprocal interaction between the former.

---

3 http://www.youtube.com/watch?v=a-Vy7NZTGcs
4 http://www.youtube.com/watch?v=RbaWC2GDYJQ
5 We take a language already from neural systems, where it is easy to identify different artificial neural networks as agents. Here by agent we mean any system capable of synchronization: a pendulum, a chaotic system of coupled differential equations, certain neural networks themselves, etc.
In neural systems capable of synchronization, uni- and bidirectionality brings in the very important concept of mutual learning vs. learning by example. A system for which a passive member can not learn by example as fast as the active components mutually learn from each other is a good candidate for neural cryptography because an attacker, usually a passive agent, would need more resources—in terms of time or available information—to cope with the synchronizing parties.

Cryptography with artificial neural networks

This ability to synchronize translates into the possibility of having two partners equipped with an artificial neural network each who can establish a secret code sending messages over a public channel. Furthermore, if mutual learning synchronizes neural networks faster than an external, passive party can do by learning only by examples then the secret code would be hard to guess by any eavesdropper. This fully enables artificial neural networks as candidate cryptographic devices [25]. Researching this possibility has been a fruitful field during the last decade [15–24, 26–33].

The so-called Tree Parity Machine (TPM) [15,16,18] –a bilayer feedforward artificial neural network– has been the dominating device for neural cryptography during the last decade. The key-exchange protocol based on synchronizing TPMs has repeatedly proven to be robust against different kind of attacks [22, 26, 30, 33]. A binary version of the TPM called Permutation Parity Machine (PPM) was introduced in 2009 [17] and was latter researched to find that it is even more robust than the original TPM against classic attacks⁶ [24].

---

⁶We use the term classic to denote those attacks involving or inspired by the learning mechanism itself. We shall be naming attacks which were previously envisioned for TPMs or that make use of the learning capabilities of artificial neural networks with the same structure as those attacked. There should not be any confusion because only the probabilistic attack is not labeled as classic within this work. It is not intended to establish a categoric definition of
Probabilistic attacks were suggested in [27] for neural cryptography in general. These should use mathematical tools—usually from the field of statistical mechanics—to calculate some underlaying probability distributions that the secret keys generated by the cryptographic protocol take this or that shape. Such attacks do not necessarily make use of the learning abilities of neural networks but put a stronger stress on finding efficient mechanisms to extract information from the few public messages between the synchronizing parties. However, artificial neural networks do learn from each other and this must also be taken into account by a candidate probabilistic attack to avoid that the learning mechanism of neural networks erases the information that the attacking algorithm is able to compile.

Even though probabilistic attacks can be easily envisioned we did not find any trace of such an algorithm in the literature since they were suggested [27] until the work that constitutes the core of this Master Thesis was released [34]. In [34] and along this Thesis a probabilistic attack is described and its success against the PPM-based key-exchange protocol is shown. To the knowledge of the author, this is not only the first probabilistic attack implemented upon neural cryptography, but also the first successful attack on any neural-based key-exchange protocol. This Thesis is extended with some preliminary results of the algorithm against TPM-based cryptography; but the considerations included are so vague that it can not be made any claim about the success or failure against the TPM synchronization algorithm yet, neither in an optimistic nor in a pessimistic direction.

---

what is and what is not classic in this field: it is sought just a working nomenclature.
Chapter 2

The Permutation Parity Machine

The Permutation Parity Machine (PPM) was introduced in 2009 by Reyes, Zimmermann and Kopitzke [17] and it was later used as a mean for neural cryptography which proved to be more robust against some classic attacks than the previous Tree Parity Machine (TPM) [24].

Because of this, PPMs constitute an interesting working bench to try out new attacks. In this chapter it is described how a PPM works and how well do simple and geometric attacks work on them. To get a better picture of the robustness of such machines the performance of these attacks is compared to the performance of the same attacks on TPMs. It will be seen how PPMs outperform TPMs thus revealing a greater fitness for cryptography.

Most of the issues dealt with in this chapter were investigated by Reyes, Zimmermann, and Kopitzke in their works on PPM synchronization and PPM-based neural cryptography [17,24].
2.1 Computing with Permutation Parity Machines

PPMs are artificial, feedforward neural networks consisting of 2 layers with an output unit in the most forward layer and \( K \) hidden units in the most backward layer. Each hidden unit has got an independent receptive field. The fact that the receptive fields are independent from each other prevents correlations between the outcomes of different hidden units, otherwise the amount of information required to completely specify the state of the system would be lower and the system would become less appropriate for cryptography.

The receptive field of each hidden unit consists of \( N \) bits. The input bits to unit \( j \in [1,K] \) are provided by \( x_j = (x_{1,j},...,x_{i,j},...,x_{N,j})^T \) and the collection \( X = (x_1,...,x_j,...,x_K) \) containing \( N \cdot K \) bits \( (x_{i,j} \in \{0,1\}) \) is known plainly as the input of the PPM. Concisely speaking, we should refer \( x_j \) as the input vector of the \( j \)-th hidden unit and \( X \) as the input matrix of the PPM.

PPMs use \( N \cdot K \) weights \( w_{i,j} \) to link the input to its final output. These weights are bits as well \( w_{i,j} \in \{0,1\} \) and they are only known to each PPM—they are private—but they can become the same in two different PPMs which synchronize using the public procedure described in [17] even if the weights remain hidden all the time. We shall return to the weights later to describe how they are set up in a PPM. By now let us focus on how a PPM computes once arbitrary weights have been provided.

Each input bit \( x_{i,j} \) has associated a weight bit \( w_{i,j} \in \{0,1\} \). From here, weight vectors \( w_j = (w_{1,j},...,w_{i,j},...,w_{N,j})^T \) can be considered, as well as a weight matrix \( W = (w_1,...,w_j,...,w_N) \) if necessary. Let us briefly note that there are \( 2^{NK} \) possible different weights, as well as \( 2^{NK} \) possible different inputs. The size of the weight space will be important later, when describing the probabilistic attack.

The weights together with the input completely determine the results of the computations performed by a PPM. At an intermediate stage it is solved a mapping from the input to the internal state \( \sigma_j \in \{0,1\} \) of each hidden unit. Firstly, each unit performs the one-by-one XOR operation on the elements of the corresponding input and weight vectors to build up its vector local field \( h_j \):

\[
 h_j = x_j \oplus w_j = h_{i,j} = x_{i,j} \oplus w_{i,j}. \tag{2.1}
\]

Then a scalar local field is derived from here as the sum of ones in the vector local field:

\[
 h_j = \sum_{i=1}^{N} h_{i,j}. \tag{2.2}
\]

The hidden unit becomes active \( (\sigma_j = 1) \) if the majority of the bits in the vector local field are 1. It is considered inactive \( (\sigma_j = 0) \) otherwise:

\[
 \sigma_j = \Theta\left(h_j - \frac{N}{2}\right), \tag{2.3}
\]

\( \Theta(x) \) representing a Heaviside step function:

\[
 \Theta(x) = \begin{cases} 
 0 & \text{for } x \leq 0, \\
 1 & \text{for } x > 0
\end{cases}. \tag{2.4}
\]
2.1. COMPUTING WITH PERMUTATION PARITY MACHINES

We name $\Sigma = (\sigma_1, ..., \sigma_j, ..., \sigma_K)$ to the ordered collection of the internal states of the hidden units. Throughout this text we shall also call this object the internal representation of the PPM. Let us note that there are $2^K$ possible internal representations so that the computations performed in the most backward layer represent already a reduction from the $2^{NK}$ possible different weights or different inputs of the system to $2^K$.

It is important to remind that all these processes remain private so an external party would never know the values of the local fields nor the internal states of the hidden units. The fact that these internal states are unknown to any external agent is also decisive for the good functioning of the system as a cryptographic mean.

The next computational step is the integration of the states of the hidden units at the output unit in the most forward layer. To do so it is considered the parity of the states of the hidden units. This constitutes the public output $\tau$ of the whole neural network:

$$\tau = \bigoplus_{j=1}^{K} \sigma_j. \quad (2.5)$$

In the line of a previous comment let us note that there are two possible outcomes. We can see how the degeneracy of the variables at the different computational levels goes from $2^{NK}$ at the weights or at the input, down to $2^K$ at the internal configurations and down to only 2 at the output level. Thus, given an input, of all the $2^{NK}$ possible different weights only $\sim 2^{NK}/2 = 2^{NK-1}$ generate each one of the possible values of $\tau$ as an output. Also, only $\sim 2^{K-1}$ of the possible internal representations are compatible with a given $\tau$.

To complete the steps described so far the PPM must be equipped with the weights, as pointed out before. To select the weights of the hidden units the PPM makes use of a pool of bits $s$ called the state vector and is aided by a matrix $\pi$. Both elements will be described in the next paragraphs.

The state vector $s$ is a pool where bits are stored. These bits become at some point the weights of the PPM so the state vector must be kept private if we wish to preserve the weights hidden—as we do for cryptography. In the next section we will see that the components of $s$ are either random bits or bits obtained after some computational process depending on the synchronization stage of two PPMs. By now let us just assume that each PPM has associated one such a pool of bits $s = (s_1, ..., s_i, ..., s_G)$, $s_i \in \{0, 1\}$, notwithstanding of how these bits were generated.

The size of the pool is taken arbitrarily larger than the number of input bits: $G \gg N \cdot K$. Using a $G$ smaller than $N \cdot K$ or larger but close to it would make that many weights correlate, as we will see right ahead. This would be similar in some way to using non-independent receptor fields: correlations between the hidden units would emerge and the fitness of PPM for cryptography would be lowered. Thus, although $G \gg N \cdot K$ is not a requirement for the PPM to compute it will be assumed always as granted as it is the best to use the system for cryptography.

A quick remark: the number of possible different state vectors is $2^G \gg 2^{NK}$. The state vector constitutes the key in PPM-based cryptography so $G$ is directly
related to the number of keys that a PPM-based key-exchange protocol can generate and to the robustness of this protocol against an attacker.

The state vector is linked to the weights by a matrix $\pi$. The matrix $\pi$ contrary to the state vector –thus contrary to the weights– is usually a public variable. It is an $N \times K$ matrix and its entries take values $\pi_{i,j} \in \{1, G\}$. The $(i,j)$-entry of $\pi$ determines what bit from $s$ must be used as a weight for the $j$-th input of the $i$-th hidden unit: $w_{i,j} = s_{\pi_{i,j}}$.

There are not any constrains about how these entries are determined but usually they will be uniform random numbers between 1 and $G$. Since $\pi$ links the state vector to the weights of the hidden units, the usage of other thing than uniform random numbers for the entries of $\pi$ will again introduce undesired correlations between different weights reducing the effectiveness of the system for cryptography. Now we can also see that the correlations announced before for small $G$ arise when it is very likely that the same bit is used as a weight for many inputs, thus the requirement $G \gg N \cdot K$.

Before computing, the PPM must be provided a matrix $\pi$. If the PPM preserves the same matrix $\pi$ for every computation the weights will not change unless the state vector itself is modified. Usually, and at least for the cryptographic purposes described in this work, a new matrix $\pi$ is provided together with the input vector at the beginning of each computation. This means, effectively, that the weights of the hidden units are set up with bits drawn by random from the state vector; and that the weights change each time that the PPM computes.

The procedure since the $\pi$ matrix and the input are provided; through the update of the weights, the local fields, the states of the hidden units; until the computation of the output $\tau$ of the PPM is known as an inner round. Many such computations complemented with an appropriate update rule for the state vector to drive two different PPMs towards convergence allows them to synchronize. In the following it is described the synchronizing procedure over a public channel introduced in [17].
2.2 Synchronizing two Permutation Parity Machines

Let there be two PPMs $A$ and $B$, each of them with all the elements described before (weights, outputs, etc). We shall label the different elements with superscript indexes –e.g. $s^A$ and $s^B$ for the state vectors. In [17] it was shown that these PPMs can synchronize over a public channel keeping their state vectors private. This means that after applying the algorithm described in [17] and in the following paragraphs $A$ and $B$ will have the same state vector while only the outputs $\tau^A$ and $\tau^B$, the input vectors $x_{i,j}$, and the matrices $\pi$ need to be public at some point.

The synchronization level between $A$ and $B$ is measured with the overlap between their state vectors as introduced in [17] and later used in [24,34]. This is defined as:

$$\rho_{AB} = \frac{G - d_H(s^A, s^B)}{G}$$

with $d_H(x, y)$ being the Hamming distance between two vectors of bits $x$ and $y$. $\rho_{AB} = 1$ if $s^A = s^B$ and $\rho_{AB} = 0$ if $s^A = \neg s^B$, where the negation must be taken bitwise. Sometimes both cases can be regarded as a synchronized state, as it will be seen soon. Usually, prior to any synchronization $A$ and $B$ are initialized with uniform- and randomly generated bits, so the initial average overlap between pairs of PPMs is 0.5.

For synchronization, PPMs complete a series of inner rounds packed up in so-called outer rounds that will be defined ahead. During these rounds of computations $A$ and $B$ will exchange some information and modify their state vectors with some rule described below. At the beginning of an inner round, input bits $x_{i,j}$ and $\pi$ matrices are generated by drawing uniformly distributed random bits –for the input– or numbers between 1 and $G$ –for $\pi$. They are public and both PPMs use them during one inner round to set up their weights and to compute their output. Thus the weights are taken from the same positions of $A$’s and $B$’s state vectors but they will not usually be the same prior to any synchronizing step because the state vectors are initialized randomly. Accordingly, the internal representations of the two PPMs shall not be the same either. Taken perfectly randomized state vectors we get $\sim 0.5$ chance that the output of either machine is 0 or 1 –thus $\sim 0.5$ chance that their outputs match. As $A$ and $B$ synchronize it becomes more probable that both machines get similar bits at their weights$^1$ thus it becomes more probable that both machines produce the same output.

A suitable learning rule must be used to update the state vectors. This rule modifies the state vectors in such a way that the probability increases that $A$ and $B$ get similar weights$^2$ during each inner round. We speak about a learning rule as a reminiscence from those neural systems that inspired PPMs in a first place. Sticking to this learning interpretation of the update rules we say that two synchronizing PPMs mutually learn from each other. This is of great importance.

$^1$Or exactly the negation in some cases, as it will be seen

$^2$Again, in some cases getting negated weights can be regarded as a synchronized configuration too.
when considering a third party learning from two synchronizing PPMs (trying
to break them, in a cryptographic language), but unable to make the machines
learn from her. As it is exhaustively shown in the literature [15–24], mutual
learning vs. learning by example is a paramount feature that enables Neural
Cryptography.

In [17] Reyes and Zimmermann show how the following rule works to achieve
synchronization: After each inner round if the two machines agree on their
output ($\tau^A = \tau^B$) they save the state of the first hidden unit $\sigma^A_1$ and $\sigma^B_1$ in a
buffer variable: $s_{a/b}(t) = (\sigma^A_1(0), ..., \sigma^A_1(t))$, where $t$ is an index over the
inner rounds at which $\tau^A = \tau^B$. This buffer variable is private to each PPM. Of
course, tracking the first hidden unit is an arbitrary choice. Whenever $s_a$ and
$s_b$ reach size $G$ each PPM substitutes its state vector by this buffer variable:
$s_a(T) \rightarrow s_a(T + 1) = s_b(T)$, where $T$ is an index over successive
updates of the state vectors. After the update of the state vector the buffer
variable is cleared and the process begins anew. The inner rounds from the first
one to the first update of the state vectors, or from one state vector update to
the next one are grouped up as an outer round – thus $T$ is an index over outer
rounds. Outer rounds are repeated until convergence – i.e. $s^A = s^B$.

Now it is clear that $A$ and $B$ agree to learn from each other whenever
$\tau^A = \tau^B$, thus mutual learning. An attacker can modify its internal variables whenever
$\tau^A = \tau^B$ or $\tau^A = \tau^E$ if she wishes, but she can not force $A$ nor $B$ to modify
their private variables. An attacker can only learn from examples. The fact
that an attacker can not induce mutual learning is the key point for neural
cryptography, as pointed out before.

In [17, 24] it is calculated the probability that such an update rule yields
state vectors with a larger overlap $\rho^{AB}(T + 1) > \rho^{AB}(T)$ given an overlap
at $T$. The process is described as a Markovian chain which can only lead to
synchronization in average. For an even number of inputs $N$ this rule drives
the state vectors towards a parallel state $s^A = s^B$. It can be shown how the
probability of remaining with equal state vectors is 1 once they have converged.
It can also be shown that if $s^A = -s^B$ (thus $\rho^{AB} = 0$, which is the minimum
possible) and again with even $N$ then the probability exists that the state vectors
increase their overlap arriving to $s^A \neq -s^B$ while $s^A \neq s^B$. In this case we can not
considered that the PPM were synchronized. Opposed to this, for an odd
number of inputs both possibilities can be regarded as synchronized (we shall
then talk about parallel or anti-parallel synchronization). It can be shown again
that if $s^A = s^B$ the probability that the state vectors diverge in future outer
rounds is exactly zero and in this case it also happens that if $s^A = -s^B$ the
probability that the state vectors increase their overlap is zero, thus $\rho^{AB} = 0$ is
a fixed point of the system. An anti-parallel alignment of the state vectors
would produce in this case similar outputs in both machines in every inner round
and new state vectors would still be anti-parallel upon update after complete
outer rounds. An extra step can be taken to force a parallel convergence using,
for example, a generator of pseudo-random numbers with the state vectors as
a seed. This does not need an extra exchange of information. For odd $N$ the
number of possible secret keys is reduced in a factor 2 with repsect to cases with
even $N$. Also, if $N$ is odd convergence is usually slower and the setup is not so
fit for cryptography.

From a dynamic point of view we can say that the system has got 1 or 2
fixed points if \( N \) is even or odd respectively. These points have got an extremal overlap and are not affected by fluctuations, thus a system in one of these fixed points remains there even considering that the inputs and \( \pi \) matrices are random. We will see that for an attacker implementing a classic attack a new and stable fixed point appears on its dynamics for an overlap \( 0 < \rho^* < 1 \) around which the dynamics will fluctuate due precisely to the randomness of the input, while \( \rho = 0 \) and \( \rho = 1 \) will be unstable fixed points which can only be reached thanks to these fluctuations [24].

During the whole process described before, including inner and outer rounds, the only variables that must be available to both PPM are the input matrices \( X \), the matrices \( \pi \) to correctly allocate bits form the state vectors into the weights, and the output of the machines \( \tau^A \) and \( \tau^B \). All the remaining elements (the state vectors \( s^A(B) \), the buffer variables \( s^A(B)_b \), the weight matrices \( W^A(B) \), and the internal representations \( \Sigma^A(B) \)) can be private all the time and only known to each PPM. This synchronization while retaining all those variables unknown to other parties is what enables PPMs to work as cryptographic means. The state vectors would play the role of a secret key and a system of two synchronizing PPMs can be used as a key-exchange protocol over a public channel—since all other variables can be made public.
2.3 Classic attacks on synchronizing PPMs

2.3.1 Introduction to attacks: success probability and scaling complexity

The question arises immediately whether it is safe enough or not to share the input matrices $X$, the $\pi$ matrices, and the outputs $\pi^{A(B)}$ of two synchronizing PPMs. To answer this question, the system had to be tested under the attack of a third party. The results obtained by Reyes and Zimmermann [24] when using some classic attacks are explained in the following, together with a description of the attacks and the comparison of these results with those from Tree Parity Machines.

Now and on we suppose the existence of an eavesdropper $E$: a third party implementing her own Permutation Parity Machine. We will identify her state vector as $s^E$ and label her other variables accordingly with the super-index $E$.

The aim of the eavesdropper is not only to guess the state vector of one of the synchronizing parties ($A$ is taken as target by default), but also to do it before synchronization is complete. This would invalidate the current protocol as it would not be secure anymore. Intercepting one synchronization process would not deem the key-exchange protocol as unsafe, though. We seek a more ambitious aim. To show that a key-exchange protocol is not safe two requirements must be met: firstly it has to be shown for some setup of the cryptographic system that the probability of violating a synchronization process is high enough; secondly it has to be shown that this high probability of breaking the system remains high whatever the setup (in our case: whatever parameters $K$, $N$, $G$) chosen. A few more words about these assessments follow.

The former condition can be checked calculating in what proportion of cases an attack is capable of intercepting the communication before the system synchronizes. We assume that a key-exchange has been intercepted if $s^E = s^A$ before $s^A = s^B$. It can also be computed the average breaking time $\langle t_b \rangle$, which is the average time of convergence of $s^E$ towards $s^A$, and compare it to the average synchronization time $\langle t_s \rangle$ of $s^A$ and $s^B$. (Any of them can be measured in inner or outer rounds.) The system will be found to be unsafe for some setup $N$, $K$ and $G$ if the average breaking time is shorter than the average synchronization time $\langle t_b \rangle < \langle t_s \rangle$. This should be directly translated into a high success probability for the attack.

But this means only that the protocol can not be trusted for some setup. To completely invalidate a cryptographic system it has to be invalidated for every possible $N$, $K$ and $G$. This would mean to try the system for every possible setup. In out case this is impossible since the parameters defining the system can take infinitely many possible values: $N = 1, \ldots, \infty$; $K = 2, \ldots, \infty$; and $G = N \cdot K, \ldots, \infty$.

To overcome this, we must research how the complexity of the synchronizing algorithm scales compared to the complexity of the attack. The three parameters mentioned before define a complexity level, and the complexity of the different algorithms—the synchronization procedure and the attack—grows as $N$, $K$ or $G$ grow. A common result in a valid cryptographic system is that the complexity of the key-exchange protocol grows in a polynomial fashion with some parameter while the complexity of the attack grows exponentially. This
2.3. CLASSIC ATTACKS ON SYNCHRONIZING PPMS

would mean that any two parties can choose an arbitrary complexity level high enough and a third party would need exponentially more resources—e.g., exponentially more public information—to achieve the same performance and thus violate the key-exchange. This is usually reflected in some variable such as \( t_s \) and \( t_b \) which present the corresponding scaling (polynomial or exponential) when plotted against some of the parameters \( N, K \) or \( G \).

Back to PPMS, although we can not test the system for all possible combinations of \( N, K \) and \( G \); to show that the system is not safe it would suffice to show that the complexity of the synchronization algorithm grows faster than the complexity of the attack. This would mean that, no matter what setup the two synchronizing parties would choose, a third party would in average be able to break the system with less resources than available for synchronization. In the following it is illustrated the work by Reyes and Zimmermann [24] which shows how the PPM-based key-exchange protocol is a robust protocol against some classic attacks in Neural Cryptography.

2.3.2 Simple and geometric attacks

In a simple attack \( E \) stores the internal state of her first hidden unit in a buffer variable whenever \( \tau^A = \tau^B \) without taking into account its own output. It tries to track \( A \)'s and \( B \)'s behavior with the minimal information available. How weak this attack is can be better understood when compared with the geometric attack. In a geometric attack \( E \) applies the same strategy as before, only now it takes into account its own output and internal representation which provides some information about how well \( E \) is performing. In the geometric attack if \( \tau^A = \tau^B = \tau^E \), \( E \) stores the state of her first hidden unit in the buffer. If \( \tau^A = \tau^B \neq \tau^E \), \( E \) knows for sure that at least one of her hidden units computed wrong and she will try to correct this to minimize the possibility of storing a wrong value in the buffer.

In [24] both attacks are analyzed as Markovian processes considering the probability of getting a larger overlap \( \rho^{AE}(T+1) > \rho^{AE}(T) \) (i.e. the probability that the attacks perform converging steps between outer rounds) given the overlap in \( T \). Their results show for the simple attack that synchronization is possible only through fluctuations because the system has got a fixed point at \( \rho^{AE,*} < 1 \) (if \( N \) is even and correspondingly two fixed points \( 0 < \rho^{AE,-} < \rho^{AE,+} < 1 \) if \( N \) is odd); and, consequently, that the probability of success of such an attack is negligible.

When \( E \) applies the correction of a geometric attack she attains only little improvement. Synchronization of \( A \) and \( E \) is again only possible through fluctuations from the fixed point \( \rho^{AE,*} < 1 \) (for even \( N \) or from the fixed points \( \rho^{AE,-} \) and \( \rho^{AE,+} \) if \( N \) is odd), which can not be moved to \( \rho^{AE,*} = 1 \) (correspondingly \( \rho^{AE,-} = 0, \rho^{AE,+} = 1 \)) with this attack alone. In [24] it is carefully considered the probability that a hidden unit will produce a wrong result depending on the local fields \( h_i \) whenever \( \tau^A \neq \tau^E \), and this information is used to flip the state of one of the hidden units before storing anything in the buffer—on this consists the correction introduced by the geometric attack. Many cases can happen: the first hidden unit might compute wrong and be corrected, thus approaching \( s^{E} \) to \( s^A \); it might compute alright and \( E \) shall correct a different unit, again approaching \( s^{E} \) to \( s^A \); it might compute wrong but \( E \) may incorrectly correct a different unit, thus diverging \( s^{E} \) from \( s^A \); and it might compute
alright but incorrectly be corrected by $E$, again diverging $s^E$ from $s^A$. With all these considerations taken into account it can be shown [24] that also for the geometric attack the probability of success is very low and can be made negligible by increasing the complexity of the problem –i.e. using larger $K$, $N$ or $G$.

The performance of these attacks can be compared to the performance of similar attacks on Tree Parity Machines [23]. We find that with parameters which keep the complexity quite low –low $N$, $K$ or $G$– the probability that any of these attacks will succeed on synchronizing PPMs is several orders of magnitude lower than the probability that they will succeed on a TPM. Indeed, taking cases favorable to the performance on TPMs, the performance on PPMs is still up to 2 orders of magnitude lower. In the worst scenario for TPMs, PPMs can be made up to 20 orders of magnitude safer within the range of values for the parameters studied in [24] and [23].

Considering these results, simple and geometric attacks are never a threaten to PPM-based key-exchange protocol. This protocol seems promising in strengthening neural cryptography so far and makes the system a more interesting target for new attacks.

2.3.3 Majority and genetic attacks

In [24] they are considered majority and genetic attacks with a direct inspiration from those implemented for TPMs. Both attacks make use of an ensemble of PPMs in different ways:

A majority attack considers the outputs of the PPMs in an ensemble of fixed size. In a first step a geometric attack is implemented in each and every machine. After any corrections required all the machines have the same output\(^3\). The most frequent internal representation is then selected to update the buffer. This buffer becomes later on the state vector, so after only one outer round this attack would effectively become a geometric attack because only the majority internal representation is used to fill the buffer. To avoid this, only those PPMs which have got $\tau^E \neq \tau^A$ before correcting the hidden units make use of the majority vote. In [24] it is shown how the overlap among the ensemble of attacking machines still increases after a few outer rounds so the majority attack becomes effectively very similar to a geometric attack.

For the genetic attack, [24] consider a population upon which mutation and selection of fittest individuals act. Departing from a randomized PPM, whenever $\tau^A = \tau^B$:

1. *Mutation*: $2^{K-1}$ new machines are created to reproduce the $2^{K-1}$ possible internal representations which agree with the desired output. This is done only if the population consists of less than $M/2^{K-1}$. If there are more individuals, the next step is applied:

2. *Selection*: A fitness function is assigned to every PPM based on the number of outputs $\tau^A$ that the attacking PPM was able to guess. Only the fittest individuals are selected.

\(^3\)Indeed: flipping the state of a hidden unit according to the geometric attack flips the output of the PPM as well.
2.3. CLASSIC ATTACKS ON SYNCHRONIZING PPMS

Individuals used in this attack are updated according to the corresponding learning rule.

As the authors argue in [24], the attack just described is more suitable for a TPM. There, the effects of the learning rule on the ensemble can be assessed right ahead while for PPMs it is necessary to wait a whole outer round before any result can be observed.

Before implementing the probabilistic attack that constitutes the core of the present work a different approach was taken to develop a genetic attack. This approach makes a more intensive use of the tools from genetic algorithms: not only mutation and selection, but also mixing of existing solutions come into play. The proposed attack considers a population of candidate state vectors which are assigned to a PPM and it uses as a fitness function the number of outputs from $A$ that each state vector guesses correctly from one whole outer round. A fraction of the members are selected after their fitness. Then they are mixed to regenerate the population and mutation is applied upon them with a certain probability. This process can be repeated many times using a same outer round until the population has extracted as much information as possible from it. This attack consumes physical time but it does not require more information than that provided by the synchronizing PPMs: it can be performed off-line. This variation of the genetic attack was promising for machines with only one hidden unit, but some difficulties arose for synchronizing PPMs with more hidden units. The attack lost its interest also because of the success of the probabilistic attack and thus its performance is not reported at all in this work. The safety of PPMs as cryptographic means remains thus intact so far.

Summing up: a majority attack is virtually reduced to a geometric one when used upon PPMs and the genetic attack considered in the literature is far from fit to be applied on PPMs; although a more adequate genetic attack can be envisioned.

After trying the classic attacks described in this chapter the PPM-based key-exchange protocol remains safe beyond any reasonable doubt and it is many orders of magnitude more robust than previous Neuro-Cryptographic means, at least against those attacks which were analyzed numerically and thoroughly enough. Once again: the current key-exchange protocol constitutes a very interesting target to challenge Neural Cryptography with new attacks, and this is what is done in the following chapter where a successful probabilistic attack is described and its performance is analyzed numerically and with care.
CHAPTER 2. PERMUTATION PARITY MACHINE
Chapter 3

Successful attack on PPM-based neural cryptography

3.1 Statement about the present chapter

As it was announced before, in this chapter it is described a probabilistic attack that has succeeded in violating the PPM-based key-exchange protocol for many sets of parameters and which has proven an adequate scaling of its complexity. Therefore we can claim that the protocol is not safe at all for cryptographic means.

The main contributions presented in the current work are contained in the scientific article authored by me and Dr. Andreas Ruttor (in this exact order, as it stands in the original paper) called Successful attack on PPM-based neural cryptography published in Physical Review E 85, 025101 (2012) [?]. According to the regulations of the International Master Program Computational Neuroscience [sic] of the Bernstein Center for Computational Neuroscience Berlin (BCCN): "For the format of the MT [Master Thesis] the following regulations apply: accepted peer-reviewed journal papers are accepted as MTs, provided that they are complemented by an appropriate general introduction"\(^1\). In accordance with this, the aforementioned scientific article is reproduced in the following. Hence, it must be specially acknowledged the participation of Dr. Andreas Ruttor in the composition of the text of this chapter.

Notwithstanding this statement, from the original article they have been omitted the names and affiliations of the authors, as they correspond to the names and affiliations of the author of the current Thesis and of the main supervisor of the current thesis; the PACS number; and the bibliography, which has been grouped up with the bibliography of the rest of the text –this might cause some rearrangement of the references with respect to the original work. Also, the text has been adapted from the format of the peer-reviewed journal (e.g. from double column) to the format of this Thesis: text is composed in only one column and the size of the figures has been adapted.

\(^{1}\)Extracted from: http://www.bccn-berlin.de/
Because the original work is faithfully reproduced below some parts might result redundant with previous chapters of this thesis. The author apologizes for this.
3.2 Successful attack on PPM-based neural cryptography

Abstract

An algorithm is presented which implements a probabilistic attack on the key-exchange protocol based on Permutation Parity Machines. Instead of imitating the synchronization of the communicating partners the strategy consists of a Monte Carlo method to sample the space of possible weights during inner rounds and an analytic approach to convey the extracted information from one outer round to the next one. The results show that the protocol under attack fails to synchronize faster than an eavesdropper using this algorithm.

Interacting feed-forward neural networks can synchronize by mutual learning [13, 14]. If two networks A and B are trained with examples consisting of random inputs and the corresponding output of the other one, their weight vectors converge. In the case of Tree Parity Machines (TPMs) this mutual synchronization of A and B requires less examples than training a third network E successfully [15,19–22]. Based on this effect a TPM-based neural key-exchange protocol has been developed [25,28,29,32] and shown to be useful in embedded devices [35,36] as well as sufficiently secure against several attacks [22].

Recently, a variant of neural cryptography has been presented by [24] which uses Permutation Parity Machines (PPMs) [17] instead of TPMs. This change increases the robustness of the key-exchange protocol against the attacks which have been tried on the TPM-based algorithm before [26,30,32]. However, it also reduces the number of possible values per weight from $2L + 1 \geq 3$ to 2, so that other attacks become more feasible. This is especially true for the probabilistic attack, which has been suggested by [27], but not implemented up to now. We have used this idea and developed an attack method specially suited for PPM-based neural cryptography. In this paper, we describe our attack and present results indicating its success.

A PPM is a neural network consisting of two layers: there are $K$ hidden units in the first layer, each of which has an independent receptive field of size $N$; and only one neuron in the second layer. Its $KN$ inputs $x_{i,j}$ with indices $i = 1, \ldots, N$ and $j = 1, \ldots, K$ are binary: $x_{i,j} \in \{0,1\}$. In order to simplify the notation they are combined into input vectors $x_j = (x_{1,j}, \ldots, x_{N,j})^\top$ or the input matrix $X = (x_1, \ldots, x_K)$ where appropriate.

The weights $w_{i,j}$ are selected elements from the state vector $s$ of the PPM, which consists of $G \gg KN$ elements $s_i \in \{0,1\}$. For that purpose a matrix $\pi$ of size $N \times K$ containing numbers $\pi_{i,j} \in \{1, \ldots, G\}$ is used, so that $w_{i,j} = s_{\pi_{i,j}}$. The weight vector $w_j = (w_{1,j}, \ldots, w_{N,j})^\top$ then determines the mapping from the input vector $x_j$ to the state $\sigma_j \in \{0,1\}$ of the $j$-th hidden unit. First, the vector local field $h_j$ is calculated as the one-by-one logical XOR operation

$$h_j = x_j \oplus w_j \implies h_{i,j} = x_{i,j} \oplus w_{i,j} \quad (3.1)$$

of the bits in $x_j$ and $w_j$. Then the unit becomes active, $\sigma_j = 1$, if the majority of elements in $h_j$ is equal to 1, otherwise it stays inactive, $\sigma_j = 0$:

$$\sigma_j = \Theta \left( h_j - \frac{N}{2} \right), \quad (3.2)$$
where

\[ h_j = \sum_{i=1}^{N} h_{i,j} \]  \hspace{1cm} (3.3)

denotes the scalar local field and

\[ \Theta(x) = \begin{cases} 0 & \text{for } x \leq 0, \\ 1 & \text{for } x > 0 \end{cases} \]  \hspace{1cm} (3.4)

is the Heaviside step function. Finally, the total output of the PPM is calculated as the binary state \( \tau \in \{0, 1\} \) of the single unit in the second layer which is set to the parity

\[ \tau = \bigoplus_{j=1}^{K} \sigma_j \]  \hspace{1cm} (3.5)

of the hidden states \( \sigma_j \).

When implementing the synchronization task, two PPMs (A and B) designed with the same settings (i.e. with same \( N, K, \) and \( G \)) are provided. The synchronization will succeed after several inner and outer rounds, which are described below.

For each inner round, the elements of the matrix \( \pi \) and the input vectors \( x_j \) are drawn randomly and independently from their corresponding value set. These quantities are provided publicly to all PPMs, which includes even an attacker E. Then both A and B compute their outputs \( \tau^A \) and \( \tau^B \) and if they agree \( (\tau^A = \tau^B) \), they store the state \( \sigma^A_1 \) and \( \sigma^B_1 \) of their first hidden units in a buffer, which remains private for each PPM.

Thus, there are: public and common input vectors \( x_j \) and the \( \pi \) matrix; public, but not necessarily equal outcomes \( \tau^A \) and \( \tau^B \); private, not necessarily equal state vectors \( s^A \) and \( s^B \); private, not necessarily equal states of the hidden units \( \sigma^A_j \) and \( \sigma^B_j \).

The inner rounds are repeated until the buffers where \( \sigma^A_1 \) and \( \sigma^B_1 \) are stored reach size \( G \). Then, an outer round is completed and each buffer becomes the new state vector in the corresponding machine, substituting the old one. The dynamics of the PPMs are such that after each outer round the state vectors \( s^A \), \( s^B \) tend to be more alike, eventually reaching full synchronization \( s^A = s^B \). The synchronization time \( t_s \) measured in the number of outer rounds is a random variable, as it depends on randomly chosen initial conditions and inputs. However, its mean value rises in a polynomial fashion with increasing size \( N \) of the input vector as well as growing size \( G \) of the state vector \( s \) [17].

Reported previous attacks on PPMs tried to mimic the behavior of the synchronizing networks using a single machine or an ensemble [24]. They showed poor performance in guessing \( s^A \) correctly. Namely, for the attacks on PPMs with \( K = 2 \) and \( G = 128 \) analyzed in [24] the probability of success did not exceed \( 10^{-5} \).

In the following, we present the description of a new attack strategy. It does not pursue to mimic the synchronizing process, but to first guess the state vector of A (or B) during an outer round and consecutively to reproduce A’s (or B’s) behavior during the given round so that a fair guessing of the bits stored in the buffer and the subsequent \( s^A \) for the next outer round can be done.

Some notation is introduced now. The synchronizing parties A and B are eavesdropped by a third agent E, which implements its own PPM with state
vector $s^E$ and output $\tau^E$. Additionally, the attacker uses a probabilistic state vector $p^E = (p_1, \ldots, p_G)^\top$ to describe its knowledge about A’s state vector $s^A$. Each element $p_i$ is an approximation of the marginal probability $P(s^A_i = 0 | D)$ that the $i$-th bit of $s^A$ is 0 given all data $D$ observed by E before, i.e., inputs and outputs of A and B, which have already been transmitted over the public channel.

At the beginning of the probabilistic attack previous information about $s^A$ is not available. Therefore E starts with a neutral hypothesis and all $p_i$ are initialized with the prior probability $P(s_i = 0) = 1/2$.

In each inner round an input $X$ and a matrix $\pi$ are provided to all PPMs. Then A and B calculate their outputs and communicate them publicly. This enables E to update $p^E$ based on the observed data $X$, $\pi$, and $\tau^A$. For that purpose the posterior probability $P(s_i = 0 | p^E, X, \pi, \tau^A)$ is estimated using a Monte Carlo approach, which is similar to approximate Bayesian computation [37].

This works by generating $M$ state vectors $s^E$ which are compatible with the current observation as well as the prior knowledge obtained in previous rounds. The $G$ elements of a candidate state vector $s^E$ are sampled independently from the Bernoulli distribution with probabilities $P(s_i = 0) = p_i$ and $P(s_i = 1) = 1 - p_i$. Of course, it is only necessary to draw bits $s_i$ which are selected as weights by $\pi$. All others can be omitted without affecting the result. This shortcut speeds the sampling up considerably if $G \gg KN$. Then $s^E$ is plugged into E’s PPM together with $X$ and $\pi$ in order to calculate $\tau^E$. If E’s output matches A’s, $\tau^E = \tau^A$, the candidate is stored; if not, it is dismissed. This procedure goes on until $M$ valid state vectors $s^E$ have been produced.

Afterwards, the desired marginal posterior probability $P(s_i = 0 | p^E, X, \pi, \tau^A)$ can be estimated as the relative frequency of $s_i = 0$ in the sample. The result is then used to update all $p_i$ which have been selected as weights in the current round. The other elements of $p^E$ remain unchanged, because the attacker gained no information about the corresponding parts of $s^A$. Of course, this computation is repeated for the next inner round.

As the space of all weight matrices $W = (w_1, \ldots, w_K)$ is of size $2^{NK}$, approximately $2^{NK-1}$ of them are compatible with a given $X$, $\pi$, and $\tau^A$. Thus if the sampling algorithm generates $M \geq 2^{NK-1}$ state vectors, it would be similar to a brute force attack. But choosing such a large parameter $M$ is only feasible for a very small number of weights.

Updating $p^E$ as described above has the effect that its elements $p_i$ converge towards 0 or 1 after several rounds, so that finally $M$ equal state vectors with

$$p_i = 0 \quad \implies \quad s_i = 1, \quad (3.6)$$
$$p_i = 1 \quad \implies \quad s_i = 0 \quad (3.7)$$

are sampled. However, defining

$$s^E_* = \begin{cases} 
0 & \text{for } p_i > 1/2, \\
1 & \text{for } p_i \leq 1/2
\end{cases} \quad (3.8)$$

as the most probable state provided $p^E$, the attack is considered a success as soon as $s^E_* = s^A$ without regard to whether all the $p_i$ have collapsed to 0 or 1 or not.
In contrast, if one or more $p_i$ have collapsed to the wrong value, $E$ might be unable to achieve the desired output $\tau_E = \tau_A$ in a later round. Such a failure clearly indicates that the estimation of some $p_i$ has gone wrong. In order to avoid an infinite loop in this case, only a finite number of attempts is made to generate $M$ valid samples $s^E$. If the limit is reached, the element $p_i$ of $p^E$ which is closest to collapse is reset to the neutral hypothesis, $p_i = 1/2$.

Usually, the algorithm will not be able to guess $s^A$ correctly in less than one outer round, therefore we need a mechanism to transfer the information gained during an outer round into the next one. Let $p^{E-}$ be the probabilistic state vector after applying the previous algorithm on all the inner rounds of a whole outer round. In order to transfer the information the attacker calculates the probability distribution for the state $\sigma^E_1$ of the first hidden unit conditioned on the probabilistic state vector $p^{E-}$ as well as the input $X$ and the matrix $\pi$ for each of the inner rounds with $\tau_A = \tau_B$. The result is then used to construct the probabilistic state vector $p^{E+}$ for the start of the next outer round.

In the following we describe an algorithm to approximate the probability that a single hidden unit has internal state $\sigma_j = 0$ given $p^E$ and the corresponding public information of an inner round. The output $\sigma_j$ depends only on the number of 1s in the vector local field $h_j$, which is equal to the scalar local field $h_j$. Here we approximate the probability distribution $P(h_j = n|p^{E-}, X, \pi)$ of
3.2. SUCCESSFUL ATTACK ON PPM...

<table>
<thead>
<tr>
<th>$N$</th>
<th>time $t_s$</th>
<th>slope $a$</th>
<th>offset $b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>even</td>
<td>0.495 ± 0.028</td>
<td>2.01 ± 0.28</td>
<td></td>
</tr>
<tr>
<td>even</td>
<td>0.1275 ± 0.0038</td>
<td>2.180 ± 0.038</td>
<td></td>
</tr>
<tr>
<td>odd</td>
<td>0.245 ± 0.076</td>
<td>9.02 ± 0.95</td>
<td></td>
</tr>
<tr>
<td>odd</td>
<td>0.157 ± 0.028</td>
<td>4.33 ± 0.35</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Linear regression with model $t = aN + b$ for average synchronization time $t_s$ and break time $t_b$.

this quantity by a binomial distribution which uses the average probability of finding a 1 in $h_j$ as parameter

$$q_j = \frac{1}{N} \sum_{i=1}^{N} [x_{i,j} p_{\pi_i,j} + (1 - x_{i,j})(1 - p_{\pi_i,j})].$$

Then, the probability

$$P(\sigma_j = 0|\mathbf{p}^{E-}, X, \pi) = \sum_{n=0}^{N/2} P(h_j = n|\mathbf{p}^{E-}, X, \pi)$$

of $\sigma_j = 0$ is given by

$$P(\sigma_j = 0|\mathbf{p}^{E-}, X, \pi) = \sum_{n=0}^{N/2} \binom{N}{n} q_j^n(1 - q_j)^{N-n}. $$

Finally, the attacker stores this result in $\mathbf{p}^{E+}$ whenever $\tau^A = \tau^B$ occurred in the inner round. This procedure succeeded in conveying enough information from one outer round to the next one.

An alternative approach to this task seems to be Monte Carlo sampling of $\sigma_1^E$ conditioned on the final $\mathbf{p}^{E^+}$. But in our simulations this method proved to be prone to fail: either $\mathbf{p}^{E}$ was effectively reset or the algorithm could not generate enough valid weight candidates at some inner round. Thus we developed and used the analytic approach instead of calculating $\mathbf{p}^{E+}$ by sampling.

The attack described in this paper is often capable of guessing the state vector $\mathbf{s}^A$ in a number of outer rounds less than the number of rounds that $A$ and $B$ needed to synchronize. This result was reproduced for many different setups of the synchronizing PPMs: varying input vector size and varying state vector size. The usual setup for cryptographic is to use even $N$, since PPMs with odd $N$ synchronize notably slower or sometimes not at all [17]. However, the algorithm was also tried for odd $N$ with an illustrative purpose and yielded satisfactory results.

As for the technicalities a sampling size of $M = 10^3$ was chosen. This implies that for $N = 2, 4$ the algorithm works similar to a brute force attack, but for large $N$ only a small part of the weight space is sampled, e.g. for $N = 8$ only around a 3% of all possible weight configurations. The absent of performance drop notwithstanding the scarce sampling highlights the efficiency of the algorithm. The mechanism to prevent the attack from getting stuck was implemented by resetting one of the bits each time that $M^2 = 10^6$ consecutive
CHAPTER 3. PROBABILISTIC ATTACK

Figure 3.2: (color online) Success probability $P_s$ of the attack as a function of $N$ for PPMs with $K = 2$. Symbols denote the percentage of successful attacks found in 100 simulations (even $N$). For odd $N$ around 25 out of 100 runs had to be stopped after 30 outer rounds without a clear result, i.e. $t_a > 30$ and $t_b > 30$. These simulations were not considered for calculating the probability of success $P_s$. 
3.2. SUCCESSFUL ATTACK ON PPM...

unsuccessful attempts of generating a valid weight candidate were reached. Finally, the attack was considered a success as soon as $s^{E_s} = s^A$ has been reached. The number of outer rounds needed to achieve this is called break time $t_b$, which varies randomly depending on the initial conditions and the course of the key exchange.

Figure 3.1 shows that the mean values of synchronization time $\langle t_s \rangle$ as well as break time $\langle t_b \rangle$ grow linearly with increasing size $N$ of the input vectors. For all cases presented here we find that the attacker is faster than the two partners on average, $\langle t_b \rangle < \langle t_s \rangle$. Additionally, linear regression results shown in table 3.1 indicate that $\langle t_b \rangle$ grows slower than $\langle t_s \rangle$, so that increasing $N$ does not improve the security of the PPM-based key exchange.

Synchronization with odd $N$ is much slower than for even $N$. Only runs with $t_b < 30$ or $t_s < 30$ were considered here to reduce computational costs. This condition also excludes failed synchronization attempts caused by reaching a stable anti-parallel weight configuration [17], which can only happen if $N$ is odd.

In Fig. 3.2 the performance of the algorithm is presented in terms of the probability of success of the attack. Functionality for many more different setups is examined here. Regarding cases with even $N$, the performance of the algorithm generally increases as $N$ or $G$ become larger. For nearly all configurations shown here the success probability $P_s$ is above 80% and it actually reaches 100% in many situations. Odd $N$ is considerably more difficult for the attacker, but nevertheless the success probability $P_s$ remains larger than 50%. These values, however, have been obtained for single runs of our algorithm on each data set. As the method is non-deterministic due to Monte Carlo sampling in each inner round, repeating it on the same observations should lead to even more success.

Consequently, the results clearly show that the PPM-based neural key-exchange protocol using the parameter values $K$, $N$, and $G$ analyzed in this paper is not secure enough for any cryptographic application. Furthermore there is no indication that increasing the sizes of input or state vectors would reduce the success probability and lead to a secure configuration.

In contrast, the complexity of successful attacks on TPM-based neural cryptography increases exponentially with the number $2L + 1$ of possible weight values, but the effort of the partners grows only proportional to $L^2$ [22]. Here $L$ has the same effect as the key size in encryption algorithms, which allows to balance speed and security. While the probabilistic attack [27] has not been tested on TPM-based neural cryptography, it is quite likely that the same scaling law for $L$ applies to its success probability. But in order to answer this open question we are going to implement and analyze such probabilistic attacks also for TPMs.

The same question could be asked regarding the security of chaos cryptography [8, 38, 39], which is based on a similar synchronization principle [7]. Consequently, probabilistic attacks should be envisioned and tested there, too. Nevertheless, the specificity of the present implementation suggests that further development is needed for attacks on chaotic cryptography.
Acknowledgments

Seoane acknowledges the financial support of Fundación Pedro Barrié de la Maza and funding grant 01GQ1001B.
Chapter 4

The Tree Parity Machine and description of a probabilistic attack on TPM-based neural cryptography

4.1 Description of the Tree Parity Machine

Similarly to a Permutation Parity Machine, a Tree Parity Machine (TPM) [15, 16, 18–22] consists of an artificial neural network with one unit in the output layer and $K$ units in the hidden layer. Each hidden unit takes $N$ inputs. We can label the weights of the hidden units as $w_{i,j}$ and its inputs $x_{i,j}$; with $i \in [1,K]$, $j \in [1,N]$. Now the weights take an integer value $w_{i,j} = l \in [-L,L]$, where $L$ is the synaptic depth. For TPMs to converge during mutual learning it is necessary that $L < \infty$ (i.e. that the synaptic depth is bounded). Now the weights are not drawn from a pool of random numbers: they are initialized randomly and the synchronizing process involves a sort of random walk of each and every weight within the range $[-L,...,L]$.

Mutual learning of TPMs proceeds as it follows: Like in the case of PPM, a binary input vector is randomly generated at the beginning of each step. This time we encode it as $x_{i,j} = \pm 1$. We calculate the local field of unit $i$ as:

$$h_i = \sum_{j=1}^{N} x_{i,j} \cdot w_{i,j}. \quad (4.1)$$

The output of the TPM is:

$$\tau = \text{sign}(\prod_{i=1}^{K} h_i), \quad (4.2)$$

where $\text{sign}(x)$ denotes the sign function: $+1$ if $x > 0$, $-1$ if $x \leq 0$. Usually we will have two TPMs $A$ and $B$, and an eavesdropper $E$. We label the variables
consequently with a superindex, as we did in previous chapters. Whenever \( \tau^A = \tau^B \) we apply an update algorithm to the weights. Depending on the update rule we will talk about Hebbian, anti-Hebbian or random walk learning. We shall focus on Hebbian learning as the others are just variations of this one. (Also, depending on the update rule chosen we can find that the machines synchronize in a parallel or an anti-parallel way.) If the synchronizing TPMs produce the same output, the weights of the hidden units are updated whenever \( \tau^A = \text{sign}(h^A_i) \) – correspondingly whenever \( \tau^B = \text{sign}(h^B_i) \). The update rule for Hebbian learning reads:

\[
\begin{align*}
  w_{i,j}^+ & \rightarrow w_{i,j} = g(w_{i,j} + x_{i,j} \tau \cdot \Theta(h_i^A) \Theta(\tau^A \tau^B)).
\end{align*}
\]

Here: \( g(x) = x \) if \( x < -L \) and \( x > L \), \( g(x) = L \) if \( x > L \), and \( g(x) = -L \) if \( x < -L \). \( \Theta(x) \) stands for the step function and its role in the update rule is just to impose the conditions \( \tau^A = \tau^B \) and that only certain hidden units update. Repeated many times, this update rule causes the weights to wander within \([-L, L]\), and to bounce against the boundary of this interval whenever one weight hits \( \pm L \). Every time a weight reaches one of the extremes, since the other machine might be updating the same unit, the weights can become closer to each other or they might diverge. The key for synchronization is that converging steps are more probable than diverging ones. The key for TPMs to work as cryptographic engines is that, for any attack tried so far, converging bounces happen less often in \( E \) than in \( A \) and \( B \), whatever the method used by \( E \) to update her weights. As for PPMs, the main feature allowing this is mutual learning: \( A \) and \( B \) can force each other to learn, but \( E \) can only try to follow. Its update requests will be ignored.

Most of the literature in Neural Cryptography is devoted to the TPM-based key-exchange protocol. Since very early simple and geometric attacks were introduced. The first one is just a third machine trying to follow the updates of \( A \) and \( B \) whenever they update and without regard of its own output or internal states. The second one is similar: a single machine trying to follow up \( A \) and \( B \)'s updates, but it also takes into account the states of its hidden units. So, in the geometric attack, if \( \tau^A = \tau^B = \tau^E \) the update is similar to that of the simple attack; but if \( \tau^A = \tau^B \neq \tau^E \) the attacker knows that at least one of its hidden units computed differently from \( A \)'s ones, so \( E \) attempts to guess which one of its hidden units is wrong and changes the sign of this unit’s output before updating. In any case, this attack supposes little improvement since the simple one. Both perform poorly enough as to consider TPM-based key-exchange protocol a reasonably safe protocol.

The little advance in the field came with genetic and majority attacks, although the system is still considered secure after them. Indeed, the performance of these two attacks is also poor. A genetic attack considers a growing population of TPMs which have descendants with some mutations and attempts to calculate the fittest individuals. Since the overlap\(^1\) between \( E \)'s and \( A \)'s weights is not public, a genetic attack can only track the latest performance of the population in guessing \( A \)'s output as a measure of fitness. A majority attack uses an ensemble with a fixed number of attacking machines and it considers the internal state—i.e. the collection \([\text{sign}(h_1), \ldots, \text{sign}(h_K)]\)—which happens more often

\(^1\)A thorough definition of overlap for TPMs follows below.
among the population given an input to decide what hidden units to update in each learning step.

In [23] we can find a thorough comparison among the attacks described above. The main result of the security analysis is that TPM-based key-exchange protocol can be made arbitrarily safe by increasing $L$. Also, it is not found any notable security increase once $K >= 2$ while synchronization becomes slower. The most promising attack within the researched range of parameters is the genetic attack, although an extrapolation of the results suggests that the majority attack might perform better for larger $L$. Altogether, previously existing attacks could not break the TPM-based key-exchange protocol. In the following a new attack on TPMs is described: it is an implementation of a probabilistic attack (first suggested as a possible attack by [27], but never implemented nor envisioned before, to our knowledge) inspired by the successful attack on Permutation Parity Machines introduced in [34].
4.2 Probabilistic attack on Tree Parity Machines

As usual, let us consider two synchronizing TPMs $A$ and $B$ with $K$ hidden units each, $N$ inputs at each hidden unit and a synaptic depth $L$. Also, let us consider the attacking machine $E$ to be a probabilistic Tree Parity Machine (pTPM), which implements the same behavior as usual TPMs do but instead of weights it has additionally got probabilist weights $p_{i,j}(l) = P(w_{i,j} = l \in [-L, L])$. Each probabilistic weight is a probability distribution telling us how probable it is that $A$ presents $l$ as a weight in the $j$-th input of the $i$-th unit. A pTPM can be given the same input $x_{i,j}$ as usual TPMs and with it it can compute the probability that each hidden unit is activated $p(h_i) = P(h_i > 0)$ or not $P(h_i <= 0) = 1 - P(h_i > 0)$ and the probability $p(\tau^E) = P(\tau^E = 1)$ that the output is $+1$ given the input. As implemented until now, to compute these quantities a pTPM approximates the distribution stored in the probabilistic weights by a Gaussian distribution with:

$$
\langle w_{i,j} \rangle = \sum_{l=-L}^{L} l \cdot p_{i,j}(l),
$$

$$
\sigma^2(w_{i,j}) = \sum_{l=-L}^{L} (l - \langle w_{i,j} \rangle)^2 \cdot p_{i,j}(l) \quad (4.4)
$$

as mean and variance. This approximation might not be necessary, but it simplifies the calculus a lot and works so far to have an initial implementation of a pTPM running. Assuming that $p_{i,j}(l)$ can be approximated by a Gaussian distribution, the sum $\sum_{j=1}^{N} x_{i,j} \cdot \langle w_{i,j} \rangle$ is a random variable whose probability distribution can itself be approximated by a Gaussian with mean $(x_i w_i) = \sum_{j=1}^{N} x_{i,j} \cdot \langle w_{i,j} \rangle$ and variance $\sigma^2(x_i w_i) = \sum_{j=1}^{N} \sigma^2(w_{i,j})$. From here, we can approximate the computation of $p(h_i)$ to the integral of a Normal with mean 0 and variance 1 between the adequate limits (namely, between $\frac{-(x_i w_i)}{\sigma(x_i w_i)}$ and $+\infty$). This approximation shall be more accurate for large $L$ and $N$ owing to the central limit theorem. Finally, we can approximate $p(\tau^E)$ as well. It was done explicitly for $K = 1, 2$ and 3:

$$
p_{K=1}(\tau^E) = p(h_1),
$$

$$
p_{K=2}(\tau^E) = p(h_1)p(h_2) + (1 - p(h_1))(1 - p(h_2)),
$$

$$
p_{K=3}(\tau^E) = p(h_1)p(h_2)p(h_3) + (1 - p(h_1))(1 - p(h_2))p(h_3) +
\quad + (1 - p(h_1))p(h_2)(1 - p(h_3)) + p(h_1)(1 - p(h_2))(1 - p(h_3)) \quad (4.5)
$$

It was not implemented any pTPM with $K > 3$ for this work, but a proxy for $p(\tau^E)$ shall be used sooner of latter as expressions grow due to scaling combinatorial.

Usually it will not be needed to know $p(\tau^E)$ since what determines whether $A$ and $B$ –thus $E$– update are $\tau^A$ and $\tau^B$, but $p(h_i)$ is needed to determine what hidden units of $E$ are updated and how. There are two possible ways to update the hidden units: one stems from the learning rule employed by the synchronization algorithm and the other one is that of the probabilistic

---

2By default we track $A$, but same attack is possible towards $B$. 
4.2. PROBABILISTIC ATTACK ON TREE PARITY MACHINES

attack itself. The former is applied only whenever \( \tau^A = \tau^B \), trying to track 
A’s behavior; the later is applied after every computation, trying to gather as 
much information as possible from the input provided and the output \( \tau^A \) given 
the input. In the following, the two update rules—which are update rules for 
\( p_{i,j}(l) \)—are explained:

- **Monte-Carlo update of \( p_{i,j}(l) \):**
  Following the philosophy in [34] and previous chapters, the probabilistic 
attack is implemented by means of a Monte-Carlo sampling of the space of 
possible weights of the TPMs. Given \( K, N, \) and \( L; \) there are \((2L + 1)^NK\) 
possible weights in a TPM from which we have no further information. 
Assuming the same scenario but knowing the output of a TPM only restricts 
the size of the search space to \(~(2L + 1)^NK/2~\). 
For each computation implemented with a TPM \( n_s \) valid samples were 
generated with a Monte-Carlo procedure. As a valid sample we understand 
a possible set of weights that, with the same input as the original TPM, 
would yield the same output. Computing the frequency with which each 
possible weight shows up in each position over all the samplings, the a-
posteriori probability is approximated \( p_{i,j}(l|x, \tau^A) \). 

Obviously, to generate the Monte-Carlo samplings we depart from the 
information gathered so far —i.e. from an a-priori probability distribution 
\( p_{i,j}(l) \). At the very beginning of the synchronization procedure we de-
part from a situation of no information, so assuming \( p_{i,j}(l) = 1/(2L + 1) \) 
for every possible weight seems a reasonable thing to do. However, the 
system usually presents some symmetry regarding parallel or anti-parallel 
synchronization (both can be implemented with different learning pro-
duces —e.g. Hebbian vs. anti-Hebbian—, but the Monte-Carlo sampling 
initially does not make any distinction between the two scenarios). The 
former initialization of the a-priori probabilities is the most symmetric one 
and implementing it seems to have some interesting effects in the time it 
takes the Monte-Carlo sampling to break the symmetry between parallel 
or anti-parallel synchronization. It was chosen to initialize \( p_{i,j}(l) \) with 
a random probability distribution (always obeying \( \sum_l p_{i,j}(l) = 1 \)). This 
way, the system is already posed to break the symmetry. The a-posteriori 
probability \( p_{i,j}(l|x, \tau^A) \) after a synchronization step becomes the a-priori 
probability of the following step and random numbers are drawn from 
there to generate the very next sampling. 

Similarly as in [34], it was necessary to implement a reset mechanism so 
that the algorithm does not get stuck trying to generate a valid sample 
of the weight space from a failed probability distribution. So, after \( n_R \) 
samples it was chosen the \( p_{i,j}(l) \) with a lower variance and it was reset 
to a random probability distribution. The reason to reset the probability 
distribution with lower variance is that we are supposed to get not valid 
samples because certain weights could have collapsed towards a wrong 
value. Probability distributions with a lower variance indicate that these 
weights are closer to collapse. The argument is similar to the one used 
in [34] and in previous chapters, but this does not seem to be the best for 
the current case, as it will be illustrated later.
Hebbian update of $p_{i,j}(l)$:

We need to implement an update rule for $p_{i,j}^-(l) \rightarrow p_{i,j}^+(l)$ whenever a learning step between $A$ and $B$ takes place. (In the following: $p^-$ and $p^+$ represent the probability distribution before and after a learning step, respectively.) This is partly trivial: a learning step is a shift of the weights in just one unit, so we can try out:

$$p_{i,j}^+(l) = p_{i,j}^-(l \mp 1), \quad (4.6)$$

if $l \neq \pm L$ and:

$$p_{i,j}^+(\pm L) = p_{i,j}^-(\pm L) + p_{i,j}^-(l \mp 1) \quad (4.7)$$

or:

$$p_{i,j}^+(\mp L) = 0 \quad (4.8)$$

whenever needed if $l = \pm L$. The signs used inside the previous equations and which one of the two last equations is used for $l = \pm L$ depend on the direction of the step of the random walk, which—in the case of the Hebbian learning rule—is determined by the product $x_{i,j}r^A$ for each weight. Also, $\text{sign}(h^A_i)$ determines whether a unit requires being updated. The direction of the update step can be solved with the public information available alone, but $E$ cannot know what units need an update. The best she can do is to compute $p(h_i)$. This probability that each hidden unit is activated (i.e. that $\text{sign}(h_i) = +1$) is used to compute a flow between the different entries of the probability distributions $p_{i,j}(l)$ as it follows:

$$p_{i,j}^+(l) = (1 - p(h_i))p_{i,j}^-(l) + p(h_i)p_{i,j}^-(l \mp 1) \quad (4.9)$$

if $l \neq \pm L$. For $l = \pm L$ the update rule reads:

$$p_{i,j}^+(\pm L) = p_{i,j}^-(\pm L) + p(h_i)p_{i,j}^-(l \mp 1) \quad (4.10)$$

or:

$$p_{i,j}^+(\mp L) = (1 - p(h_i))p_{i,j}^-(L). \quad (4.11)$$

again depending on the direction of the updating step.

As a measure of the performance of the attack described above we do not need to implement any measure different from the order parameters described in the literature [23]. We use the matrix $f^A_{i,a,b} = P(a \in w^A_i \cap b \in w^B_i)$. ($i$ labels the matrix since there is one such a matrix associated to each hidden unit. $a$ and $b$ label the entries of the matrix.) This can be easily extended to compute the overlap between a TPM and a pTPM. Since the probabilistic weights provide plain probabilities of finding $w_{i,j} = l$ for each $l$, it is straightforward to calculate $f^A_{i,a,e} = P(a \in w^A_i \cap e \in w^B_e)$: the probability that $A$ has got a weight with value
4.2. PROBABILISTIC ATTACK ON TREE PARITY MACHINES

\( a \) and that \( E \) has got a weight with value \( e \) at the same time. From this:

\[
Q_i^A = \sum_{a=-L}^{L} \sum_{c=-L}^{L} a^2 f_{a,c}^i,
\]
\[
Q_i^E = \sum_{a=-L}^{L} \sum_{c=-L}^{L} e^2 f_{a,c}^i,
\]
\[
R_i^{AE} = \sum_{a=-L}^{L} \sum_{c=-L}^{L} a e f_{a,c}^i.
\]

(4.12)

Consequently:

\[
\rho_i^{AE} = \frac{R_i^{AE}}{\sqrt{Q_i^A Q_i^E}} \Rightarrow
\]
\[
\Rightarrow \rho^{AE} = \langle \rho_i^{AE} \rangle = \frac{\sum_{i=1}^{N} \rho_i^{AE}}{N}.
\]

(4.13)
4.3 Preliminary results

So far they can not be offered any definitive—not even remotely complete—results about the performance of the attack described above. Exhaustive simulations will be carried out soon to explore its efficiency with different setups and to deduce how the complexity of the attack scales for increasing synchronization complexity, as it is the usual procedure [23, 26, 30, 32] and as it was done for PPMs [34]. By now, only single synchronization events have been simulated and attacked at the same time, and these are the scarce results outlined in the following. There are not statistics enough to draw any conclusions apart from being optimistic or pessimistic about the attack. Thus, so far, the robustness of the TPM-based key-exchange protocol remains the same as before this work. This work does not make any claim on the safety of the protocol and every following word must be taken as speculative.

In figures 4.1 and 4.2 it is illustrated the performance of the attack. The settings for the machines are: \( N = 10 \), and \( L = 50 \) with \( K = 1 \) in figure 4.1 and \( K = 2 \) in figure 4.2 yielding \( 10^1 \) and \( 10^2 \) possible weights for the synchronizing TPMs. Let us note the large synaptic depth of the weights: \( L = 50 \) —this is the parameter usually modified to make TPM-based neural cryptography safer. This is a large \( L \) when compared to the settings used in the literature to test previous attacks. Meanwhile, \( N \) is here much smaller than in previous studies. In both cases, neither the TPMs (red) where able to fully synchronize (although in figure 4.1 they could approach a full overlap), nor could the pTPM (green and blue) completely guess \( A \)'s weights. Simulations were stopped before any convergence.

In both cases we can observe how the attack is able to approach full convergence with \( A \)'s weights: in the first case, it does so some steps before the TPMs approach full convergence; and in the second case much before the two TPM even begin to converge. Once again, this has not got any implication for the robustness of the TPM-based key-exchange protocol. We have not got a through study of multiple cases to check whether the success of the attack is of any statistical significance or if it just happened by chance.

But some insights can be extracted from these test cases. Let us note the following:

- In figure 4.1 we can observe how the overlap of the most probable weight suddenly drops many times while the overlap for \( p_{i,j}(l) \) remains large before dropping as well. This is due to the reset mechanism implemented in the pTPM, to which the former overlap is more sensitive. These drops reveal that the reset mechanism is erasing some incorrect weights, so maybe choosing the \( p_{i,j}(l) \) with lower variance is not a good idea for this attack. Also, the current reset mechanism just annihilates any information gathered for one probabilistic weight. It should be possible to improve the reset procedure to retain at least some of this information.

- In figure 4.3 it is displayed the very beginning of a case were both the synchronization and the attack take a long time. Synchronization has not began yet and the overlap between TPMs remains around 0.1. Let us
Figure 4.1: **Synchronization vs. dynamic attack.** $K = 1, N = 10, L = 50$ as general setup. For the probabilistic attack: $n_s = 10000 \ll 10^{10}$, $n_R = 100\ 000$. The synchronization of two TPMs (red) takes fairly more time than it takes the overlap between the probabilistic weights and the target TPM’s weights to converge to close to 1 (green). The overlap between the target’s weights and the most probable weights is also tracked (blue).
Figure 4.2: **Synchronization vs. dynamic attack.** $K = 2$, $N = 10$, $L = 50$ as general setup. For the probabilistic attack: $n_s = 10000 << 10^20$, $n_R = 100\ 000$. Synchronization between TPMs (red) could not even begin in the current case, while the overlap between $A$’s weights and $E$’s probabilistic weights (green) has raised up to almost 1 in relatively few rounds. Again, the overlap between $A$’s weights and $E$’s most probable weights is displayed (blue).
4.3. PRELIMINARY RESULTS

Figure 4.3: Synchronization vs. dynamic attack, initial steps. $K = 2$, $N = 100$, $L = 50$ as general setup. For the probabilistic attack: $n_s = 10000$, $n_R = 100\,000$. This case takes a long time to synchronize (red) and also for the attack to make any improvement. It is illustrated how the overlap with the probabilistic weights (green) is close to 0, probably due to some symmetry between parallel and anti-parallel synchronization that the probabilistic weights preserve. Also, it is shown how the overlap with the most probable weight (red) of the pTPM readily reaches a larger overlap due to a forced temporary commitment towards parallel or anti-parallel alignment of the weights.

remember that because of Hebbian learning, TPMs are forced to parallel-synchronize their weights. On the other hand, we observe how the overlap with the probabilistic weights is close to 0 all the time while the overlap with the most probable weight ranges between $-0.05$ and 0.2. The probabilistic part of the attack has not got any preference for either parallel or anti-parallel synchronization and it undergoes a process of spontaneous symmetry breaking which arises from the data provided and the output of $A$ alone. This process takes a time during which the attack retains information from both possible cases –parallel and anti-parallel synchronization. This can prevent the algorithm to converge earlier. Opposed to this, the most probable weight is forced to commit towards one of the two options at each iteration because only one most probable weight can be chosen each time. We could use this information to force a symmetry breaking earlier and maybe improve the performance of the algorithm.
Summing up the scarce results we have discussed: We have found some cases in which the probabilistic attack is promising, but these examples have not got any significance towards the robustness of the TPM-based key-exchange protocol. There is not any statistical evidence that these results did not happen by chance alone. We can still extract some useful information to further improve the probabilistic attack described here: one regarding the reset mechanism and another one regarding the necessary and resource consuming symmetry breaking process. It is expected that thorough experiments will be carried out soon, so the results will be complemented and safety of the cryptographic procedure could be assessed.
Bibliography

[1] Cybenko, G.
Approximation by Superpositions of a Sigmoidal Function.

[2] Dourlens, S.
Neuro-cryptographie appliquée.

*Neural Networks: A Comprehensive Foundation*
(Prentice Hall, 1998).

[4] Bishop, Christopher M;
*Pattern Recognition and Machine Learning*
(Springer Verlag, 2006).

[5] Buck, J.; and Buck, E.;
Mechanism of Rhythmic Synchronous Flashing of Fireflies.

A synchronized quorum of genetic clocks.

Synchronization of chaotic systems.

[8] Cuomo, K. M. and Oppenheim, A. V.
Circuit implementation of synchronized chaos with applications to communications.

Sequential synchronization of chaotic systems with an application to communication.

Chaos-based communications at high bit rates using commercial fibre-optic links.
Synchronization of coupled biological oscillators under spatially heterogeneous environmental forcing.

[12] Solé, R. V.; Menéndez de la Prida, L.
Controlling Chaos in Discrete Neural Networks.

Dynamics of Interacting Neural Networks.

Interacting Neural Networks.

Disorder generated by interacting neural networks: application to econophysics and cryptography.

[16] Kinzel, W.

[17] Reyes, O. M.; Kopitzke, I.; and Zimmermann, K.-H.
Permutation parity machines for neural synchronization.

Interacting neural networks and cryptography.

Cryptography based on neural networks - analytical results.

Mutual learning in a tree parity machine and its application to cryptography.

The Theory of Neural Networks and Cryptography.

[22] Ruttior, A.; Kinzel, W.; and Kanter, I.
Dynamics of neural cryptography.

[23] Ruttior, A.
*Neural Synchronization and Cryptography*
Permutation parity machines for neural cryptography.  


Secure key-exchange protocol with an absence of injective functions.  

[27] Klimov, A.; Mityaguine, A.; and Shamir, A.  
Analysis of neural cryptography.  

Public channel cryptography by synchronization of neural networks and chaotic maps.  

[29] Ruttor, A.; Kinzel, W.; Shacham, L.; and Kanter, I.  
Neural Cryptography with feedback.  

Cooperating attackers in neural cryptography.  

[31] Klein, E.; Mislovaty, R.; Kanter, I.; Ruttor, A.; and Kinzel, W.  
Synchronization of neural networks by mutual learning and its applications to cryptography.  

Neural cryptography with queries.  

Genetic attack on neural cryptography.  

[34] Seoane, L. F. & Ruttor, A.  
Successful attack on PPM-based neural cryptography  

[35] Volkmer, M. and Wallner, S.  
Tree Parity Machine Rekeying Architectures.  
[36] Mühlbach, S. M. and Wallner, W.
Secure communication in microcomputer bus systems for embedded devices.

Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems.

[38] Grassi, G. and Mascolo, S.
A system theory approach for designing cryptosystems based on hyper-chaos.

Public-channel cryptography based on mutual chaos pass filters.