AGH UNIVERSITY OF KRAKOW

Faculty of Computer Science, Electronics and Telecommunications
Institute of Telecommunications

FIELD OF SCIENCE
Engineering and Technology

SCIENTIFIC DISCIPLINE
Information and Communication Technology

DOCTORAL DISSERTATION

Enhancing Tree Parity Machines with Non-Binary Input Vectors Without Compromising Cryptographic Key Security

Usprawnienie działania sztucznych sieci neuronowych Tree Parity Machines poprzez wykorzystanie niebinarnych wektorów wejściowych z zachowaniem bezpieczeństwa otrzymanego klucza kryptograficznego

Author:
mgr inż. Miłosz Stypiński

Supervisor:
dr hab. inż. Marcin Niemiec, prof. AGH

Krakow, 2025

Podziękowania

Pragnę z całego serca podziękować mojemu promotorowi, Profesorowi Marcinowi Niemcowi, za poświęcony czas, anielską cierpliwość oraz nieustanne dzielenie się motywacją do działania. Życzę każdemu dyplomantowi, aby miał zaszczyt współpracować z tak wspaniałym Promotorem.

Szczególne podziękowania kieruję do mojej Narzeczonej Kasi, a – mam nadzieję – przyszłej Żony, za niekończące się wsparcie. Bez jej zrozumienia, motywacji i pomocy w codziennych obowiązkach ukończenie tej rozprawy nie byłoby możliwe.

Na końcu chciałbym wyrazić wdzięczność wobec wszystkich osób, które w trudnych chwilach wspierały mnie dobrym słowem i choćby w najmniejszym stopniu przyczyniły się do powstania tej pracy.

Spis treści

1 Autoreferat

Niniejszy autoreferat stanowi podsumowanie spójnego cyklu publikacji, które składają się na rozprawę doktorską. Przedstawiono motywację, cel oraz listę publikacji. Przede wszystkim jednak scharakteryzowano mechanizm synchronizacji sztucznych sieci neuronowych w celu ustalenia współdzielonego klucza kryptograficznego przy użyciu niebinarnych wektorów wejściowych oraz rozwiązanie mające na celu zwiększenie bezpieczeństwa ustalanego w ten sposób ciągu binarnego.

1.1 Wstęp

Rozprawa doktorska skupia się na tematyce usprawnienia protokołu ustalania klucza kryptograficznego wykorzystującego sztuczne sieci neuronowe. Rozważanym rozwiązaniem jest protokół wzajemnej synchronizacji sztucznych sieci neuronowych o strukturze nazwanej Tree Parity Machine (TPM). Celem rozprawy jest poprawa efektywności rozważanego rozwiązania poprzez autorskie usprawnienie metody uczenia sztucznych sieci neuronowych TPM, poprzez zastosowanie niebinarnych wektorów uczących w procesie wzajemnej synchronizacji. Przedstawione rozwiązanie zostało poddane dogłębnej analizie wydajności i bezpieczeństwa. Potwierdzono, że zastosowanie niebinarnych wektorów wejściowych poprawia bezpieczeństwo protokołu wykorzystującego sieci TPM poprzez przyspieszenie ich synchronizacji. Jednak w efekcie zastosowania tej metody zdiagnozowany został problem, który przejawia się w nierównomiernym rozkładzie bitów otrzymanego klucza kryptograficznego, co prowadzi do obniżenia entropii klucza. Finalnie jednak problem ten został rozwiązany poprzez zdefiniowanie algorytmu wyrównującego wagi, którego działanie sprowadza rozkład statystyczny wartości występujących w kluczu kryptograficznym do rozkładu jednostajnego.

Przedstawiam rozprawę doktorską o tytule: „Usprawnienie działania sztucznych sieci neuronowych Tree Parity Machines poprzez wykorzystanie niebinarnych wektorów wejściowych z zachowaniem bezpieczeństwa otrzymanego klucza kryptograficznego“ (*ang. „Enhancing Tree Parity Machines with Non-Binary Input Vectors Without Compromising Cryptographic Key Security“*). Tezę rozprawy można sformułować w następującej postaci: **wykorzystanie niebinarnych wektorów wejściowych oraz algorytmu wyrównującego wagi usprawnia proces synchronizacji w sieciach neuronowych o strukturze Tree Parity Machine (TPM), jednocześnie zachowując bezpieczeństwo otrzymanego klucza kryptograficznego**.

Rozprawa jest cyklem spójnych tematycznie publikacji i składają się na nią następujące pozycje.

1. Niemiec, M., Dziech, A., Stypiński, M., Derkacz, J., „Quantum-Based Solutions for the Next-Generation Internet“, Information & Security: An International Journal, vol. 43, no. 1, pp. 62-72, 2019.
   doi: 10.11610/isij.4306

2. Stypiński M., Niemiec M., „Synchronization of Tree Parity Machines Using Nonbinary Input Vectors“, IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 1, pp. 1423-1429, 2024.
doi: 10.1109/TNNLS.2022.3180197
Impact Factor (2023) = 10.2

3. Stypiński, M., „Analiza zależności pomiędzy parametrami sieci TPM a czasem synchronizacji“, Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne, SIGMA-NOT, sp. z.o.o., t. 4, str. 169-172, 2022.
doi: 10.15199/59.2022.4.35

4. Stypiński, M., Niemiec, M., „Impact of Nonbinary Input Vectors on Security of Tree Parity Machine“, Communications in Computer and Information Science, Springer International Publishing, pp. 94-103, 2022.
doi: 10.1007/978-3-031-20215-5_8

5. Stypiński, M., Niemiec, M., „Security of Neural Network-Based Key Agreement Protocol for Smart Grids“, Energies, MDPI, vol. 16, no. 10: 3997, 2023.
doi: 10.3390/en16103997
Impact Factor (2023) = 3.0

6. Stypiński M., Niemiec M., „Weight equalization algorithm for tree parity machines“, Computer Networks, vol. 257, 2025.
doi: 10.1016/j.comnet.2024.110988
Impact Factor (2023) = 4.4

7. Stypiński, M., „Wpływ parametrów sieci typu Tree Parity Machine na bezpieczeństwo algorytmu wyrównującego wagi“, Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne, SIGMA-NOT, sp. z.o.o., t. 4, str. 83-86, 2024.
doi: 10.15199/59.2024.4.13

1.2 Motywacja

Współcześnie wykorzystywane protokoły, takie jak SSL (*ang. Secure Sockets Layer*) oraz jego następca TLS (*ang. Transport Layer Security*), tworzą zabezpieczony tunel transmisyjny zbudowany z wykorzystaniem potencjalnie niebezpiecznej infrastruktury [17]. Tworzenie takiego tunelu składa się z następujących faz: wybór parametrów tunelu, uwierzytelnienie oraz uzgodnienie klucza kryptograficznego. W efekcie strony komunikacji mają pewność z kim się komunikują oraz posiadają klucze kryptograficzne, które są wykorzystywane do szyfrowania transmisji.

Proces uzgodnienia kluczy kryptograficznych jest zwykle integralną częścią zabezpieczenia komunikacji. Jednym z popularnych protokołów uzgadniania klucza kryptograficznego jest protokół Diffiego-Hellmana, który w praktyce zrewolucjonizował komunikację z wykorzystaniem niezaufanego medium [5]. Wykorzystanie protokołów z tej rodziny umożliwia utajnienie z wyprzedzeniem (*ang.*

forward secrecy). W efekcie złamanie pojedynczego klucza sesji nie dostarcza żadnych informacji na temat kluczy użytych w przeszłości. W przypadku, gdy nie jest wymagane utajnienie z wyprzedzeniem, do dystrybucji poufnych kluczy można wykorzystać szyfry asymetryczne. Nadawca znając klucz publiczny odbiorcy może zaszyfrować poufny klucz kryptograficzny, a wtedy jedynie odbiorca dysponując komplementarnym kluczem prywatnym jest w stanie deszyfrować taką wiadomość.

Jednak kryptografia asymetryczna wykorzystuje warunkowe bezpieczeństwo, które polega na braku możliwości odwrócenia wykorzystanej operacji algebraicznej leżącej u podstaw algorytmu. W przypadku protokołu Diffiego-Hellmana jest to brak możliwości obliczenia logarytmu dyskretnego, w przypadku protokołu ECDH (*ang. Elliptic Curve Diffie-Hellman*) jest to trudność odwrócenia operacji na punktach krzywej eliptycznej, a w przypadku RSA bezpieczeństwo jest zagwarantowane przez niemożność faktoryzacji dużych liczb [3].

W roku 1994 Peter W. Shor zaproponował algorytm będący w stanie efektywnie rozkładać na czynniki pierwsze duże liczby złożone oraz skuteczne liczyć logarytmy dyskretne [25]. Jednak zdefiniowany algorytm posiada złożoność obliczeniową należącą do klasy kwantowego czasu wielomianowego z ograniczonym błędem (*ang. bounded-error quantum polynomial time, BQP*) [16]. Zastosowanie tego algorytmu wymaga komputera kwantowego o odpowiedniej liczbie kubitów. Obecnie technologia obliczeń kwantowych rozwija się bardzo dynamicznie. Choć na ten moment nie stanowi ona bezpośredniego zagrożenia dla kryptografii asymetrycznej to jednak istnieje możliwość, że w przyszłości powstanie komputer kwantowy, który będzie mógł zagrozić obecnym fundamentom bezpiecznej komunikacji [1].

Rozwój komputerów kwantowych otworzył nowy kierunek badań – kryptografię postkwantową (*ang. post-quantum cryptography*), czyli badania nad algorytmami odpornymi na ataki z udziałem komputerów kwantowych, np. na których działał będzie algorytm Shora. Pierwszym standaryzacyjnym efektem badań jest konkurs ogłoszony przez National Institute of Standards and Technology (NIST), którego przedmiotem jest przedstawienie bezpiecznej alternatywy dla aktualnej kryptografii asymetrycznej. W konkursie wyłoniono finalistów – algorytmy których bezpieczeństwo jest zapewnione przez brak możliwości odwrócenia funkcji skrótu oraz przez algebraiczne operacje oparte na kratach [14]. Protokół synchronizacji sztucznych sieci neuronowych TPM, wykorzystujący uczenie wzajemne i będący tematem rozprawy, również cechuje się odpornością na ataki z wykorzystaniem komputera kwantowego [19].

Publikacja nr 1 z cyklu stanowi wprowadzenie w rozważaną tematykę – omawia możliwe zagrożenia związane z rozwojem obliczeń kwantowych, wskazuje kierunki ewolucji kryptografii asymetrycznej oraz przedstawia zasady kryptografii kwantowej [16].

1.3 Studium literatury

Uczenie wzajemne z wykorzystaniem sztucznych sieci neuronowych jest rozwiązaniem alternatywnym do ustalania kluczy symetrycznych przy użyciu krypto-

grafii asymetrycznej i jednocześnie odpornym na ataki z udziałem komputera kwantowego. Sieci neuronowe o strukturze TPM zostały zaproponowane w pracach Metzler et. al. [13] oraz Kinzel et. al [11]. Wspomniani autorzy udowodnili, że dwie specyficzne sieci neuronowe są w stanie zsynchronizować się do takiego samego stanu po odpowiedniej liczbie iteracji. Zastosowanie sieci TPM w kontekście uzgodnienia klucza kryptograficznego zostało zaproponowane w [10], gdzie klucz kryptograficzny jest tworzony z wektora uzyskanych wag w efekcie synchronizacji sztucznych sieci neuronowych. Podsumowanie działania sieci TPM wraz z opisem ryzyka zostało przedstawione w pracy [19].

W kolejnych latach sieci TPM doczekały się wielu usprawnień oraz potencjalnych scenariuszy zastosowań. Pierwszym usprawnieniem było zastosowanie wstępnie udostępnionej tajnej wartości, która dynamicznie definiowała granicę wartości wag sieci neuronowej [2]. Kolejno zaproponowano wykorzystanie dodatkowej warstwy ukrytej oraz uczenie wzajemne inspirowane naturą [23]. Inny rodzaj usprawnienia zaproponowano w [6], gdzie wykorzystano rozkład Gaussa do wstępnej inicjalizacji wartości wag warstwy ukrytej. Kolejne usprawnienia dotyczyły wektora wejściowego i zbioru będącego podstawą losowania jego wartości. W [7] zaproponowano wykorzystanie wektora uczącego składającego się liczb zespolonych o binarnych współczynnikach. Później zaproponowano uogólnienie tej metody, gdzie wykorzystano wektory binarne zamiast wartości binarnych, co skutkowało wykorzystaniem matrycy binarnej jako wejścia do omawianej sieci neuronowej [9].

Synchronizacja wzajemna została poddana wielu testom bezpieczeństwa. Wszystkie zdefiniowane ataki bazują na ataku man-in-the-middle, gdzie atakujący podsłuchuje wymianę danych pomiędzy podmiotami uzgadniającymi klucz kryptograficzny. Następnie na tej podstawie próbuje uzyskać dane na temat uzgodnionego klucza kryptograficznego. Pierwszy atak polega na wykonaniu iteracji procesu synchronizacji pomimo niespełnionych warunków koniecznych [18]. W efekcie, złośliwa sieć TPM szybciej dostosowuje się do sieci uzgadniających klucz kryptograficzny. Kolejny zaproponowany atak polega na wykorzystaniu więcej niż jednej sieci neuronowej, bądź jest wykonywany przez więcej niż jednego atakującego [24]. Zbiór takich sieci jest wykorzystywany do głosowania dotyczącego sposobu uczenia wszystkich z nich. Ostatni zdefiniowany atak wykorzystuje algorytm genetyczny, którego celem jest utworzenie populacji synchronizowanych sieci TPM, a następnie na podstawie funkcji dopasowania (*ang. fitness function*), wybór kolejnych pokoleń [20]. Dostępne ataki zostały podsumowane w [12].

Pomimo eksperymentalnej natury sieci neuronowych typu TPM zaproponowane zostały scenariusze wykorzystujące synchronizację wzajemną w celu uzgodnienia klucza kryptograficznego. Pierwsza propozycja wykorzystania sieci TPM została zaproponowana w [32], gdzie autorzy zaimplementowali algorytm uczenia wzajemnego w systemach wbudowanych. Kolejna implementacja sieci TPM z wykorzystaniem mikrokontrolerów została zaproponowana w [8]. Potencjalne zastosowanie zostało przedstawiono w [22], gdzie autorzy zaproponowali zastosowanie struktur TPM w sieciach bezprzewodowych. Aplikacja sieci TPM w kwantowej dystrybucji klucza kryptograficznego została opisana w [15], gdzie

zaproponowano wykorzystanie uczenia wzajemnego jako mechanizmu eliminacji błędów występujących w kluczu kryptograficznym.

1.4 Rozważane rozwiązanie i zaproponowane usprawnienia

Tematem pracy doktorskiej jest usprawnienie działania algorytmu uzgodnienia klucza kryptograficznego z wykorzystaniem sztucznych sieci neuronowych typu TPM poprzez zastosowanie niebinarnych wektorów wejściowych. Rozwiązanie to zostało zaproponowane w **publikacji nr 2 cyklu** [30]. Rozszerzona analiza wpływu proponowanego usprawnienia na czas synchronizacji została przedstawione w **publikacji nr 3 cyklu** [27].

Ponadto, w ramach cyklu przeprowadzona została analiza bezpieczeństwa rozważanego usprawnienia oraz zaproponowano potencjalne zastosowanie omawianych sieci neuronowych. Wyniki tych badań zostały przedstawione w **publikacjach nr 4 i 5 cyklu** [26, 29].

W efekcie zastosowania niebinarnych wektorów wejściowych bezpieczeństwo wymiany klucza kryptograficznego ulega poprawie. Niestety, wykorzystanie niebinarnych wektorów wejściowych skutkuje negatywnym wpływem na otrzymany wektor wag, będący podstawą do otrzymania klucza kryptograficznego. Problem ten został zaadresowany w **publikacji nr 6 cyklu** [31]. W artykule tym zaproponowany został algorytm, który przeciwdziała wspomnianym wadom poprzez wyrównywanie prawdopodobieństwa występowania poszczególnych wartości w wektorze wag, zanim zostanie on zamieniony na klucz kryptograficzny. **Publikacja nr 7 cyklu** cyklu rozwija analizę działania algorytmu wyrównującego wagi oraz bada jego zależność od parametrów sieci neuronowej o strukturze TPM [27].

1.4.1 Opis działania modelu

Struktura sieci TPM to zwykle dwuwarstwowy perceptron o binarnych wejściach i binarnej odpowiedzi, który może zostać wykorzystany do ustalenia klucza kryptograficznego. W procesie uzgadniania klucza, nazywanego również synchronizacją lub uczeniem wzajemnym, dwie sieci neuronowe iteracyjnie wymieniają się danymi uczącymi, bądź wektorami wejściowymi i towarzyszącą im odpowiedzią sieci. Wykorzystując te dane sieci TPM stosują odpowiednie reguły uczące. W efekcie obie strony komunikacji otrzymują identyczne sieci TPM. Wagi pierwszej warstwy otrzymane z tak zsynchronizowanych sieci tworzą klucz kryptograficzny, który może zostać wykorzystany do zabezpieczania komunikacji.

Pierwsza warstwa modelu sieci TPM składa się z K neuronów, z których każdy posiada N wejść. Dla każdego wejścia istnieje powiązana z nim waga w_{kn}. Wagi mogą przyjmować wartości ze zbioru liczb całkowitych ograniczonych przez parametr L, a więc należą do zbioru $\{w_{kn} : w_{kn} \in \mathbb{Z} \wedge -L \leq w_{kn} \leq L\}$. Funkcja aktywacji każdego neuronu pierwszej warstwy to zmodyfikowana funkcja signum. Binarna natura sztucznej sieci neuronowej TPM wymaga aby funkcja

aktywacji neuronów każdej warstwy zwracały dwie wartości, na potrzeby przyszłych obliczeń będą to wartości -1 oraz 1. Wzór (1) prezentuje zmodyfikowaną funkcję signum.

$$\sigma(x) = \begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases} \tag{1}$$

Sieć TPM przyjmuje na wejściu binarny wektor X, którego długość jest równa $K \cdot N$, ponieważ dla każdego z K neuronów istnieje N wejść. Podobnie jak w funkcji aktywacji neuronów pierwszej warstwy, wektor X składa się z wartości -1 i 1, czyli ostatecznie przyjmuje wartość ze zbioru $X \in \{-1; 1\}^{K \cdot N}$.

Dla tak przygotowanego wektora wejściowego odpowiedź sieci neuronowej TPM jest liczona w następujący sposób:

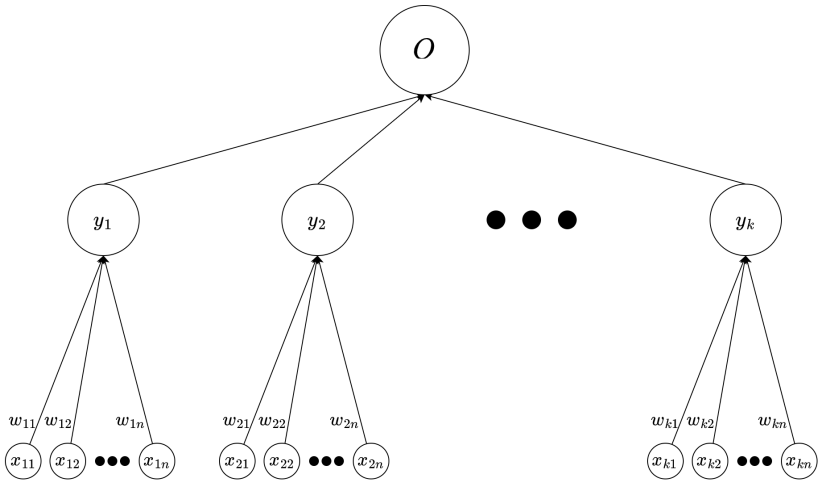
1. początkowo liczone są odpowiedzi neuronów pierwszej warstwy y_k, które są równe wartości funkcji (1) dla sumy iloczynów wag i ich odpowiadającym wejściom (2);
$$y_k = \sigma(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}) \tag{2}$$
2. następnie liczony jest iloczyn wartości y_k co stanowi odpowiedź O całej sieci neuronowej.
$$O = \prod_{k=1}^{K} y_k \tag{3}$$

Odpowiedź struktury TPM obliczona na podstawie wzorów (2) i (3) ma charakter binarny i zawiera się w zbiorze $\{-1; 1\}$, ponieważ iloczyn binarnych wartości y_k również jest binarny. Struktura sieci TPM została przedstawiona na rysunku 1.

1.4.2 Algorytm uzgodnienia klucza kryptograficznego

Proces uczenia jest specyficzny dla uczenia sieci neuronowych typu TPM. Z uwagi na nieciągły charakter wartości jaki mogą przyjmować wagi oraz binarne odpowiedzi neuronów tworzących sieć, algorytm propagacji wstecznej nie jest możliwy do zastosowania.

Synchronizacja dwóch struktur TPM polega na wzajemnej wymianie wylosowanych wektorów wejściowych oraz odpowiedzi struktur dla danego wektora. Następnie wykorzystywane są odpowiednie reguły uczące, które gwarantują, że uczenie zakończy się w skończonym czasie [32]. W efekcie udanej synchronizacji dwie sieci neuronowe TPM posiadają dokładnie takie same wartości wag, które tworzą wektor W. Następnie, tak uzyskany wektor W może zostać wykorzystany do celów kryptograficznych. Algorytm będący podstawą wymiany klucza kryptograficznego pomiędzy dwoma podmiotami (A i B), nazwany uczeniem wzajemnym (*ang. mutual learning*). Działa on w następujący sposób:



Rysunek 1: Struktura sieci Tree Parity Machine [31]

1. podmioty biorące udział w synchronizacji uzgadniają wymiary sieci TPM poprzez wybór parametrów K, L, N;
2. obie sieci neuronowe są niezależnie inicjalizowane poprzez wylosowanie wartości wektora wag W, który spełnia wcześniej opisane warunki;
3. jeden z uczestników synchronizacji dokonuje wyboru binarnego wektora wejściowego X_i (i oznacza iterację omawianego algorytmu), a następnie wysyła go przez publiczny kanał do drugiego uczestnika;
4. obliczana jest odpowiedź obu struktur TPM dla wylosowanego wektora X_i, a następnie uczestnicy przesyłają sobie wzajemnie wartości odpowiedzi (odpowiednio O^A oraz O^B);
5. jeżeli odpowiedzi obu sieci spełniają następujący warunek $O^A = O^B$, następuje krok zbliżający obie sieci neuronowe do siebie nawzajem poprzez zastosowanie reguł uczących, które zostaną omówione w dalszej części;
6. kroki 3–5 są wykonywane do czasu osiągnięcia pełnej synchronizacji sieci neuronowych, to znaczy aż wektory wag W^A i W^B są identyczne.

Wspomniane reguły uczące są odpowiedzialne za aktualizację wag struktur TPM w sposób gwarantujący zakończenie synchronizacji w skończonym czasie. Istnieją trzy różne reguły uczące, które zostały przedstawione poniżej.

- *Reguła Hebbian*

$$w_{kn}(i+1) = w_{kn}(i) + O(i)x_{kn}(i)\Theta(y_k(i), O(i)) \tag{4}$$

- *Reguła Anti-Hebbian*

$$w_{kn}(i+1) = w_{kn}(i) - O(i)x_{kn}(i)\Theta(y_k(i), O(i)) \tag{5}$$

- *Reguła random walk*

$$w_{kn}(i+1) = w_{kn}(i) + x_{kn}(i)\Theta(y_k(i), O(i)) \tag{6}$$

W powyższych regułach zastosowana została funkcja $\Theta(a, b)$, która jest odpowiedzialna za sprawdzenie czy jej argumenty są sobie równe. W przypadku, gdy $a = b$, wówczas funkcja Θ zwraca wartość 1, natomiast wartość 0 jest zwracana w przeciwnym wypadku. Ponadto, przez parametr i rozumiana jest i-ta iteracja działania algorytmu.

W przypadku, gdy wartość wagi $w_{kn}(i+1)$ spełnia jeden z następujących warunków: $w_{kn}(i+1) < -L$ lub $w_{kn}(i+1) > L$, to wartość zostanie zamieniona odpowiednio na $-L$ i L. Ten krok zapewnia, że w każdym momencie wartość wagi w_{kn} będzie znajdować się w przedziale $< -L; L >$.

Protokół synchronizacji jest procesem niedeterministycznym i może zostać skończony w różnej liczbie iteracji. Na długość trwania uczenia wzajemnego mają wpływ parametry sieci TPM. Wraz ze wzrostem wartości każdego z parametrów omawianej struktury, proces wydłuża się [21]. Kolejnym czynnikiem, który może zarówno pozytywnie jak i negatywnie wpłynąć na czas trwania synchronizacji jest rozkład losowy będący podstawą wyboru wartości wektora wag W [6].

1.4.3 Zaproponowane usprawnienia

Bezpieczeństwo sztucznych sieci neuronowych typu TPM zależy od wielu czynników. Wśród nich warto wymienić: szybkość synchronizacji, dobór parametrów, jakość uzyskanego klucza kryptograficznego czy zdolność atakujących do przeprowadzenia skutecznego ataku man-in-the-middle [21]. W ramach badań prowadzonych na rzecz rozprawy doktorskiej zostało zaproponowane usprawnienie polegające na wykorzystaniu niebinarnych wektorów wejściowych podczas uczenia wzajemnego. Rozwiązanie to zostało dokładnie opisane w **publikacji nr 2 cyklu**.

Wykorzystanie niebinarnych wektorów wejściowych wiąże się z wprowadzeniem dodatkowego parametru M. Struktura wykorzystująca niebinarne wektory wejściowe jest nazywana siecią Nonbinary Tree Parity Machine (NBTPM). Nowy parametr może przyjmować wartości ze zbioru dodatnich liczb naturalnych ($M \in \mathbb{N}^+$). Wówczas wektor wejściowy X będzie składał się z elementów x_{kn} znajdujących się w przedziale od $-M$ do M. Taki wektor X będzie miał następującą postać: $X = [x_{11}, \ldots, x_{kn}]$, gdzie $x_{kn} \in \{x : x \in \mathbb{Z} \wedge -M \leq x \leq -1 \vee 1 \leq x \leq M\}$, a k i n oznaczają n-te wejście k-tego neuronu.

Parametr M nie wpływa na architekturę sztucznej sieci neuronowej typu TPM. Nieznaczne zmiany mają miejsce w algorytmie uczenia wzajemnego. Zmiany te są następujące:

1. uczestnicy wymiany klucza kryptograficznego, poza wcześniej zdefiniowanymi parametrami (K, L, N), muszą również uzgodnić wartość parametru M, która musi spełniać warunek $M \in \mathbb{N}^+$;

2. krok drugi pozostaje bez zmian;

3. jeden z uczestników synchronizacji dokonuje wyboru wektora wejściowego $X_i = [x_{11}, \ldots, x_{kn}]$, takiego, że $x_{kn} \in \{x : x \in \mathbb{Z} \wedge -M \leq x \leq -1 \vee 1 \leq x \leq M\}$, a następnie wysyła go przez publiczny kanał do drugiego uczestnika.

Kroki 4-5 pozostają bez zmian. Ponadto, przedstawione reguły uczące dalej znajdują zastosowanie i skutecznie synchronizują sieci typu NBTPM. W dalszej części pracy zostaną przedstawione skutki zastosowania struktury NBTPM oraz zaproponowany algorytm zwiększający bezpieczeństwo proponowanego rozwiązania.

1.4.4 Analiza bezpieczeństwa

Wykorzystanie niebinarnych wektorów wejściowych skutkuje przyspieszeniem procesu synchronizacji sieci neuronowych typu TPM. Jednak skutkuje to również wspomnianą degradacją jakości uzyskanego klucza kryptograficznego. Ponadto, przed wykorzystaniem sieci NBTPM wymagana jest dokładna analiza wpływu zaproponowanego parametru M na wszystkie czynniki związane z bezpieczeństwem uzgadniania klucza kryptograficznego.

W ramach pracy nad rozprawą doktorską przeprowadzone zostały analizy wpływu różnych wartości parametru M na:

- czas synchronizacji (liczbę iteracji uczenia wzajemnego),
- długość klucza kryptograficznego,
- charakterystykę uzyskanego klucza kryptograficznego,
- podatność na znane ataki kryptograficzne.

Poniżej przedstawione zostały symulacyjne wyniki wymienionych analiz. W związku z tym, że rysunki i tabele pochodzą z publikacji w języku angielskim, zachowano w nich oryginalne tytuły i zapisy ułamków dziesiętnych, używając kropki zamiast przecinka.

A. Czas synchronizacji

Główną zaletą zastosowania parametru M w sieciach neuronowych typu NBTPM jest skrócenie czasu synchronizacji, a w efekcie uzyskanie dłuższego klucza kryptograficznego w takim samym czasie w stosunku do struktur TPM. W tabeli 1 przedstawiono otrzymane wyniki symulacyjne dla parametrów $M \in \{1; 2; 3; 4; 5\}$ i $N \in \{40; 50; 60\}$. Wartości K i L były równe odpowiednio 3 i 5.

Można zauważyć, że wzrost parametru M znacząco wpływa na redukcję mediany liczb iteracji wymaganych do pełnej synchronizacji. W szczególności

początkowy wzrost parametru M skutkuje znaczącym obniżeniem wymaganych iteracji. Na podstawie wartości dla $M = 1$ i $M = 2$ można zauważyć, że mediana czasu synchronizacji dla każdego ze scenariuszy skróciła się co najmniej dwukrotnie.

Wyniki badań, zbieżne z powyższą obserwacją, rozważające szersze spektrum wartości parametrów sieci NBTPM zostały przedstawione w **publikacji nr 3 cyklu** [27].

Tabela 1: Czas synchronizacji NBTPM dla różnych wartości parametru M [30]

		Czas synchronizacji			
M	N	**Średnia**	**Minimum**	**Maksimum**	**Mediana**
1	40	709 ± 490	313	2176	648
2		290 ± 216	103	965	273
3		172 ± 138	75	484	156
4		114 ± 82	39	314	105
5		84 ± 64	28	249	78
1	50	714 ± 453	333	1981	666
2		316 ± 209	145	769	296
3		172 ± 126	69	455	158
4		118 ± 88	39	280	108
5		87 ± 64	28	216	82
1	60	733 ± 398	391	1763	715
2		320 ± 202	140	652	306
3		182 ± 122	79	421	172
4		121 ± 87	43	284	117
5		90 ± 70	34	233	81

B. Długość klucza kryptograficznego

W celu uzyskania klucza kryptograficznego, uczestnicy po ukończonej synchronizacji wykorzystują wektor W jako klucz kryptograficzny. W przypadku jednorodnego rozkładu wartości elementów wektora W długość klucza kryptograficznego wyrażona w bitach jest równa $len(W) = K \cdot N \cdot log(2L + 1)$.

Jednak już Andreas Ruttor w swojej rozprawie doktorskiej zauważył niejednorodność rozkładu wag po zakończonym procesie synchronizacji [19]. Ponadto, rozbieżność pomiędzy rozkładem wartości wektora W a rozkładem jednorodnym rośnie wraz ze wzrostem wartości parametru M. Efekt ten zostanie opisany w dalszej części rozprawy.

W celu estymacji efektywnej długości klucza kryptograficznego zaproponowana została funkcja, która za pomocą oszacowanej entropii, przybliża liczbę bitów, które można uzyskać po udanej synchronizacji sieci NBTPM. Wzór na metrykę $\hat{len}$ został podany w (7).

$$\hat{len} = \lfloor K \cdot N \cdot \hat{E}(W) \rfloor \tag{7}$$

W powyższym równaniu wartość $\hat{E}(W)$ oznacza szacunkową wartość entropii dla jednego wektora W. Ze względu na to, że wartość $E(W)$ nie jest znana a priori, szacowanie wartości $\hat{E}(W)$ następuje w wyniku symulacji. Ponadto, w celu dokładnego oszacowania wartości $\hat{E}(W)$, symulacje powinny być przeprowadzone dla każdego rozważanego zestawu parametrów (K, L, M, N).

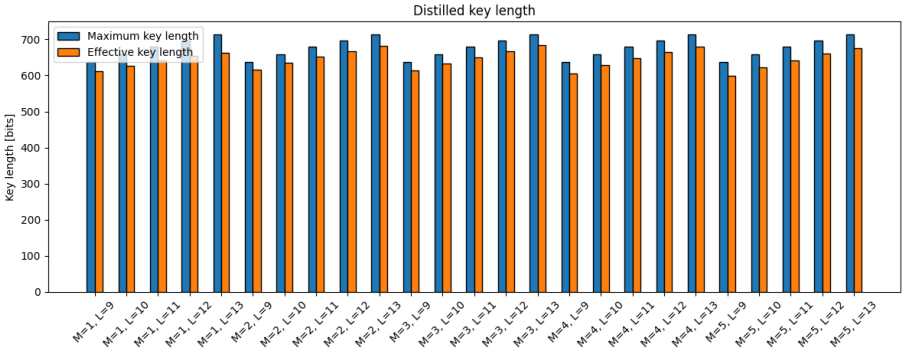
Wyniki symulacyjne wskazują, że efektywna długość klucza $\hat{len}$ jest odwrotnie proporcjonalna do wartości parametru M. Ponadto, można zauważyć, że wartość oszacowanej entropii $\hat{E}$ jest jedynie zależna od parametru M. Natomiast efektywna długość klucza rośnie wraz ze wzrostem parametrów N i L, ponieważ bezpośrednio wynika to ze wzoru (7). Pierwsze wyniki zostały zaprezentowane w Tabeli 2. Pogłębiona analiza dotycząca długości otrzymanego klucza kryptograficznego została zaprezentowana w **publikacji nr 4 cyklu**, z której m.in pochodzi rysunek 2.

Tabela 2: Entropia i oszacowana efektywna długość klucza [30]

M	N	**Entropia** ($\hat{E}$)	**Oszacowana efektywna długość klucza** ($\hat{len}$)
1	40	3.374	404
2		3.354	402
3		3.305	396
4		3.238	388
5		3.158	378
1	50	3.386	507
2		3.368	505
3		3.315	497
4		3.248	487
5		3.186	477
1	60	3.402	612
2		3.379	608
3		3.325	598
4		3.263	587
5		3.204	576

C. Charakterystyka uzyskanego klucza kryptograficznego

Uzgodniony klucz kryptograficzny powinien mieć odpowiednie właściwości statystyczne, aby jego wykorzystanie nie stanowiło potencjalnego słabego punktu zabezpieczonej komunikacji. W idealnych warunkach uzgodniony klucz charakteryzuje się idealną losowością, jednorodnością rozkładu, brakiem cykli oraz niezależnością od czynników zewnętrznych. Jednak w rzeczywistości najczęściej sto-



Rysunek 2: Efektywna długość klucza kryptograficznego [26]

sowane są generatory liczb pseudolosowych, które cechują się w pewien sposób deterministycznym charakterem działania.

Przeprowadzone badania symulacyjne w ramach pracy nad rozprawą doktorską wskazują, że wraz ze wzrostem wartości parametru M prawdopodobieństwo wystąpienia zdarzenia $w_{kn} = -L \vee w_{kn} = L$ w wektorze W rośnie, co zostało nazwane efektem wartości krańcowych [30]. Efekt ten został również zauważony dla binarnych wektorów wejściowych w [19].

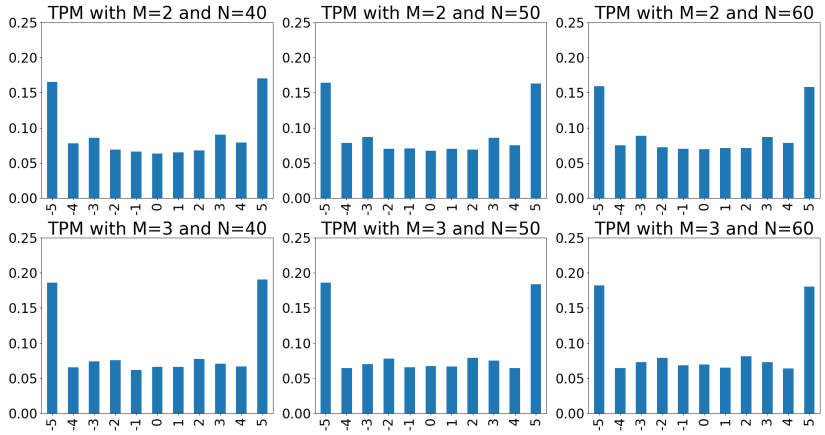
Niech $P(A)$ będzie tożsame z $P(w_{kn} \in \{-L; L\})$. W celu obliczenia $P(A)$, zacznijmy od oszacowania $P(w_{kn} = L)$. Niech parametr i oznacza rozważaną iterację. Zakładając krok synchronizacji zakończony aktualizacją wag, $P(w_{kn} = L)$ składa się z następującej sumy prawdopodobieństw warunkowych: $\sum_{m=0}^{M} P(w_{kn}(i+1)|x(i) = m)$. Rozważając pierwszy składnik tej sumy, możemy zauważyć, że $P(w_{kn}(i+1)|x(i) = 0) = 1$ tylko dla $w_{kn}(i) = L$. Podobna sytuacja następuje dla drugiego składnika rozważanej sumy, czyli $P(w_{kn}(i+1)|x(i) = 1) = 1$, wtedy i tylko wtedy, gdy $w_{kn}(i) = L$ lub $w_{kn}(i) = L - 1$. Wówczas możemy zauważyć, że liczba zdarzeń, które spełniają warunek jest równa $1+2+\cdots+M$. Natomiast moc zbioru zdarzeń elementarnych jest równa $(2M+1) \cdot (2L+1)$, ponieważ, zakładając równomierny rozkład wag i elementów wektora wejściowego, możliwa jest każda kombinacja wartości wag z przedziału $< -L; L >$ oraz wartości elementu wektora wejściowego z przedziału $< -M; M >$. Sytuacja $P(w_{kn} = -L)$ jest analogiczna do $P(w_{kn} = L)$. Biorąc pod uwagę powyższe fakty, prawdopodobieństwo $P(A)$ zostało przedstawione w równaniu (8).

$$\begin{aligned} P(A) &= (P(A|x_{kn} = 0) + \ldots + P(A|x_{kn} = M)) \\ &= 2 \cdot \left(\frac{1 + \ldots + (M+1)}{(2M+1)(2L+1)} \right) \\ &= \frac{(M+1)(M+2)}{(2M+1)(2L+1)} \end{aligned} \tag{8}$$

Warto zauważyć, że wraz ze wzrostem parametru M prawdopodobieństwo wystąpienia efektu wartości krańcowej zwiększa się, a w efekcie jakość klucza

kryptograficznego ulega degradacji. W związku z powyższym dobór wartości M jest kluczowy dla zapewnienia bezpieczeństwa wymiany klucza kryptograficznego z wykorzystaniem NBTPM, co zostanie podkreślone w dalszych częściach rozprawy. Powyższe wnioski zostały potwierdzone w dwóch publikacjach z cyklu [30, 26].

Rysunek 3 przedstawia nierównomierny rozkład wag dla parametrów $M = 2$ oraz $M = 3$ [30]. Ponadto, dalsza analiza została przedstawiona na rysunku 4, gdzie zaprezentowano rozbieżność pomiędzy rozkładem poszczególnych wartości wektora W, a rozkładem jednostajnym. Drugi wykres wyraźnie przedstawia, że wraz ze wzrostem parametru M, rozkład wag coraz bardziej odbiega od rozkładu jednostajnego.

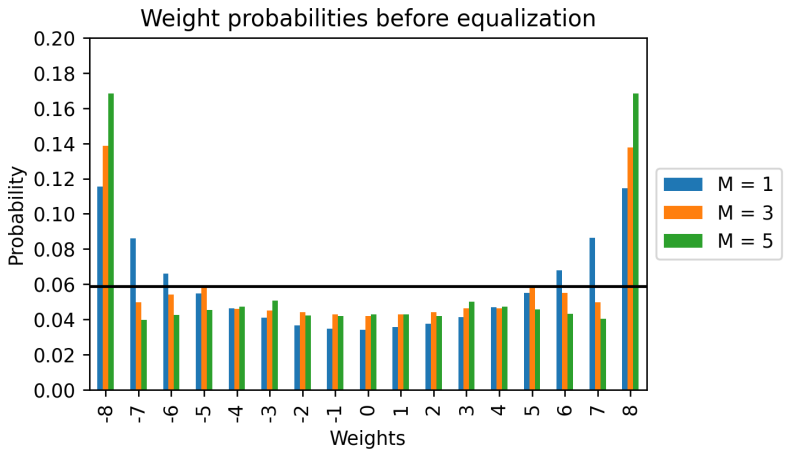


Rysunek 3: Prawdopodobieństwo wystąpienia wag w wektorze W dla parametrów $M = 2$ i $M = 3$ [30]

D. Podatność na znane ataki kryptograficzne

Algorytmy uzgodnienia klucza kryptograficznego powinny umożliwiać wymianę klucza przez niezabezpieczony kanał komunikacyjny. W takim medium mogą występować ataki typu man-in-the-middle, czyli ataki podczas których złośliwy podmiot jest w stanie podsłuchiwać całą transmisję podczas wymiany wiadomości. Atak skupiony jest na podsłuchu – nie rozważany jest scenariusz, w którym wysyłane wiadomości ulegają modyfikacji na skutek działania atakującego. Zabezpieczeniem przed taką sytuacją jest wzajemne uwierzytelnianie użytkowników i kontrola integralności, np. poprzez algorytmy kryptografii asymetrycznej i zastosowanie funkcji skrótu. W związku z powyższym, w celu oceny bezpieczeństwa struktury NBTPM rozważono podatność na bierny podsłuch.

Rozpatrzmy atak typu man-in-the-middle na synchronizację pomiędzy sieciami A i B. Niech złośliwy podmiot będzie posiadał swoją sieć TPM C i możliwość podsłuchiwania całego ruchu wymienionego pomiędzy A i B. Wówczas,



Rysunek 4: Rozbieżność pomiędzy rozkładem wag, a rozkładem jednostajnym [31]

w przypadku każdej iteracji protokołu uzgodnienia klucza kryptograficznego mogą wystąpić następujące sytuacje:

1. $O^A \neq O^B$ – sieci uzgadniające klucz kryptograficzny zwróciły różną odpowiedź, a w efekcie synchronizacja nie następuje, ani dla sieci A i B, ani dla atakującego C;
2. $O^A = O^B \neq O^C$ – sieci A i B zwróciły taką samą odpowiedź co skutkuje udaną iteracją synchronizacji, w takim przypadku atakujący nie aktualizuje swojej sieci C;
3. $O^A = O^B = O^C$ – wszystkie trzy sieci TPM odpowiedziały taką samą wartością, następuje aktualizacja wag we wszystkich sieciach, a rozważana sytuacja zbliża zarówno podmioty uzgadniające klucz, jak i atakującego, do udanej synchronizacji.

W celu udanej wymiany klucza podmioty powinny zrobić wszystko, aby zmaksymalizować częstotliwość występowania warunku 2, minimalizując jednocześnie występowanie sytuacji 3.

W przypadku protokołu uzgodnienia klucza kryptograficznego z wykorzystaniem sieci neuronowych NBTPM zdefiniowano cztery rodzaje ataków, które mogą zostać zastosowane podczas wykorzystania niezabezpieczonego medium [12]. Poniżej przedstawione są wspomniane ataki, wraz ze schematem ich działania.

- Klasyczny atak man-in-the-middle – atak w którym złośliwy podmiot nasłuchuje wymianę danych pomiędzy sieciami A i B, a następnie uczy swoją sieć zgodnie z algorytmem uzgodnienia klucza kryptograficznego, czyli tylko i wyłącznie w przypadku, gdy $O^A = O^B = O^C$. Skuteczność ataku rośnie wraz z ze wzrostem liczby złośliwych sieci neuronowych.

- Atak geometryczny – w przypadku zgodności odpowiedzi wszystkich sieci atak geometryczny jest identyczny jak man-in-the-middle. Natomiast w przypadku, gdy $O^A = O^B \neq O^C$, atakujący znajduje neuron o najmniejszej wartości iloczynu skalarnego wektora wag W i wektora wejściowego X. Po znalezieniu neuronu spełniającego ten warunek, jego wyjście σ jest zamieniane na $\sigma' = -\sigma$. Po takim podstawieniu następuje iteracja uczenia sieci C zakończona sukcesem.

- Atak większościowy – atak kryptograficzny w którym atakujący wykorzystuje wiele sieci TPM (załóżmy, że liczba sieci TPM atakującego to I). W sytuacji, gdy $O^A = O^B$, atakujący oblicza odpowiedź wszystkich sieci neuronowych, jednocześnie zapamiętując dla każdej z nich wektor odpowiedzi ukrytej warstwy $\sigma^I = (\sigma_1^I, \ldots, \sigma_K^I)$). Następnie znajdowany jest najczęściej występujący wektor σ i on zostaje wykorzystany do kontynuacji uczenia wszystkich sieci atakującego. Wraz z postępem synchronizacji, wszystkie sieci atakującego upodobniają się do siebie.

- Atak genetyczny – atak wykorzystujący nie więcej niż I sieci TPM. Podczas każdej iteracji atakujący tworzy dla każdej z istniejących już sieci jej 2^K wariantów i dla każdego z nich wybiera inną wartość wektora σ. Następnie synchronizuje sieci TPM z wykorzystaniem odpowiadającego jej wektora σ. W przypadku, gdy populacja jest zbyt liczna, funkcja dopasowania jest wykorzystywana do odrzucenia najgorszych jednostek w populacji.

Każda kolejna iteracja ataku ma za zadanie stopniowo dopasować wartości wag sieci TPM atakującego do wag sieci podmiotów, które dążą do ustalenia klucza kryptograficznego. W celu obliczenia podobieństwa złośliwej sieci zdefiniowany został współczynnik podobieństwa sieci TPM. Niech A i B oznaczają sieci, dla których będzie liczony współczynnik podobieństwa S. Dla tak zdefiniowanych sieci, wzór na współczynnik $S_{A,B}$ został przedstawiony w (9). Funkcja Θ została zdefiniowana w sekcji 1.4.2.

$$S_{A,B} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta(w_{kn}^A, w_{kn}^B)}{K \cdot N} \tag{9}$$

W **publikacji nr 5 cyklu** [29] przedstawiono symulacyjne wyniki synchronizacji sieci NBTPM o parametrach $M \in \{1; 2; 3; 4; 5\}, L \in \{9; 10; 11; 12; 13\}$ z wykorzystaniem niezabezpieczonego medium transmisyjnego. Podmiot atakujący uzgadnianie klucza kryptograficznego wykonywał wszystkie zdefiniowane powyżej ataki kryptograficzne. W tabeli 3 przedstawiono prawdopodobieństwo, że atakujący skutecznie zsynchronizuje swoją sieć TPM w wyniku przeprowadzenia każdego ze wspomnianych ataków.

Ataki zdefiniowane w [24, 20, 12, 18] są równie skuteczne na sieci NBTPM jak i na strukturę TPM. Można zauważyć nieznaczącą poprawę w odporności na ataki geometryczne i większościowe wraz ze wzrostem parametru L. Niestety atak genetyczny był skuteczny niemalże w każdej przeprowadzonej symulacji, jednak warto zauważyć że funkcja dopasowania była idealną funkcją oceniającą

poziom synchronizacji. W tym celu wykorzystano współczynnik S, do którego atakujący nie miałby dostępu w rzeczywistym ataku. Wzrost współczynnika M negatywnie wpływa na odporność struktury NBTPM na zdefiniowane ataki. Z drugiej strony, liczba wymaganych iteracji algorytmu znacząco spada, co jest widoczne w szczególności pomiędzy wartościami $M = 1$ i $M = 2$. Podczas, gdy dla tych wartości mediana iteracji zmalała ponad dwukrotnie, skuteczność ataków wzrosła o nie więcej niż 20%. Standardowy atak typu man-in-the-middle okazuje się nieskuteczny dla wartości $M \leq 4$. Dla $M = 5$, atakującemu sporadycznie udawało się zakończyć atak sukcesem.

Tabela 3: Liczba iteracji synchronizacji i skuteczność ataków.

Liczba iteracji						**Udane ataki**			
M	**L**	**Średnia**	**Mediana**	**Maks.**	**Min.**	**Standardowy**	**Większościowy**	**Geometryczny**	**Genetyczny**
1	9	2263.19 ± 1139.781	2182	5711	890	0.0	0.228	0.816	0.999
1	10	2823.69 ± 1413.413	2739	8478	1152	0.0	0.204	0.822	1.0
1	11	3475.25 ± 1814.419	3331	9087	1440	0.0	0.211	0.819	0.999
1	12	4198.83 ± 2209.911	4052	12608	1869	0.0	0.196	0.799	1.0
1	13	5064.37 ± 2820.405	4826	13067	1751	0.0	0.186	0.763	1.0
2	9	927.72 ± 500.816	893	2342	302	0.0	0.271	0.899	0.995
2	10	1131.29 ± 579.344	1098	2847	445	0.0	0.246	0.882	1.0
2	11	1353.59 ± 700.158	1306	3690	558	0.0	0.221	0.850	0.997
2	12	1617.92 ± 802.934	1568	4584	651	0.0	0.215	0.871	0.999
2	13	1908.82 ± 951.497	1839	4434	799	0.0	0.206	0.832	0.999
3	9	514.64 ± 308.012	489	1559	200	0.0	0.330	0.938	0.996
3	10	625.88 ± 348.143	598	1509	238	0.0	0.291	0.917	0.998
3	11	749.19 ± 413.828	714	1783	259	0.0	0.298	0.911	0.999
3	12	888.02 ± 487.737	853	2626	359	0.0	0.282	0.900	0.997
3	13	1031.67 ± 534.667	991	2616	435	0.0	0.235	0.892	0.999
4	9	331.62 ± 209.76	314	986	128	0.0	0.395	0.952	0.998
4	10	403.55 ± 245.221	383	1020	138	0.0	0.360	0.949	0.997
4	11	485.4 ± 289.757	461	1298	177	0.0	0.327	0.941	0.996
4	12	566.62 ± 325.635	544	1568	229	0.0	0.316	0.926	0.998
4	13	661.08 ± 367.354	631	1707	270	0.0	0.322	0.915	0.998
5	9	235.25 ± 158.482	219	695	85	0.003	0.452	0.963	0.998
5	10	285.76 ± 184.754	267	830	96	0.001	0.410	0.966	0.995
5	11	339.3 ± 214.747	320	1043	121	0.001	0.364	0.961	0.997
5	12	403.74 ± 254.932	379	1099	151	0.001	0.348	0.948	0.997
5	13	459.22 ± 272.12	438	1449	171	0.0	0.338	0.938	0.997

1.4.5 Algorytm wyrównujący wagi

Wzrost parametru M w sieci NBTPM skutkuje przyspieszoną synchronizacją, a w efekcie może prowadzić do uzgodnienia dłuższego klucza kryptograficznego. Jednak, rozkład wartości wektora w tak wygenerowanym kluczu kryptograficznym, wraz ze wzrostem parametru M jest coraz dalszy od rozkładu jednostajnego [30, 26]. W celu przeciwdziałania negatywnym aspektom towarzyszącym

strukturze NBTPM w **publikacji nr 6 cyklu** zaproponowany został algorytm wyrównujący wagi [31]. Proponowany algorytm składa się z trzech etapów: wyrównywanie wag, odrzucenia wartości, podstawienie. Opis faz algorytmu został przedstawiony poniżej.

A. Wyrównywanie wag

Pierwsza faza algorytmu polega na zamianie częściej występujących wartości w wektorze wag W na występujące rzadziej. W tym celu algorytm iteruje po każdej wartości zaczynając od początku wektora. Ponadto, zapamiętywana jest liczba dotychczasowych wystąpień każdej możliwej wartości. Podczas każdej iteracji i algorytm sprawdza czy rozważany element jest dotychczas najczęściej występującym elementem. Jeżeli tak, następuje podmiana i-tego elementu wektora W na wartość występującą dotychczas najrzadziej. Dla kilku wartości o równie małym prawdopodobieństwie, brana jest wartość najmniejsza. Formalny zapis fazy wyrównywania wag został przedstawiony w Algorytmie 1

Algorytm 1 Faza wyrównywania wag

Wejście: $W \leftarrow [w_{11}, \cdots, w_{1n}, w_{21}, \cdots, w_{kn}]$, K, N, L
Wyjście: W
  $F \leftarrow [f_{-L}, \cdots, f_L] = [0, \cdots, 0]$
  $k \leftarrow 1$
  dopóki $k \leq K$ **wykonaj**
    $n \leftarrow 1$
    dopóki $n \leq N$ **wykonaj**
      jeśli $W_{kn} \in argmax\, F$ **to**
        $w_{kn} \leftarrow min(argmin\, F)$
      koniec jeśli
      $F_{w_{kn}} \leftarrow F_{w_{kn}} + 1$
      $n \leftarrow n + 1$
    koniec dopóki
    $k \leftarrow k + 1$
  koniec dopóki

B. Odrzucenie wartości

Kolejną fazą algorytmu jest odrzucenie wag. Otrzymany klucz kryptograficzny jest dłuższy od efektywnej długości klucza [30], którą można uzyskać. W przypadku rozkładu jednorodnego, każda z wag niesie $log_2(2L + 1)$ informacji. Niestety, synchronizacja sieci TPM z natury generuje nierównomierny rozkład wartości wektora W, co jest dalej amplifikowane przez wykorzystanie NBTPM. Rozważana faza ma na celu odrzucić nadmiarowe wartości na podstawie wcześniej oszacowanej entropii $\hat{E}(W)$. W tym celu następuje szacowanie dla każdej iteracji jaka jest otrzymana długość klucza i czy nie przekracza efektywnej długości klucza możliwej do uzyskania. Przekroczenie tej wartości skutkuje

odrzuceniem rozważanej wagi. W tym celu wymagane jest oszacowanie wartości $\hat{E}(W)$ przed synchronizacją. Formalny zapis tej fazy jest przedstawiony w Algorytmie 2.

Algorytm 2 Faza odrzucenia wartości

Wejście: K, N, L, W przed wyrównaniem wartości
Wyjście: W'

```
E(W) = -∑_{l=-L}^{L} p_l log_2 p_l
len_current ← 0
W' ← () , będący wektorem zmiennej długości
k ← 1
dopóki k ≤ K wykonaj
  n ← 1
  dopóki n ≤ N wykonaj
    jeśli len_current < ((k − 1) · N + n) · E(W) to
      W'.append(w_kn)
      len_current ← len_current + log_2(2L + 1)
    koniec jeśli
    n ← n + 1
  koniec dopóki
  k ← k + 1
koniec dopóki
```

C. Faza podstawienia

Ostatni etap algorytmu to randomizacja sekwencji bitowych poprzez podstawienie. W tym celu wykorzystywane są funkcje skrótu, które charakteryzują się efektem lawinowym. Efekt ten powoduje, że zmiana choćby jednego bitu w argumencie funkcji skutkuje tym, że średnio połowa bitów wartości funkcji będzie się różnić.

W tym etapie wektor W' jest zamieniany na ciąg binarny. Następnie, w zależności od wykorzystanej funkcji skrótu H, ten ciąg jest jest dzielony na bloki o długości B, gdzie B oznacza długość binarnej wartości funkcji H. Kolejnym krokiem jest podmiana każdego z bloków na odpowiedź funkcji H dla tego bloku. Reszta bitów powstałych na skutek podziału wektora W' na bloki jest odrzucana

D. Weryfikacja

W celu zweryfikowania zaproponowanego algorytmu został przeprowadzony zestaw testów oceniający jakość uzyskanego klucza kryptograficznego. Testy przeprowadzane były przed i po wyrównywaniu wag. Zestaw testów składał się z 15 testów zaproponowanych przez NIST w dokumencie SP 800-22 Rev 1a [4], który definiuje testy oceniające własności generatorów liczb pseudolosowych. Testy sprawdzają między innymi: częstotliwość występowania poszczególnych wartości w całej sekwencji losowej jak i podsekwencjach, wyniki analizy spek-

tralnej, entropię źródła, występowanie podciągów w sekwencji czy możliwość kompresji z wykorzystaniem algorytmów bezstratnych. Dokłada lista testów została przedstawiona poniżej.

1. *Test częstotliwości (Frequency (Monobit) Test)* – sprawdza częstotliwość występowania zer i jedynek w ciągu binarnym.
2. *Blokowy test częstotliwości (Frequency Test within a Block)* – sprawdza częstotliwość występowania zer i jedynek w podciągach.
3. *Test serii (Runs Test)* – analizuje, czy oscylacja między jedynkami a zerami pozostaje w granicach dla losowego ciągu.
4. *Test najdłuższej serii jedynek w bloku (Test for the Longest Run of Ones in a Block)* – analizuje najdłuższą sekwencję takiej samej wartości bitu w ciągu binarnym
5. *Test rangi macierzy binarnej (Binary Matrix Rank Test)* – sprawdza liniową zależność pomiędzy podciągami rozważanej sekwencji binarnej.
6. *Test dyskretnej transformaty Fouriera (Discrete Fourier Transform (Spectral) Test)* – analizuje wynik transformaty Fouriera dla rozważanej sekwencji
7. *Test dopasowania szablonu bez nakładania (Non-overlapping Template Matching Test)* – sprawdza częstotliwość występowania wcześniej zdefiniowanej sekwencji bitów.
8. *Test dopasowania szablonu z nakładaniem (Overlapping Template Matching Test)* – działanie testu jest analogiczne do testu poprzedniego, jednak rozważana jest sekwencja bit po bicie, a nie co długość bloku.
9. *Test Mauera* – test analizuje wynik bezstratnego algorytmu kompresji dla sekwencji losowej.
10. *Test złożoności liniowej (Linear Complexity Test)* – ocenia wymaganą długość rejestru przesuwnego z liniowym sprzężeniem zwrotnym zdolnego wygenerować rozważany ciąg binarny.
11. *Test serii* – sprawdza częstotliwość występowania wszystkich permutacji M-bitowych sekwencji.
12. *Test przybliżonej wartości entropii (Approximate Entropy Test)* – celem testu jest porównanie częstotliwości nakładających się bloków o dwóch kolejnych długościach.
13. *Test sum skumulowanych (Cumulative Sums (Cusum) Test)* – test ocenia czy sumy skumulowane rozważanego ciągu binarnego mieszczą się w dopuszczalnym zakresie. Test jest uruchamiany zarówno od przodu, jak i od tyłu sekwencji.

14. *Test losowego spaceru (Random Excursions Test)* – sprawdza ile razy konkretny stan został odwiedzony podczas losowego spaceru z sumą skumulowaną. Rozważane są stany od -4 do 4 z pominięciem 0.

15. *Test losowego spaceru, drugi wariant (Random Excursions Variant Test)* – test jest taki sam jak poprzedni, jednak w tym przypadku rozważane są stany od -9 do 9 z pominięciem 0.

W celu uzyskania klucza kryptograficznego przeprowadzono 1000 symulacji sieci NBTPM o następujących parametrach: $K = 3$, $L = 8$, $M = \{1; 3; 5\}$, $N = 60$. Dla uproszczenia zmiany wartości wagi na wektor binarny usunięto wartości równe 0 w wektorze W, co skutkowało, że każda waga mogła być zapisana jako 4 bitowy wektor. Wynikiem testu jest wartość P, która powinna być większa od $0,01$ dla testu zaliczonego pozytywnie. Wyniki przed i po zastosowaniu algorytmu równoważenia wag przedstawiono w tabeli 4. Testy 14 oraz 15 obliczane są dla różnych wartości stanów, stąd wyniki zawierają wiele wartości. Dla testu 14 wyniki posortowane są od stanu -4 do 4, a dla testu 15 wyniki posortowano od -9 do 9.

Wyniki testów wskazują, że wykorzystanie algorytmu równoważącego wagi skutkuje poprawą charakterystyki otrzymanego klucza kryptograficznego. Przed zastosowaniem rozważanego algorytmu jedynie testy 5, 10, 14 i 15 zwróciły wynik pozytywny. Nawet dla parametru $M = 1$, czyli dla klucza kryptograficznego otrzymanego z synchronizacji standardowej sieci TPM, większość testów zwróciła wynik negatywny. Natomiast, po zastosowaniu algorytmu wyrównującego wagi jedynie testy 3, 7 i 13 dla $M = 1$, testy 2 i 4 dla $M = 3$ oraz testy 12 i 14 dla $M = 5$ miały wartość P zbliżoną do negatywnego wyniku, czyli wartości mniejszej od $0,01$ [31].

Dodatkowo w **publikacji nr 7 cyklu** sprawdzono, że działanie algorytmu wyrównującego wagi nie zależy od wyboru parametrów rozważanej sieci neuronowej [28]. Ponadto, wyniki testów NIST 800-22 rev. 1 wskazują, że klucz kryptograficzny otrzymany przy użyciu klasycznej sieci TPM nie jest ciągiem prawdziwie losowym. W efekcie każda sieć neuronowa wykorzystująca uczenie wzajemne, niezależnie od jej wariantu, może czerpać korzyści z zastosowania zaproponowanego algorytmu wyrównującego wagi.

1.5 Podsumowanie

Autoreferat przedstawia protokół uzgadniania klucza kryptograficznego bazujący na sztucznych sieciach neuronowych o strukturze TPM oraz autorskie usprawnienia, które skutkują poprawą jego bezpieczeństwa. Zaproponowane usprawnienie polega na wykorzystaniu niebinarnych wektorów wejściowych podczas procesu wzajemnej synchronizacji sieci neuronowych TPM. W wyniku badań opracowany został również algorytm wyrównujący wagi otrzymywane w procesie synchronizacji, którego zastosowanie skutkuje poprawą jakości otrzymanego klucza kryptograficznego.

Wykorzystanie niebinarnych wektorów wejściowych powoduje przyspieszenie procesu synchronizacji, a w efekcie umożliwia wygenerowanie dłuższego klucza

Tabela 4: Wyniki zestawu testów NIST przed i po działaniu algorytmu [31]

Test	Wartość P					
	$M = 1$		$M = 3$		$M = 5$	
	przed	po	przed	po	przed	po
1)	0.0	0.961	0.0	0.665	0.0	0.397
2)	0.0	0.66	0.0	0.011	0.0	0.734
3)	0.0	0.063	0.0	0.686	0.0	0.155
4)	0.0	0.726	0.0	0.073	0.0	0.394
5)	0.461	0.587	0.791	0.953	0.104	0.491
6)	0.0	0.171	0.0	0.623	0.0	0.517
7)	0.0	0.052	0.0	0.369	0.0	0.668
8)	0.0	0.443	0.0	0.268	0.0	0.462
9)	0.0	0.518	0.0	0.135	0.0	0.903
10)	0.126	0.813	0.294	0.337	0.535	1.0
11)	(0.0, 0.711)	(0.522, 0.788)	(0.0, 0.028)	(0.717, 0.452)	(0.0, 0.0)	(0.713, 0.667)
12)	0.0	0.3	0.0	0.477	0.0	0.025
13) (w przód)	0.0	0.676	0.0	0.812	0.0	0.746
13) (w tył)	0.0	0.63	0.0	0.758	0.0	0.583
14)	0.845, 0.278, 0.89, 0.915, 0.626, 0.24, 0.082, 0.25	0.982, 0.402, 0.113, 0.56, 0.685, 0.933, 0.721, 0.109	0.014, 0.0, 0.341, 0.021, 0.514, 0.883, 0.211, 0.053	0.418, 0.351, 0.631, 0.638, 0.93, 0.642, 0.423, 0.468	0.926, 0.932, 0.232, 0.579, 0.326, 0.877, 0.046, 0.021	0.394, 0.216, 0.085, 0.396, 0.028, 0.346, 0.805, 0.478
15)	0.42, 0.374, 0.357, 0.414, 0.512, 0.71, 0.783, 0.722, 0.538, 0.806, 0.67, 0.545, 0.816, 0.712, 0.436, 0.357, 0.325, 0.339	0.088, 0.104, 0.207, 0.411, 0.537, 0.564, 0.331, 0.113, 0.108, 0.745, 0.63, 0.949, 0.914, 0.65, 0.569, 0.437, 0.358, 0.481	0.903, 0.949, 1.0, 1.0, 0.803, 0.637, 1.0, 0.386, 0.317, 0.617, 0.773, 0.655, 0.705, 0.677, 0.546, 0.579, 0.606, 0.628	1.0, 0.974, 0.967, 0.821, 0.588, 0.755, 0.56, 0.236, 0.293, 0.453, 0.174, 0.268, 0.684, 0.707, 0.734, 0.934, 0.821, 0.552	0.878, 0.87, 0.861, 1.0, 1.0, 0.811, 0.671, 0.855, 0.527, 0.752, 0.715, 0.888, 0.72, 0.673, 0.634, 0.661, 0.683, 0.701	0.53, 0.33, 0.258, 0.228, 0.195, 0.259, 0.224, 0.142, 0.141, 0.948, 0.175, 0.103, 0.3, 0.539, 0.401, 0.265, 0.308, 0.287

kryptograficznego w takim samym czasie w porównaniu do oryginalnej struktury TPM. Skutkiem zastosowania proponowanego usprawnienia jest amplifikacja efektu wartości granicznych, czyli nierównomiernego rozkładu wartości klucza kryptograficznego otrzymanego z zsynchronizowanej sieci neuronowej. W efekcie otrzymany klucz posiada rozkład wartości odbiegający od rozkładu jednorodnego, a jego efektywna długość jest krótsza niż długość klucza w zapisie binarnym. Dzięki zaproponowanemu algorytmowi równoważącemu wagi po procesie synchronizacji istnieje możliwość przeciwdziałania efektowi wartości granicznych, a przetworzony klucz kryptograficzny posiada cechy ciągu losowego o rozkładzie jednostajnym. Tym samym postawiona teza, że **wykorzystanie niebinarnych wektorów wejściowych oraz algorytmu wyrównującego wagi usprawnia proces synchronizacji w sieciach neuronowych o strukturze TPM, jednocześnie zachowując bezpieczeństwo otrzymanego klucza kryptograficznego** została udowodniona, a wyznaczony cel rozprawy został osiągnięty.

Wyniki przeprowadzonych analiz bezpieczeństwa oraz wydajności wskazują, że odpowiedni dobór parametrów sieci TPM czy NBTPM jest krytycznym aspektem skutecznego ustalania bezpiecznych kluczy kryptograficznych. Badania wskazują, że nieumiejętny wybór rozmiaru sztucznej sieci neuronowej pozwala złośliwym podmiotom odtworzyć część lub nawet całość ustalonego sekretu. Równie istotny jest świadomy wybór parametru M, charakteryzującego niebinarne wektory wejściowe. Z tego względu w ramach niniejszej rozprawy przeprowadzono szereg badań związanych z zaproponowanym rozwiązaniem.

Zysk płynący z zastosowania niebinarnych wektorów uczących przewyższa straty jakie to rozwiązanie implikuje. Zaproponowane potencjalne zastosowania dowodzą, że nawet w środowisku o ograniczonych zasobach obliczeniowych lub w celu redukcji błędów w kwantowej dystrybucji klucza sieci TPM są w stanie relatywnie szybko się zsynchronizować tworząc klucz kryptograficzny. Przedstawione usprawnienia zwiększają konkurencyjność sieci neuronowych w procesie uzgadniania kluczy kryptograficznych i mogą przyczynić się do stosowania tego rozwiązania na większą skalę.

1.6 Bibliografia

[1] 2024 Quantum Computing Report: The Current & Future State — quera.com. `https://www.quera.com/blog-posts/current-and-future-state-of-quantum-computing`. [Accessed 15-09-2024].

[2] A. M. Allam, H. M. Abbas,, M. W. El-Kharashi. Authenticated key exchange protocol using neural cryptography with secret boundaries. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013.

[3] American National Standards Institute. *ANSI X9.63-1999: Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography*, 1999. ANSI X9.63-1999.

[4] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray,, S. Vo. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. 2010.

[5] W. Diffie, M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[6] M. Dolecki, R. Kozera. *The Impact of the TPM Weights Distribution on Network Synchronization Time*, page 451–460. Springer International Publishing, 2015.

[7] T. Dong, T. Huang. Neural cryptography based on complex-valued neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4999–5004, November 2020.

[8] H. Gomez, Ó. Reyes,, E. Roa. A 65nm cmos key establishment core based on tree parity machines. *Integration*, 58:430–437, 2017.

[9] S. Jeong, C. Park, D. Hong, C. Seo,, N. Jho. Neural cryptography based on generalized tree parity machine for real-life systems. *Security and Communication Networks*, 2021:6680782, Feb 2021.

[10] I. Kanter, W. Kinzel,, E. Kanter. Secure exchange of information by synchronization of neural networks. *Europhysics Letters*, 57(1):141, jan 2002.

[11] W. Kinzel, R. Metzler,, I. Kanter. Dynamics of interacting neural networks. *Journal of Physics A: Mathematical and General*, 33(14):L141, apr 2000.

[12] A. Klimov, A. Mityagin,, A. Shamir. *Analysis of Neural Cryptography*, page 288–298. Springer Berlin Heidelberg, 2002.

[13] R. Metzler, W. Kinzel,, I. Kanter. Interacting neural networks. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, 62(2 Pt B):2555–2565, 2000.

[14] National Institute of Standards and Technology. *Module-lattice-based key-encapsulation mechanism standard*, August 2024. FIPS 203.

[15] M. Niemiec. Error correction in quantum cryptography based on artificial neural networks. *Quantum Information Processing*, 18(6), 2019.

[16] M. Niemiec, A. Dziech, M Stypiński,, J. Derkacz. Quantum-based solutions for the next-generation internet. *Information & Security: An International Journal*, 43(1):62–72, 2019.

[17] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[18] M. Rosen-Zvi, E. Klein, I. Kanter,, W. Kinzel. Mutual learning in a tree parity machine and its application to cryptography. *Physical Review E*, 66(6), December 2002.

[19] A. Ruttor. *Neural Synchronization and Cryptography.* PhD thesis, University of Würzburg, 2007.

[20] A. Ruttor, W. Kinzel, R. Naeh,, I. Kanter. Genetic attack on neural cryptography. *Phys. Rev. E*, 73:036121, Mar 2006.

[21] É. Salguero Dorokhin, W. Fuertes,, E. Lascano. On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key. *Security and Communication Networks*, 2019:1–10, March 2019.

[22] A. Sarkar, J. Dey,, A. Bhowmik. Multilayer neural network synchronized secured session key based encryption in wireless communication. *Indonesian Journal of Electrical Engineering and Computer Science*, 14(1):169, 2019.

[23] A. Sarkar, Mohammad Z. Khan, M. M. Singh, A. Noorwali, C. Chakraborty,, S. K. Pani. Artificial neural synchronization using nature inspired whale optimization. *IEEE Access*, 9:16435–16447, 2021.

[24] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter,, W. Kinzel. Cooperating attackers in neural cryptography. *Physical Review E*, 69(6), June 2004.

[25] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[26] M. Stypiński, M. Niemiec. Impact of nonbinary input vectors on security of tree parity machine. In Andrzej Dziech, Wim Mees,, Marcin Niemiec, editors, *Multimedia Communications, Services and Security*, pages 94–103, Cham, 2022. Springer International Publishing.

[27] M. Stypiński. Analiza zależności pomiędzy parametrami sieci tpm a czasem synchronizacji. *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, 1(4):169–172, 08 2022.

[28] M. Stypiński. Wpływ parametrów sieci typu tree parity machine na bezpieczeństwo algorytmu wyrównującego wagi. *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, 4:81–84, 09 2024.

[29] M. Stypiński, M. Niemiec. Security of neural network-based key agreement protocol for smart grids. *Energies*, 16(10), 2023.

[30] M. Stypiński, M. Niemiec. Synchronization of tree parity machines using nonbinary input vectors. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):1423–1429, January 2024.

[31] M. Stypiński, M. Niemiec. Weight equalization algorithm for tree parity machines. *Computer Networks*, 257:110988, 2025.

[32] M. Volkmer, S. Wallner. Tree parity machine rekeying architectures. *IEEE Transactions on Computers*, 54(4):421–427, 2005.

2 Zbiór publikacji

W tym rozdziale przedstawiono zbiór spójnych tematycznie artykułów naukowych będących główną częścią rozprawy doktorskiej. W tabeli 5 przedstawiono procentowy udział autora w publikacji, liczbę punktów ministerialnych oraz wskaźnik Impact Factor czasopisma. Oświadczenia współautorów zostały dołączone na końcu tego rozdziału.

Tabela 5: Procentowy udział w publikacjach będących częścią cyklu

Artykuł	**Punkty MNiSW**	**Udział procentowy**
Niemiec, M., Dziech, A., Stypiński, M., Derkacz, J., „Quantum-Based Solutions for the Next-Generation Internet", Information & Security: An International Journal, vol. 43, no. 1, pp. 62-72, 2019. doi: 10.11610/isij.4306	5	25%
Stypiński M., Niemiec M., „Synchronization of Tree Parity Machines Using Nonbinary Input Vectors", IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 1, pp. 1423-1429, 2024. doi: 10.1109/TNNLS.2022.3180197 *Impact Factor (2023) = 10.2*	200	75%
Stypiński, M., „Analiza zależności pomiędzy parametrami sieci TPM a czasem synchronizacji", Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne, SIGMA-NOT, sp. z.o.o., t. 4, str. 169-172, 2022. doi: 10.15199/59.2022.4.35	20	100%
Stypiński, M., Niemiec, M., „Impact of Nonbinary Input Vectors on Security of Tree Parity Machine", Communications in Computer and Information Science, Springer International Publishing, pp. 94-103, 2022. doi: 10.1007/978-3-031-20215-5_8	80	90%
Stypiński, M., Niemiec, M., „Security of Neural Network-Based Key Agreement Protocol for Smart Grids", Energies, MDPI, vol. 16, no. 10: 3997, 2023. doi: 10.3390/en16103997 *Impact Factor (2023) = 3.0*	140	80%
Stypiński M., Niemiec M., „Weight equalization algorithm for tree parity machines", Computer Networks, vol. 257, 2025. doi: 10.1016/j.comnet.2024.110988 *Impact Factor (2023) = 4.4*	100	80%
Stypiński, M., „Wpływ parametrów sieci typu Tree Parity Machine na bezpieczeństwo algorytmu wyrównującego wagi", Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne, SIGMA-NOT, sp. z.o.o., t. 4, str. 83-86, 2024. doi: 10.15199/59.2024.4.13	20	100%

Information & Security

Published since 1998

Niemiec, Dziech, Stypiński & Derkacz,
vol. 43, no. 1 (2019): 62-72
https://doi.org/10.11610/isij.4306

ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

Quantum-based Solutions for the Next-generation Internet

Marcin Niemiec* (✉), *Andrzej Dziech*, *Miłosz Stypiński, Jan Derkacz

AGH University of Science and Technology, Mickiewicza 30, 30-059 Krakow, Poland
http://agh.edu.pl

ABSTRACT:

Quantum physics influences modern computer science and communications. We observe new quantum-based solutions which are being implemented in practical networks. This article introduces these techniques and explains how they can change the next-generation Internet and communication. The basics of quantum mechanics and such effects as birefringence and entanglement are briefly introduced. Next, the most popular quantum-based solutions which are good candidates for modern services in the next-generation Internet are described. The quantum key distribution process, as well as the idea and validation of a quantitative approach to security in quantum cryptography, are presented. In addition to security methods, quantum-based solutions for data processing and transmission are explained: superdense coding, the medium access control protocol with quantum entanglement and the idea of quantum computing. The final section concludes the article and indicates current initiatives and future directions.

ARTICLE INFO:

RECEIVED: 19 SEP 2019
REVISED: 28 AUG 2019
ONLINE: 17 SEP 2019

KEYWORDS:

quantum techniques, next-generation Internet, communications, security

✉ Corresponding Author: Tel.: +48 126173937; E-mail: niemiec@kt.agh.edu.pl

Introduction

Interest in quantum-based mechanisms for communications is growing rapidly. Solutions such as quantum cryptography or quantum random number generators are not just theories, but are being used in practice. However, these mechanisms are not very widespread as yet; they will probably form part of the next-generation Internet.

The main advantages of these methods are security and performance. Quantum-based cryptography and true random number generators increase security to a level unachieved by any previous solutions. Superdense coding and other techniques based on quantum entanglement can improve the bit-rate of communication systems. It seems that even technological problems such as effective quantum repeaters will be solved in the near future. Then, quantum-based solutions will be widely implemented in practical networks.

This article presents a few quantum-based solutions which are developing rapidly. Some of them are already being used in practice. Others are still theoretical models and have not been implemented yet. Nevertheless, all of them aspire to be components of the next generation of communications.

Quantum Basics

Quantum mechanics is a special branch of physics. It describes the behaviour of matter and energy at the atomic and subatomic levels. Nowadays, some features of quantum mechanics are being used to design new methods of improving the security and performance of modern communication networks. At the beginning of this article, some of the principles and models which have a direct connection with quantum-based solutions for communications will be presented.

Qubit

A *bit* is a basic term of communications and computer science. It is a unit of computer information which can have only two possible values: 0 or 1. A unit of quantum information is called a *qubit* (quantum bit). A qubit can have two possible values (normally 0 or 1). However, it can also be a superposition of both. This superposition can be presented as a vector in the Bloch sphere (Figure 1) – an abstract sphere with antipodal opposite points representing two specific states. Two extreme opposite points in the sphere are conventionally written as $|0\rangle$ and $|1\rangle$. Any quantum state is called *ket* and presented as $|\psi\rangle$.

We can assume that each state in quantum mechanics is represented as a vector in Hilbert space. Then, two basic states, $|0\rangle$ and $|1\rangle$ are orthonormal and we can define a given qubit as a superposition of two orthonormal quantum states:

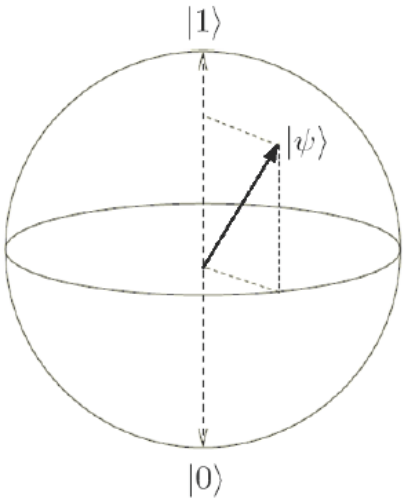


Figure 1: A qubit in the Bloch sphere.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where:

$$|\alpha|^2 + |\beta|^2 = 1$$

This means that when we perform a measurement on a qubit, we obtain the state $|0\rangle$ with the probability $|\alpha|^2$ or the state $|1\rangle$ with the probability $|\beta|^2$. Therefore, the sum of these probabilities equals 1. This physical interpretation means that the quantum state before the measurement could be in superposition, but the measurement may damage this state. The object's state will be determined by the measurement and will be equal to $|0\rangle$ or $|1\rangle$. This observation leads directly to the Heisenberg Uncertainty Principle.

Heisenberg Uncertainty Principle

In 1927, Werner Heisenberg published one of the fundamental principles of quantum physics.[1] He proved that it is not possible to measure accurately both the position and momentum of a physical system. These quantities can only be determined with some characteristic uncertainty. The mathematical notation of this principle is presented as follows:

$$\Delta x \cdot \Delta p \geq \frac{h}{4\pi}$$

where Δx and Δp are measurement uncertainties of a particle's position and momentum respectively, and h is Planck's constant $(h \approx 6.626 \cdot 10^{-34}\, J \cdot s)$. This well-known equation shows that measuring the position of a particle with high precision makes it impossible to measure momentum precisely. In general, the Uncertainty Principle states that it is impossible to measure some physical quantities with the same, high precision. Position and momentum are only one example of these pairs. Another example is the polarization of photons.

Birefringence

Birefringence is the decomposition of light into two rays when it passes through certain materials – for example, calcite crystals. The decomposition depends on the polarization of the photons. We can assume that:

- horizontally-polarized photons appear in the upper ray,
- vertically polarized photons appear in the lower ray.

Diagonally polarized light (45°) is decomposed into two rays: some photons appear in the upper ray and the rest will be observed in the lower ray of the calcite crystal. However, if we pass a single photon with the polarization 45°, it cannot be simultaneously vertical and horizontal at the output. It 'chooses' only one of the polarizations, with a probability of ½. An experiment with polarized photons and a birefringent calcite crystal is presented in Figure 2. Using the considered calcite crystal, we are able to measure perfectly the photons with vertical and horizontal polarizations (0° and 90°). However, we lose information about diagonally-polarized photons.

Entanglement

Usually, the quantum state of single particles is independent of others. However, we can produce pairs of particles which interact in a very interesting way: if we measure the state of one particle, then the state of the second particle can be fully determined. This means that we need only measure one particle to know the states of both. Additionally, the states of the particles are completely random before measurement occurs. The entangled state of two different particles (indices 1 and 2) can be presented as:

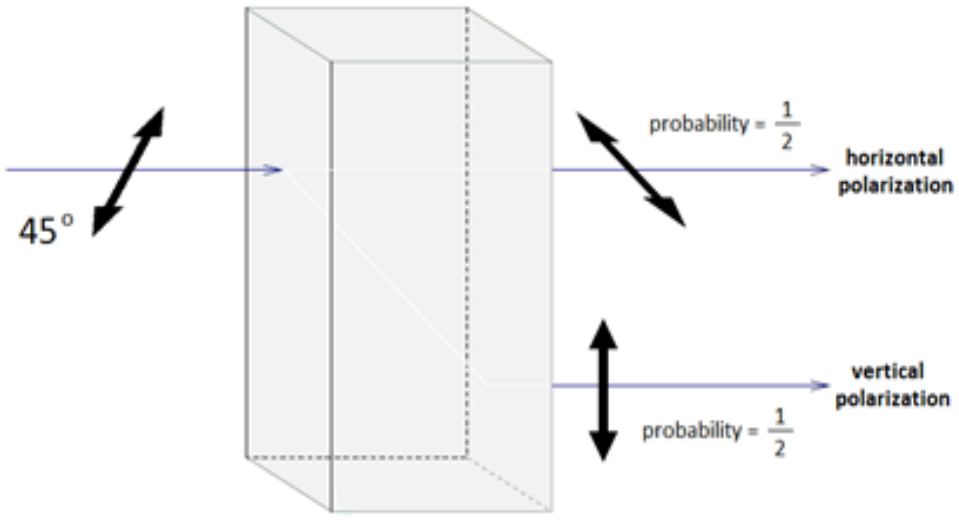


Figure 2: Polarized photon (45°) entering a birefringent calcite crystal.

$$\frac{1}{\sqrt{2}}(|\updownarrow\rangle_1 |\leftrightarrow\rangle_2 + |\leftrightarrow\rangle_1 |\updownarrow\rangle_2)$$

This means that if the first particle has vertical polarization, the second has horizontal polarization, and if the first particle has horizontal polarization, the second has a vertical polarization. Both possibilities can appear with a probability ½. It should be noted that the entanglement phenomenon still retains this feature, even if the particles are separated by great distances.

Quantum-based Solutions

Interest in quantum effects for computer science and communications continues to grow. In this section, the most popular quantum-based solutions are presented. They are good candidates for modern services in the next generation of the Internet and communications.

Quantum key distribution

Secure distribution or agreement of encryption keys are crucial to data confidentiality. Currently, when we use modern ciphers with popular key distribution methods, we are not sure if an intruder is eavesdropping on the communication. In this way, a hidden intruder can scan the network and obtain sensitive data. Quantum key distribution ensures a very high level of security, because it is not possible to eavesdrop on the communication in a passive way.[2] If an eavesdropper reads the distributed key, this will change the quantum states of the photons and will thus be revealed. This is possible because measurement influences the quantum state, and it is not possible to clone an unknown quantum state.

Popular quantum key distribution protocols, such as BB84,[3] are based on the polarization of single photons, which carry information coded in quantum states (i.e. different polarizations: vertical, horizontal, diagonal). In this way, the recipient and potential eavesdropper do not know which detector should be used to measure the polarization precisely. It is not a problem for the intended recipient – when they announce the configuration of a detector which was used during the measurement of a received photon, the sender confirms that the obtained result is correct or asks for this bit to be deleted from the final key because the obtained result is not certain.

What about eavesdropping on a quantum distributed key? If the eavesdropper chooses an inappropriate detector to perfectly measure the polarization, the polarization of the photon is changed. Sender and recipient uncover the eavesdropper if they compare part of the obtained key. In this way, passive eavesdropping is not possible. If someone wants to eavesdrop photons and read confidential information, they will change the quantum states of the photons.

Quantum cryptography

Quantum key distribution ensures a very high level of security. However, it is only part of the complete key establishment process. For example, the sender and recipient must estimate errors in the distributed key by computing the Quantum Bit Error Rate (QBER). The QBER is defined as the ratio of the number of wrong bits to the total number of bits. It is worth emphasizing that not only Eve is responsible for errors – they may occur because of disturbance in the quantum channel, optical misalignment, noise in detectors, and so on.
After the bit error estimation, Alice and Bob use key distillation protocols. These protocols usually involve two steps:

- key reconciliation – in this step sender and recipient must find and correct or delete occurred errors,
- privacy amplification – sender and recipient should strengthen their privacy and construct the final key by deleting some of the distributed bits.

However, effective management of security and efficiency in quantum cryptography is needed. To solve this problem, Niemiec and Pach considered security in a quantitative way.[4, 5] This approach is crucial when we want to map different end-user requirements to a quantum cryptography system. Using this strategy, end-users can choose an appropriate security level. In reference to the measure of information introduced by Hartley,[6] the measure of security *J(k)* in quantum key distribution was proposed. This function was defined as:

$$J(k) = \log \frac{k}{n}$$

where *log* represents the natural logarithm, *k* is the number of uncovered bits, and *n* is the length of the key. Additionally, the entropy of security and function of entropy of security *S(k)* in quantum cryptography were defined. This function is analogous to the entropy defined by Shannon [7] and was defined as:

$$S(k) = -\frac{k}{n} \log \frac{k}{n}$$

If we divide the function *S(k)* by the number of bits *n*, we obtain a general relationship which we can use to control the security and efficiency of a system with quantum cryptography. Additionally, this function has one maximum in the value 0.1 (it corresponds to the situation when we uncover and compare approx. 37 % bits of the distributed key). Therefore, we are able to define two different levels of security: basic and advanced. This idea makes it possible to personalize end-users' security.

The approach was verified by numerous simulations [8] performed with a QKD Simulator tool.[9] The simulator calculated the difference between real values of QBER (called *true_QBER*) and QBER calculated using the proposed

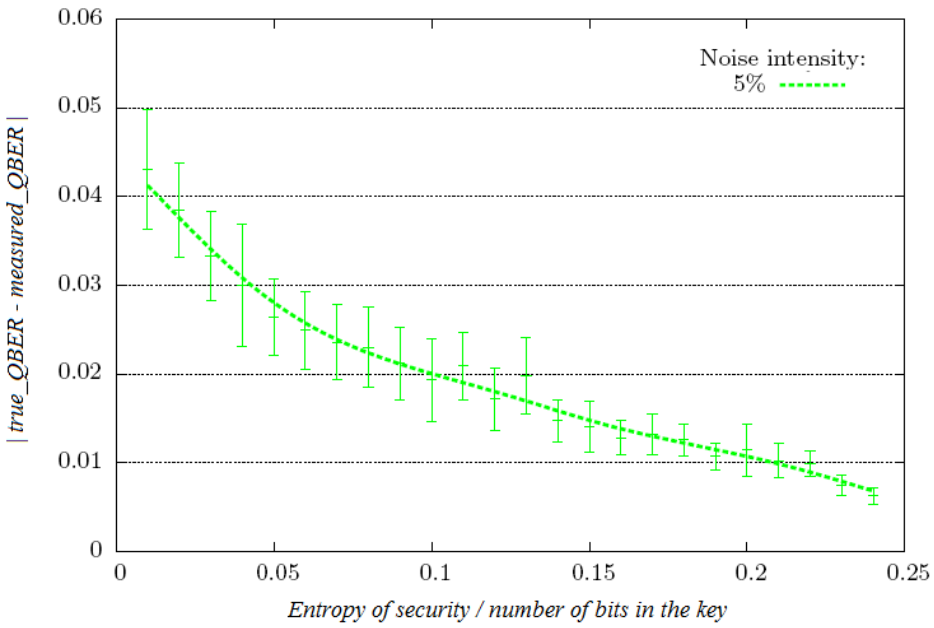


Figure 2: Validation of a quantitative approach to security in quantum cryptography (simulations of QBER).[8]

method (called *measured_QBER*). During the simulations, the length of all distributed keys was 1000 bits; however, the eavesdropped bits were different, ranging between one and 1000 bits. Figure 3 presents the results of simulations with noise intensity of 5% – typical noise in real quantum channels. The function in the graph is decreasing, with the most significant changes (exponential curve) observed at the basic security level (for values smaller than 0.1). Smaller changes (linear curve) are observed at the advanced security level (for values greater than 0.1). This means that at the basic security level, security increases faster than at the advanced level.

Quantum random number generator

The generation of random numbers plays a crucial role in many modern applications, mainly used by the security services. The security of encrypted data is based on the strength of its keys. Therefore, the keys must be generated randomly. Otherwise, an intruder can predict the key and read confidential data. Random numbers are also needed in other applications – during numerical simulations of complex systems, in the gaming industry, etc. Nowadays, software generators are commonly used. Unfortunately, computers are deterministic systems, and they produce pseudo-random numbers. Therefore, it is impossible for a software program to generate a sequence of truly random numbers.

Because of the principles of physics, quantum-based generators are an excellent source of randomness. One example is a photon with polarization 45° entering a birefringent calcite crystal (Figure 2). We have already mentioned that a photon with the polarization 45° 'chooses' only one of two possible polarizations (0° or 90°) with a probability of ½. We can put detectors at the outputs of

the crystal and detect the resultant photons. Horizontally-polarized photons appearing in the upper ray can generate bit 0, and vertically polarized photons appearing in the lower ray can generate bit 1. This can be an excellent random number generator.

Another example is transmission upon a semi-transparent mirror; this solution is implemented in practice.[10] A single photon can be reflected or transmitted. It is intrinsically random, and any external parameters do not influence this process. Using this technique, sequences of truly random numbers can be produced.

Quantum communications

Quantum effects support not only data security, but also the efficiency of communication networks. An example of such a solution is superdense coding. Using superdense coding it is possible to send two bits of classical information using just one qubit.[11] To achieve this improvement, both the sender and recipient must share an entangled pair of photons. If the sender wants to send a two-bit message (four different possibilities: 00, 01, 10 or 11), they perform a single qubit operation on their qubit. This operation transforms the qubit into one of four orthonormal states – according to the message which the sender wants to send. Now, the transformed qubit is sent to the recipient, who can perform a measurement on the joined pair and obtain the two-bit message.

Another example proposes applying quantum effects to ALOHA, a well-known medium access control (MAC) protocol. There, entangled pairs of photons can improve the capacity utilization of shared media. In systems using slotted ALOHA, the time dimension is divided into time slots.[12] At the beginning of each slot, users decide whether to send a packet or not. For two users, the packet will be successfully delivered if only one user transmits in a single time slot. If neither user sends a packet, or both users attempt to send their packets in the same time slot, the transmission is lost. Therefore, only half the time slots will be utilized on average. How can this be improved? Sandor Imre presents a system using the ALOHA protocol and shared entangled pairs of photons.[13] In the proposed system, even the packet collision and the unused channel carry information. Therefore, sending one classical bit and using quantum entanglement, it is possible to transmit as many as 2.5 classical bits on average. This significantly improves the efficiency of communication.

Quantum computer

The evolution of quantum mechanics emerged into another direction of quantum-based solutions which is quantum computing. The operation of a quantum computer is essentially different from a standard one. Instead of operating on bits, a quantum computer uses qubits as a basic mean of data in its registers. Hence, the state of the register can be presented as a superposition of all possible values of an n-length vector from $\{0,1\}^n$ space,[14] which can be defined as:

$$|\psi\rangle = \sum_{x=0}^{n} \alpha_x |x\rangle$$

where:

$$\sum_{x=0}^{n} |\alpha_x|^2 = 1$$

Not fully deterministic nature of quantum computer creates new opportunities as well as new threats to computing and communications. Regarding a few types of computing problems, the quantum computer enables the usage of quantum algorithms which solve the task in a more efficient manner. Example problems which are possible to solve on the quantum computer are searching the unstructured database using Grover's algorithm [15] and integer factorization and finding a discrete algorithm using Shor's algorithm.[16]

The latter algorithm is a potential threat to global cybersecurity. Using Shor's algorithm implementation on a powerful quantum computer could break the fundamental security of commonly used asymmetric cryptography. However, the post-quantum cryptography is a trend that aims to develop quantum-proof security algorithms.

Further development

The transition towards quantum communication and the Internet can be divided into few milestones. Each of them is an important improvement in the quality and security of quantum-based solutions.[17]

Currently, the technology allows us to create a trusted repeaters network in order to perform QKD. This solution is based on a trusted node between the sender and receiver, which extends the maximum distance between the communicating parties and removes the need for a full mesh network. The next steps to achieve end-to-end quantum communication are to prepare and measure networks. These networks enable the transmission of prepared one-qubit state to any node in the network. Security-wise this step is a significant improvement due to the fact that the intermediary node is superseded. This means QKD is available between any nodes in the network since any eavesdropping disturbs the quantum state of transmitted qubits.

Further development of quantum technology could result in the ability to share entangled qubits between any two nodes in the network. Here, the nodes are assumed to have a capability of deterministically measuring the state of the qubit. More advanced technology would enable nodes to store the state of the qubit for some time using quantum memory. This would allow performing blind quantum computing, secret sharing, and time-limited clock synchronization. The last step is the transition from each node being capable of performing small few-qubit fault-tolerant operations to full quantum computing.

Conclusions

Security is one of the most important challenge of modern telecommunications.[18] Entirely new ways of solving problems of security and performance are

currently being developed. Laws of physics ensure that every eavesdropper can be uncovered, bit rates can be increased, truly random numbers can be generated, etc. Some of these solutions are already being implemented; however, they are not very popular because such communication systems need to work under special conditions (i.e. direct quantum channel between the sender and the recipient). Other techniques are still at the theoretical model stage and have not been applied yet. Nevertheless, quantum-based solutions allow users to improve the security and performance of communication networks. Therefore, quantum-based services should become popular solutions in the next-generation Internet.

Acknowledgement

This work was supported by the ECHO project which has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement no 830943.

References

[1] Werner Heisenberg, "Uber den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik," *Zeitschrift fur Physik*, 43 (1927): 172–198.

[2] Gilles Van Assche, *Quantum Cryptography and Secret-Key Distillation* (Cambridge: Cambridge University Press, 2006).

[3] Charles H. Bennett and Gilles Brassard, "Public Key Distribution and Coin Tossing," *IEEE International Conference on Computers, Systems, and Signal Processing* (1984), 175–179.

[4] Marcin Niemiec and Andrzej R. Pach, "The Measure of Security in Quantum Cryptography," *IEEE Global Telecommunications Conference - GLOBECOM* (2012).

[5] Marcin Niemiec and Andrzej R. Pach, "Management of security in quantum cryptography," *IEEE Communications Magazine* 51, no. 8 (2013): 36–41.

[6] Ralph V. Hartley, "Transmission of information," *Bell System Technical Journal* 7 (1928): 535–563.

[7] Claude E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal* 27 (1948): 379–423, 623–653.

[8] Marcin Niemiec, "Design, Construction and Verification of a High-Level Security Protocol Allowing to Apply the Quantum Cryptography in Communication Networks," Ph.D. Thesis, AGH University of Science and Technology, 2011.

[9] Marcin Niemiec, Łukasz Romański and Marcin Święty, "Quantum cryptography protocol simulator," Multimedia communications, services and security, volume 149 of Communications in Computer and Information Science, Springer Berlin Heidelberg (2011): 286–292.

[10] ID Quantique White paper, "Random number generation using quantum physics," April 2010.

[11] Charles H. Bennett and J. Wiesner, "Communication via one- and two-particle operators on Einstein–Podolsky–Rosen states," *Phys. Rev. Lett.* 69, (1992): 2881-2884.

[12] Lawrence G. Roberts, "ALOHA Packet System With and Without Slots and Capture," *Computer Communications Review* 5, no 2 (1975): 28–42.

[13] Sandor Imre, "Quantum Hyperdense Coding for Distributed Communications," *Quantum Physics*, *arXiv:1210.2856* (2012).

[14] Artur Ekert, Patrick Hayden, and Hitoshi Inamori, "Basic concepts in quantum computation," *Quantum Physics, arXiv:quant-ph/0011013* (2010).

[15] Lov K. Grover, "A fast quantum mechanical algorithm for database search," *Quantum Physics, arXiv:quant-ph/9605043* (1996).

[16] Peter W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA (1994): 124-134.

[17] Stephanie Wehner, David Elkouss, Ronald Hanson, "Quantum Internet: A Vision for the Road Ahead," *Science* 362, no 6412 (2018).

[18] Paweł Korus and Andrzej Dziech, "Efficient Method for Content Reconstruction with Self-Embedding," *IEEE Transaction on Image Processing*, vol. 22, no 3 (2013): 1134 - 1147.

About the Authors

Marcin **Niemiec** was awarded his PhD and PhD Hab. in telecommunications in 2011 and 2019 respectively. Currently he works as assistant professor at the Department of Telecommunications, AGH University of Science and Technology, Poland. His research interests focus on cybersecurity. He has been involved in numerous European projects (FP6, FP7 and H2020). He is the co-author of over 80 publications. E-mail: niemiec@kt.agh.edu.pl.

Andrzej **Dziech** received his PhD.Hab. in telecommunications. He is full professor since 1986. His fields of interest are related to digital communication, image and data processing, information and coding theory, intelligent monitoring, data protection. He is the author of 240 publications, including 5 books and he was supervisor of 19 Ph.D. students. He was coordinator of many national and international projects, including the FP7 project INDECT.

Miłosz **Stypiński** received his B.SC in ICT from AGH University of Science and Technology, Krakow, Poland in 2018. Now, he is a M.Sc. student of ICT at AGH University of Science and Technology. His current research interests focus on cybersecurity and data analysis.

Jan **Derkacz** is a staff member of the Department of Telecommunications at AGH University of Science and Technology. He actively participated in several national and international research projects in the area of Information and Communications Technologies and Security.

Synchronization of Tree Parity Machines Using Nonbinary Input Vectors

Miłosz Stypiński and Marcin Niemiec

***Abstract*— Neural cryptography is the application of artificial neural networks (ANNs) in the subject of cryptography. The functionality of this solution is based on a tree parity machine (TPM). It uses ANNs to perform secure key exchange between network entities. This brief proposes improvements to the synchronization of two TPMs. The improvement is based on learning ANN using input vectors that have a wider range of values than binary ones. As a result, the duration of the synchronization process is reduced. Therefore, TPMs achieve common weights in a shorter time due to the reduction of necessary bit exchanges. This approach improves the security of neural cryptography.**

***Index Terms*— Artificial neural networks (ANNs), key agreement, mutual learning, neural cryptography, security.**

I. Introduction

Secure key agreement is one of the basic steps in secure channel establishment. The algorithms responsible for the key exchange must ensure that no eavesdroppers are able to reproduce the secure key. Applied key agreement protocols are based on mathematical operations which have no computationally efficient inversion, for example, factorization of a large number of problems or other derived problems.

Quantum computing poses a real threat to applied cryptography systems. Currently used algorithms, based on the public-key cryptography approach, offer conditional security. Efficient derivation of a secure key from exchanged fragmentary information may break the security of the key agreement protocol. Currently, there is one known algorithm—Shor's algorithm—capable of factorizing large numbers. Hence, it can extract exchanged keys and break all applied asymmetric cipher cryptography [1]. However, the successful implementation of this algorithm requires a quantum computer with a sufficient number of qubits.

Some modern cryptography techniques—such as quantum cryptography and neural cryptography—are able to overcome this problem and provide a variety of quantum-proof algorithms. The tree parity machine (TPM) is one such solution. It achieves a key agreement functionality by mutual learning of two artificial neural networks (ANNs). Mutual learning cannot be reduced to either primes factorization or discrete logarithm problems, hence it is not susceptible to quantum computing.

This brief proposes an acceleration to the mutual learning process of TPMs. The following novelties are considered:

1) introduction of nonbinary input vectors used for the synchronization of TPM;
2) proposal of a new parameter M describing input vector;
3) verification of the proposed solution in the insecure environment (with eavesdropper);
4) analysis of observed phenomenon called extrema values effect.

The proposed improvement results in a shorter synchronization, which at the same time enhances security properties.

This brief is structured as follows. Section II presents related works relevant to the subject of this brief. Section III outlines the architecture of TPMs, the process of mutual learning, secure key agreement protocol, exchanged key length, and security of TPMs. Section IV describes the nonbinary input TPM architecture, entropy, and its appliance in terms of quality assessment of exchanged key. Additionally, Section IV presents observed phenomena called extrema values effect. Section V describes the methodology of the performed simulations and an analysis of the gathered results.

Manuscript received 22 April 2021; revised 30 October 2021 and 28 February 2022; accepted 22 May 2022. Date of publication 13 June 2022; date of current version 5 January 2024. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the ECHO Project under Grant 830943. *(Corresponding author: Miłosz Stypiński.)*

The authors are with the Institute of Telecommunications, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: stypinski@agh.edu.pl; niemiec@agh.edu.pl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3180197.

Digital Object Identifier 10.1109/TNNLS.2022.3180197

II. Related Works

Kanter *et al.* [2] and Rosen-Zvi *et al.* [3] in 2002 successfully performed key agreement via mutual learning. After that, various efforts took place to enhance the proposed TPMs as a competitive security solution. Santhanalakshmi *et al.* [4] proposed the usage of a genetic algorithm in finding optimal weight vectors for training TPM. This solution reduces the synchronization time but increases the computational complexity. Allam *et al.* [5] proposed improvement based on a shared secret. As a result, the complexity of known attacks is increased while maintaining the same synchronization time. Another usage scenario was described in [6] where TPM was used as an error correction mechanism for key agreed in quantum cryptography system. Another approach significantly changing the TPM architecture is introduced in [7]. By using the original whale optimization algorithm for synchronization and by adding another hidden layer into the neural network, authors achieve higher security compared to standard TPM. Dong and Huang [8] describe enhancement achieved by learning TPM with complex-valued input vectors. This solution is generalized in [9], where vector-valued inputs are proposed. A different approach is described by Sarkar [10], where chaos-generated input vectors are utilized.

III. Tree Parity Machine

ANNs are increasingly popular, finding applications in fields including security. Kanter *et al.* [2] and Rosen-Zvi *et al.* [3] introduce a novel approach for the key agreement functionality implemented with neural networks, which are explained in more detail in this section.

A. Tree Parity Machine Architecture

A TPM is a two-layered perceptron-structured ANN with discrete weights, binary input, and binary output [11]. The input vector $X = [x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{k1}, \ldots, x_{kn}]$, $K, N, k, n \in \mathbb{N} \wedge k \leq K \wedge n \leq N$ has KN elements, where K denotes the number of inputs for each neuron in the first layer, and N indicates the number of neurons in the first layer. Every element x_{kn} of input vector X can have one of two possible values, either -1 or 1.

The first layer consists of neurons similar to the McCulloch–Pitts model [12]. Every input x_{kn} is connected to the kth neuron and has its corresponding weight. The values of the weights are the only difference from the former model. Every weight w_{ij} can take a value

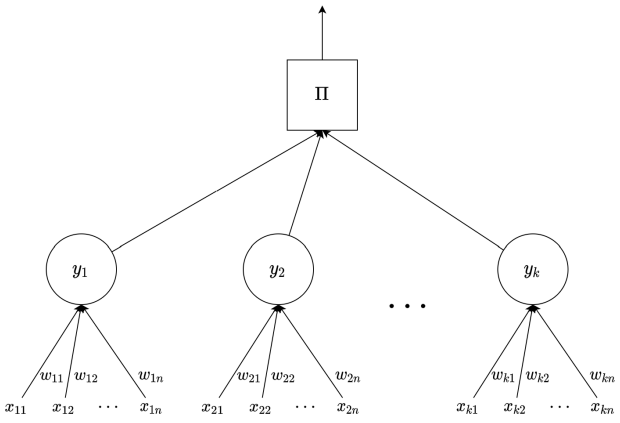


Fig. 1. Architecture of the TPM.

between $-L$ and L, where $L \in \mathbb{Z}$ is the parameter of the TPM and denotes the minimum/maximum possible weight value of the input neurons.

The output of the aforementioned neurons is based on the slightly changed signum function σ. The formula of the function is presented in (1). It differs from the regular signum function in that it never returns zero. The value of 0 is mapped either to 1 or -1, based on whether the side is the sender or the recipient of the communication [13]. The recipient and sender side is denoted by r and s, respectively. The parties decide beforehand which side is the sender and the recipient

$$\sigma\left(x^{r/s}\right) = \begin{cases} 1, & x^r \geqslant 0 \vee x^s > 0 \\ -1, & x^r < 0 \vee x^s \leqslant 0. \end{cases} \tag{1}$$

The argument for the neuron's activation function is the sum of the products of the input vector's elements with corresponding weight. The exact formula is presented in the following equation:

$$y_k = \sigma\left(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}\right). \tag{2}$$

The final result O of the TPM is the product of each of the outputs from hidden neurons from the first layer (3)

$$O^{r/s} = \prod_{k}^{K} y_k^{r/s}. \tag{3}$$

The overall architecture of the TPM is shown in Fig. 1.

B. Key Agreement Protocol

The parties performing the key agreement execute the protocol, which results in the secure shared key known only to the participating parties. This is usually achieved through the exchange of some information through an unsecured channel and by performing mathematical operations whose results are only known to the authorized parties [14]. The first and most popular key agreement protocol was proposed by Diffie and Hellman [15].

TPM offers functionality that can be adopted for key exchange purposes. The protocol for two parties consists of the following steps [13].

1) Both participants must agree on all the parameters for the TPM (K, L, N) and initialize their TPMs with random weights.
2) The key agreement participants publicly exchange a previously chosen binary random input vector X.
3) Each party computes the output from their TPM and publishes the results.
4) If the outputs match, both participants apply the appropriate learning rule that updates the weights of TPM accordingly.
5) Steps 2–4 are repeated until full synchronization of both TPMs is achieved.

The full synchronization is equivalent to every corresponding weight of both TPMs being equal to each other, at which point both TPMs are the same.

The aforementioned learning rules are responsible for updating the weights of each TPM in such a way that the synchronization process finishes in finite time [16]. There are three different learning rules that can be used in the process of updating weights [17].

1) *Hebbian Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) + O(t)x_{kn}(t)\Theta(y_k(t), O(t)). \tag{4}$$

2) *Anti-Hebbian Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) - O(t)x_{kn}(t)\Theta(y_k(t), O(t)). \tag{5}$$

3) *Random Walk Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) + x_{kn}(t)\Theta(y_k(t), O(t)) \tag{6}$$

where $\Theta(a, b)$ denotes the function returning 1 if $a = b$ and 0 otherwise, and parameter t denotes the iteration in the key agreement algorithm.

The synchronization process of two parity machines is not a deterministic algorithm. The number of iterations is not fixed and depends on the size and parameters of the TPM. However, it is shown that the time is finite and can be easily estimated by users [18]. The process takes longer for larger TPM sizes (K and N) and maximum weight value (L). Other factors that affect the number of iterations required for two TPMs to finish mutual learning include distribution of initial weights and learning rule [19].

C. Security of Tree Parity Machines

Security of key agreement protocol is crucial for communication. Any eavesdropper being able to reproduce the key based on the messages exchanged between parties or any other source breaks the security of the channel. Subsequently, such a situation depreciates the secure key exchange protocol. Hence, it is crucial to assess the security of any novel algorithm or protocol.

TPMs has been studied extensively. Javurek and M. Turčaník [18] and Martínez Padilla *et al.* [20] identify four distinct types of attacks that TPM may be vulnerable to as follows.

1) *Brute force attack:* Research shows that it is impossible to find the exact key as a result of a brute force attack against TPMs in polynomial time.
2) *Genetic algorithm for weight prediction:* It has been shown that only TPMs with a single neuron in the second layer are vulnerable to this type of attack.
3) *Man-the-middle interception attack:* Studies show that on average 60% of weights were synchronized in the eavesdropper's TPM.
4) *Sign of weight classification using neural networks:* Martínez Padilla *et al.* [20] demonstrate that classification using ANNs has near 100% accuracy in determining the sign of the weight in the TPM, which reduces the time needed by the brute force attack by almost half.

The studies show that, by utilizing these attack vectors, it is possible to gain some information about the key. Hence, cryptosystems should be aware of this threat and counteract it in order to minimize the likelihood of key reconstruction.

D. Man-in-the-Middle Attack

Synchronization of two TPMs without additional layers of security is a process prone to man-in-the-middle attacks. This attack relies on the possibility of placing a node C between parties A and B performing a key agreement. The node eavesdrops on all the messages shared between A and B. Based on information collected, node C may be able to gain unauthorized access to information sent between A and B. Moreover, if the nodes are not mutually authenticated, the adversarial party may be able to alter the messages accordingly to attempt an attack with a higher probability of success.

In terms of TPMs, man-in-the-middle attacks come down to capturing all the input vectors X and outputs of parties being intercepted. An adversarial TPM performs the learning process on acquired data. There are three scenarios to be considered while intercepting the key exchange. Let A and B be the parties wishing to exchange the key and let C be an intruder able to perform a man-in-the-middle attack. The three scenarios are as follows.

1) If $\Pi_A \neq \Pi_B$, no TPMs is synchronized during this step.
2) If $\Pi_A = \Pi_B \neq \Pi_C$, only TPMs A and B are synchronized, while TPM C (attacker) does not update its weights.
3) If $\Pi_A = \Pi_B = \Pi_C$, all the TPMs update their weights accordingly.

The last scenario brings the adversarial party closer to obtaining the exchanged key. Hence, this situation should be avoided at all costs.

IV. Nonbinary Input Vectors

The TPMs use binary vectors X for input [2] during the synchronization process. This brief introduces a new approach: nonbinary input vectors used to synchronize TPMs for a secure key agreement protocol. The authors propose that the mutual learning process that uses the vectors with a greater range of possible values of every element influence the synchronization time of two TPMs. Simulations performed in Section IV verify this proposition and indicate that this approach can significantly increase the security of neural cryptography.

A. Nonbinary Vector Tree Parity Machine Architecture

So far, the exact TPM was defined by parameters K, L, N. In this brief, the authors introduce a new parameter M, denoting the minimum/maximum value of each element of input vector X. Hence, the input vector will have the following form: $X = [x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{k1}, \ldots, x_{kn}]$, where $x_{kn} \in \{x : x \in \mathbb{Z} \wedge -M \leq x \leq -1 \vee 1 \leq x \leq M\}$. Thus, during the synchronization process, the entities can use nonbinary input vectors, instead of binary vectors which are currently used in practical implementations.

Introducing the M parameter does not affect the architecture of the TPM. The new learning process slightly differs from the original. The changes are in the first two steps of the key agreement protocol presented in Section III-B. The first two steps now read as follows.

1) Both synchronization participants must agree on all the parameters for the TPM (K, L, M, N) and initialize their TPMs with random weights.
2) The key agreement participants publicly exchange a previously chosen random input vector X, which now consists of values ranging from $-M$ to M.

Points 3–5 remain unchanged.

Furthermore, formulas shown in Section III are still valid despite more divergent values of the learning vectors. However, simulations presented in Section V show that as the input vectors are more differentiated, the distribution of settled keys is less similar to the uniform distribution. Therefore, unbiased estimation of key length is required.

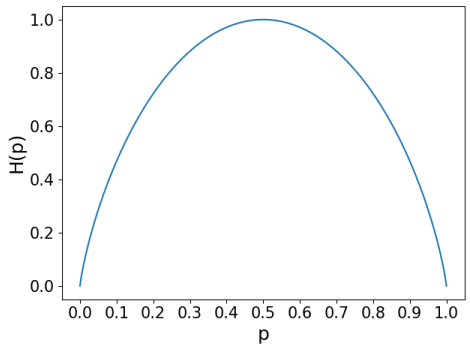


Fig. 2. Entropy of the source generating two different values with the same probability.

TABLE I
Synchronization Time of TPMs With Differentiated Input Vectors

M	N	Synchronization time			
		Average	**Minimum**	**Maximum**	**Median**
1	40	709 ± 490	313	2176	648
2		290 ± 216	103	965	273
3		172 ± 138	75	484	156
4		114 ± 82	39	314	105
5		84 ± 64	28	249	78
1	50	714 ± 453	333	1981	666
2		316 ± 209	145	769	296
3		172 ± 126	69	455	158
4		118 ± 88	39	280	108
5		87 ± 64	28	216	82
1	60	733 ± 398	391	1763	715
2		320 ± 202	140	652	306
3		182 ± 122	79	421	172
4		121 ± 87	43	284	117
5		90 ± 70	34	233	81

B. Synchronized Key Quality

The quality of random numbers generation has a significant impact on the final security of the cryptosystem. A true random number generator produces every available output with equal probability. Unfortunately, computers are incapable of generating fully random numbers. Frequently, numbers are generated based on a pseudorandom number generator. This requires a seed supplied beforehand, which is the starting point of the pseudorandom number sequence, and each further number depends on it. Many contemporary implementations lack important features like good mathematical foundations, lack of predictability, and cryptographic security [21].

Entropy is one of the measures which assesses the quality of the generated numbers. Let us assume the random source generates I different numbers $\alpha_1, \alpha_2, \ldots, \alpha_i$ with corresponding probabilities $p_1, p_2, \ldots, p_i$. Entropy for such a defined source is presented in the following equation [22]:

$$H(p_1, p_2, \ldots, p_i) = -\sum_{i=1}^{I} p_i \log_j p_i. \tag{7}$$

The base of logarithm j denotes the units in which entropy is measured, for example, for 2 and e units are bits and nats, respectively [23].

Let us consider a random source that produces two outputs with, either 0 or 1 with corresponding probabilities $P(X = 0) = p$ and $P(X = 1) = 1 - p$. The entropy for the described source is presented

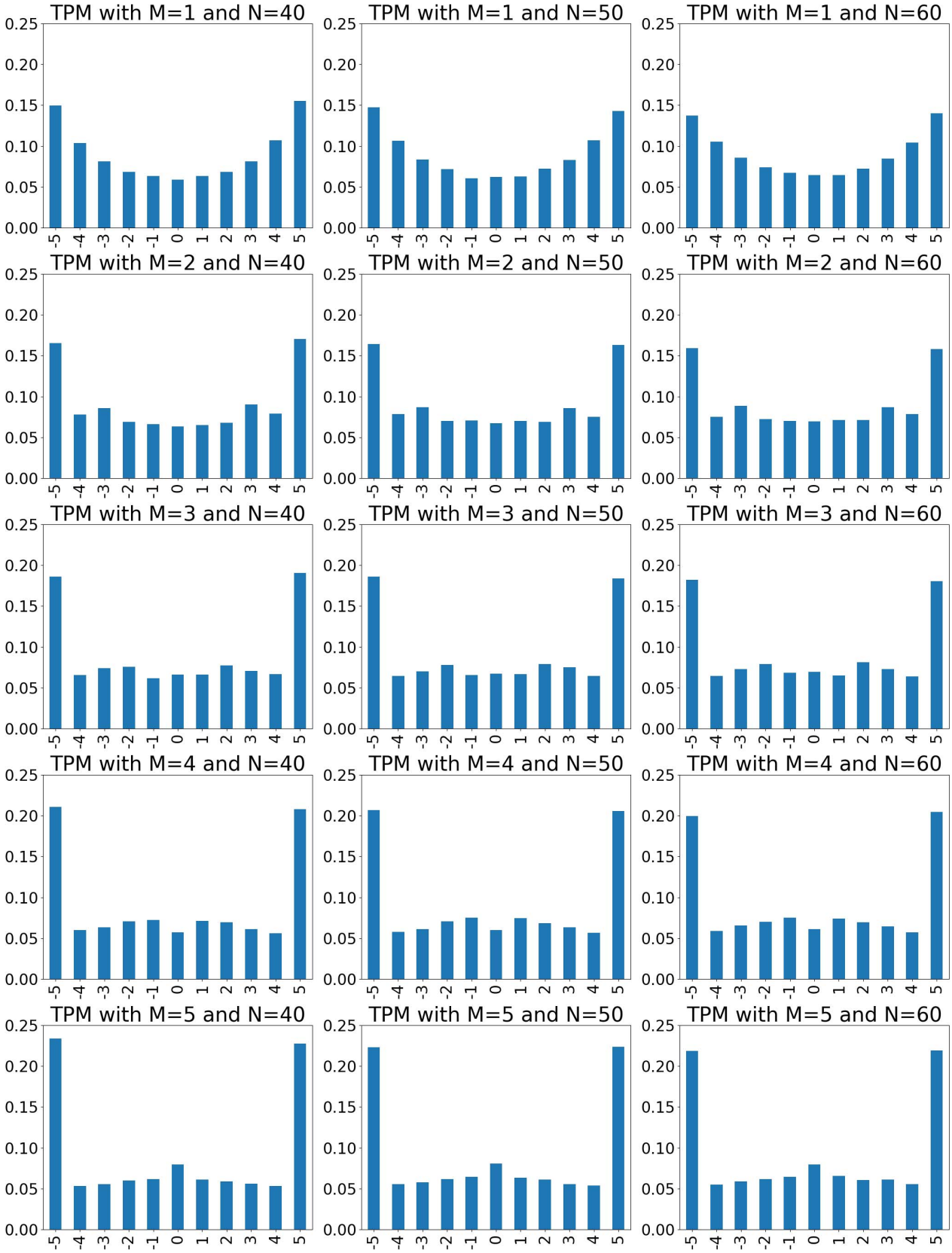


Fig. 3. Probability distribution of weights in TPMs using nonbinary input vectors.

in the following equation [23]:

$$H(p) = -p\log_j p - (1-p)\log_j(1-p). \tag{8}$$

Fig. 2 shows the plot of the entropy of the aforementioned two-value random source. The maximum of the function is reached for $p = 0.5$, where $H(p) = 1$, which is the equal probability for values 0 and 1.

Hence, entropy values increase as the probability distribution of X gets closer to the uniform distribution. This can be generalized to sources producing more outcomes.

The entropy function can be used later to assess the quality of the keys generated by different types of TPM. Taking into account (9), the effective length of a key should depend on the entropy of the synchronized weights (not just their values).

C. Agreed Key Length

After the synchronization process, both parties share identical keys. The keys are distilled from weights of the TPM, which are the same after the mutual learning process. The key length depends on the size of the TPM as well as the parameter L, which indicates the minimum/maximum value the weights may reach during synchronization. Assuming the ideal uniform distribution of the weights, the key length is equal to $K \cdot N \cdot log_2(2L+1)$. However, the distribution of the weights differs from the uniform distribution [17]. Hence, the entropy should be used to measure the quality of the key exchanged between the parties. The updated key length is defined as follows:

$$\text{length}_{\text{key}} = K \cdot N \cdot E(W) \tag{9}$$

where $E(W)$ indicates an average entropy of the weights. Entropy itself is presented in Section IV-B.

However, the exact distribution of weights is not known beforehand. Taking this fact into consideration, (9) should be updated. The estimated effective key length shown in (10) uses the estimated entropy based on the simulation results. Additionally, we propose using the floor function in the equation since the unit of effective key are bits

$$\hat{\text{length}}_{\text{key}} = \lfloor K \cdot N \cdot \hat{E}(W) \rfloor. \tag{10}$$

It should be noted that (9) indicates the theoretical maximum key length that can be extracted from mutual weights. However, a dedicated algorithm that equalizes the probability can be used to obtain a cryptography key from an unevenly distributed numerical sequence. This algorithm must be deterministic since both parties retrieve the cryptographic key from weights simultaneously.

V. Verification

This section presents the impact of the new parameter M, indicating the maximum/minimum value of the input vectors during the synchronization process and how it affects the required iterations in the learning process and the quality of the output key.

A. Methodology

To validate the proposed improvement, a dedicated TPM simulation framework has been created. The proposed framework allows mimicking a real network scenario where two parties willing to synchronize their TPMs exchange all the intermediate information via a public channel (parties perform actions from Section IV-A). Hence, the agreed TPM parameters and all the input vectors are available to eavesdroppers. The simulation framework is available in the public domain.[1]

The quality of the output key is measured in its effective length. The effective length is calculated on the basis of (10). Furthermore, simulation scenarios cover multiple sets of TPMs sizes. For each scenario, statistical analysis was prepared based on 1000 simulations. The presented confidence intervals are calculated with a 95% probability. These scenarios include all possible combinations of parameters $N \in \{40, 50, 60\}$ and $M \in \{1, 2, 3, 4, 5\}$. For all simulation scenarios, parameters K and L are equal to 3 and 5, respectively. Additionally, all the simulations are performed using the Hebbian learning rule (4). Synchronization time, entropy, and effective key length are measured in order to compare the chosen scenarios. Furthermore, we performed man-in-the-middle attack scenarios during which we measured the average synchronization score of the malicious TPM.

[1]The framework source code is available at *https://github.com/mstypinski/tpm*

B. Results

The synchronization process becomes longer as the size of the TPM increases; it also generates a longer key for cryptographic purposes. However, simulations presented in Table I reveal that the TPM size and parameters are not the only elements that have an impact on the duration of the synchronization process. Multiple simulations were performed with different values of parameter M. An increase in the parameter M value, which limits the maximum and minimum possible values x_i of the input vector X, reduces the synchronization time significantly. The synchronization time in Table I is expressed as a number of output bits exchanged between the parties to achieve full synchronization between the two TPMs (learning iterations). Thus, the volume of data exchanged between the parties performing key agreement decreases as the value of parameter M increases.

It should be noted that faster synchronization increases security. This is because as the value of parameter M increases, the key agreement process takes less time, hence a longer and more secure key is obtained in a shorter period of time. This makes this solution more competitive among other key exchange protocols.

Salguero Dorokhin *et al.* [24] conducted research regarding the optimal TPM structure for establishing a 512-bit cryptographic key. Based on their research, TPM parameters (K, L, N) providing the most security are $(8, 16, 8)$. On average, the full synchronization of such a TPM took 218.37 iterations. Comparing the above to the conducted research in this brief, TPMs having parameters (M, N) set to one of the following pairs (3, 50), (4, 50), (4, 60) perform better in terms of synchronization time while allowing to obtain cryptographic keys longer than 512 bits.

It is worth mentioning the synchronization time grows insignificantly with the increase of N. This allows extending distilled cryptographic keys without significantly extending synchronization time. The parameter having the highest impact on synchronization time is L as synchronization time grows proportionally to L^2. However, the same parameter is responsible for improving the security features of TPM [25]. The introduction of parameter M enables preserving comparable security features but results in a decrease in synchronization time.

C. Extrema Values Effect

Numerous simulations of the TPM learning process using nonbinary input vectors led to the discovery of an effect named by the authors the extrema value effect. A similar effect is shown in [10], however, only binary input vectors are considered in this brief.

Faster synchronization times and lower numbers of messages exchanged between users have an impact on the distribution of weights. As the minimum/maximum x_i increases, the probability $P(w_{kn} = M)$ and $P(w_{kn} = -M)$ also increases. As a result, the probability distribution of weights becomes less similar to the uniform distribution. Hence, every weight of the TPM carries less random information. The exact distribution of weights is presented in Fig. 3.

TABLE II
ENTROPY AND EFFECTIVE KEY LENGTH FOR DIFFERENT TPM PARAMETERS

M	N	Entropy	Estimated effective key length
1	40	3.374	404
2		3.354	402
3		3.305	396
4		3.238	388
5		3.158	378
1	50	3.386	507
2		3.368	505
3		3.315	497
4		3.248	487
5		3.186	477
1	60	3.402	612
2		3.379	608
3		3.325	598
4		3.263	587
5		3.204	576

TABLE III
SYNCHRONIZATION TIME OF TPMS WITH DIFFERENTIATED INPUT VECTORS

M	N	Adversarial TPMs synchronization score (S_{score})			
		Average	Minimum	Maximum	Median
1	40	0.174 ± 0.047	0.058	0.425	0.167
2	40	0.18 ± 0.055	0.058	0.592	0.167
3	40	0.205 ± 0.079	0.075	0.933	0.183
4	40	0.243 ± 0.101	0.083	0.983	0.208
5	40	0.28 ± 0.136	0.083	1	0.233
1	50	0.17 ± 0.039	0.073	0.373	0.16
2	50	0.179 ± 0.047	0.08	0.513	0.167
3	50	0.204 ± 0.06	0.087	0.527	0.187
4	50	0.232 ± 0.085	0.087	0.993	0.207
5	50	0.255 ± 0.107	0.067	1	0.213
1	60	0.168 ± 0.037	0.072	0.361	0.167
2	60	0.183 ± 0.048	0.061	0.417	0.172
3	60	0.204 ± 0.066	0.067	0.65	0.183
4	60	0.231 ± 0.084	0.078	0.956	0.206
5	60	0.252 ± 0.117	0.089	1	0.211

Assuming that in tth iteration, the update of all weights for all the neurons is successful and the distribution of all the weights $w_{kn}(t)$ and elements of input vector $x_{kn}(t)$ is uniform, let us estimate the probability of the weight $w_{kn}(t+1)$ being equal to L or $-L$ in the next iteration. We can only consider one of these values since the situation is analogous to the other. The probability $P(w_{kn}(t+1) = L)$ is equal to $P(w_{kn}(t+1) = L|x_{kn} = 0) + P(w_{kn}(t+1) = L|x_{kn} = 1) + \cdots + P(w_{kn}(t+1) = L|x_{kn} = M-1) + P(w_{kn}(t+1) = L|x_{kn} = M)$. Let us consider $P(w_{kn}(t+1) = L|x_{kn} = 0)$. This event only occurs providing $w_{kn} = L$, and therefore the probability is equal to $(1/(2M+1)(2L+1))$, as there is only one satisfactory event for all combinations of input and weight values. For $\mathrm{P}(w_{kn}(t+1) = L|x_{kn} = 1)$, there are two satisfactory events $w_{kn} = L$ and $w_{kn} = L-1$. In conclusion, the probability of the event A where $w_{kn}(t+1) = L$ or $w_{kn}(t+1) = -L$ is presented in the following equation:

$$\begin{aligned} P(A) &= 2 \times (P(A|x_{kn} = 0) + \cdots + P(A|x_{kn} = M)) \\ &= 2 \times \left(\frac{1 + \cdots + (M+1)}{(2M+1)(2L+1)} \right) \\ &= \frac{(M+1)(M+2)}{(2M+1)(2L+1)}. \end{aligned} \tag{11}$$

As the parameter M increases the probability of the weights being equal to either L or $-L$ raises in the next iteration, resulting in diminished TPM robustness. This fact makes parameter selection a key element of TPM security.

The unequal distribution of weights in the TPM results in a reduction of the effective key length since as the entropy value becomes lower, the less secure bits might be retrieved from the key. Entropy values and effective key lengths are presented in Table II. To visualize the proportion between effective key length, the results for the considered M and N parameters are presented in Fig. 4.

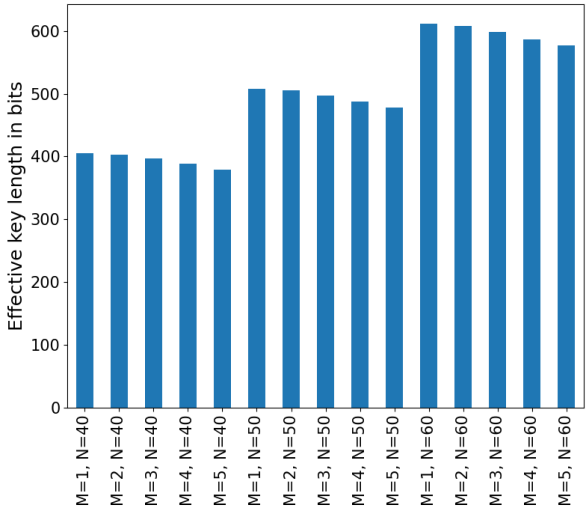


Fig. 4. Effective key length of TPMs with different parameters.

D. Susceptibility to a Man-in-the-Middle Attack

Many research considerations address TPM vulnerability to man-in-the-middle attacks. Therefore, simulations with adversarial TPMs have been conducted while utilizing learning by nonbinary input vectors.

We assumed the worst-case scenario in which the adversarial neural network was able to eavesdrop on all of the data exchanged between the parties performing the key exchange. During the simulations, the final synchronization score S_{score} was gathered for the adversarial neural network. The synchronization score measures the similarity between two TPMs. The more common weights there are, the higher the score value is assigned. Hence, the formula needs to return higher values with the progress of the learning process. For equal TPMs, the synchronization score is equal to 1. The formula for calculating the end score is presented in (12). In the following equation, w^A denotes weights of adversarial TPMs and function $\Theta(a, b)$ is defined in Section III:

$$S_{\text{score}} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta\left(w_{kn}, w_{kn}^{A}\right)}{K \times N}. \tag{12}$$

In terms of security, the attacker's TPM should have the lowest synchronization score possible.

The synchronization score of adversarial TPMs are presented in Table III. Additionally, simulation results are shown in Fig. 5 to visualize the relationship between scenarios with different TPMs. Increased values of parameter M result in higher median synchronization scores, hence the TPM is more prone to man-in-the-middle attacks. When parameter M was equal to L, we observed situations where the synchronization score was equal to 1. This means that the relationship between parameters $M < L$ should be preserved to ensure security. Additionally, the median is

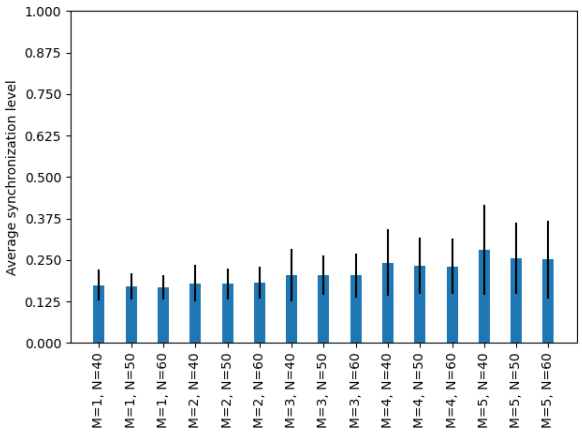


Fig. 5. Synchronization score of adversarial TPMs.

inversely proportional to the number of inputs N, and therefore the impact of nonbinary input vectors on the synchronization score is less clear for larger TPMs. Furthermore, the confidence intervals are considerable. This variability makes it difficult to predict the attacker's malicious TPM weights.

VI. Summary

Correct selection of TPM parameters is a key issue in implementing secure key agreement protocols for neural cryptography. It is crucial to find a tradeoff between effective key length, synchronization time, and security of the final key which is used by users to protect data in the network environment. This comes down to selecting the appropriate network size, extreme values of the weights, and learning vectors.

This brief proposes an improved way of learning TPMs. A significant acceleration of the key agreement process was achieved by utilizing a nonbinary input vector. This reduces the volume of data exchanged between the parties performing key agreements. Faster synchronization increases security levels; in particular, it mitigates the risk of the key being obtained by an intruder using a man-in-the-middle attack. However, speeding up the process results in an unequal distribution of weights in the TPM. This was measured by calculating the effective key length based on the entropy of each weight. The proposed solution was also verified in an insecure environment in which two TPMs are subject to a man-in-the-middle attack.

We envisage that future work will explore the development of a secure key exchange protocol using nonbinary input vectors in TPMs during mutual learning. This work will be focused on studying the extrema values effect thoroughly and minimizing the reduction of effective key length.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, Jan. 1999.

[2] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Europhys. Lett.*, vol. 57, no. 1, pp. 141–147, Jan. 2002.

[3] M. Rosen-Zvi, I. Kanter, and W. Kinzel, "Cryptography based on neural networks analytical results," *J. Phys. A, Math. Gen.*, vol. 35, no. 47, pp. L707–L713, Nov. 2002, doi: 10.1088/0305-4470/35/47/104.

[4] S. Santhanalakshmi, T. S. B. Sudarshan, and G. K. Patra, "Neural synchronization by mutual learning using genetic approach for secure key generation," in *Recent Trends in Computer Networks and Distributed Systems Security*, S. M. Thampi, A. Y. Zomaya, T. Strufe, J. M. A. Calero, and T. Thomas, Eds. Berlin, Germany: Springer, 2012, pp. 422–431.

[5] A. M. Allam, H. M. Abbas, and M. W. El-Kharashi, "Authenticated key exchange protocol using neural cryptography with secret boundaries," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[6] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," *Quantum Inf. Process.*, vol. 18, no. 6, p. 174, Jun. 2019, doi: 10.1007/s11128-019-2296-4.

[7] A. Sarkar, M. Z. Khan, M. M. Singh, A. Noorwali, C. Chakraborty, and S. K. Pani, "Artificial neural synchronization using nature inspired whale optimization," *IEEE Access*, vol. 9, pp. 16435–16447, 2021.

[8] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4999–5004, Nov. 2020.

[9] S. Jeong, C. Park, D. Hong, C. Seo, and N. Jho, "Neural cryptography based on generalized tree parity machine for real-life systems," *Secur. Commun. Netw.*, vol. 2021, Feb. 2021, Art. no. 6680782.

[10] A. Sarkar, "Secure exchange of information using artificial intelligence and chaotic system guided neural synchronization," *Multimedia Tools Appl.*, vol. 80, no. 12, pp. 18211–18241, Feb. 2021, doi: 10.1007/s11042-021-10554-3.

[11] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[12] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[13] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 421–427, Apr. 2005.

[14] M. Just, *Key Agreement*. Boston, MA, USA: Springer, 2005, p. 325, doi: 10.1007/0-387-23483-7_218.

[15] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[16] W. Kinzel, *Theory of Interacting Neural Networks*. Hoboken, NJ, USA: Wiley, 2002, ch. 9, pp. 199–217, doi: 10.1002/3527602755.

[17] A. Ruttor, "Neural synchronization and cryptography," Ph.D. dissertation, Dept. Phys. Astron., Univ. Würzburg, Würzburg, Germany, 2007.

[18] M. Javurek and M. Turcanik, "Synchronization of two tree parity machines," in *Proc. New Trends Signal Process. (NTSP)*, Oct. 2016, pp. 1–4.

[19] M. Dolecki and R. Kozera, "Distance of the initial weights of tree parity machine drawn from different distributions," *Adv. Sci. Technol. Res. J.*, vol. 9, no 26, pp. 137–142, 2015, doi: 10.12913/22998624/2380.

[20] J. Martínez Padilla, U. Meyer-Baese, and S. Foo, "Security evaluation of tree parity re-keying machine implementations utilizing side-channel emissions," *EURASIP J. Inf. Secur.*, vol. 2018, no. 1, pp. 1–16, Apr. 2018, doi: 10.1186/s13635-018-0073-z.

[21] M. E. O'Neill, "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation," Harvey Mudd College, Claremont, CA, USA, Tech. Rep., HMC-CS-2014-0905, Sep. 2014.

[22] E. Simion, "Entropy and randomness: From analogic to quantum world," *IEEE Access*, vol. 8, pp. 74553–74561, 2020.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory* (Wiley Series in Telecommunications and Signal Processing). Hoboken, NJ, USA: Wiley, 2006.

[24] E. S. Dorokhin, W. Fuertes, and E. Lascano, "On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key," *Secur. Commun. Netw.*, vol. 2019, Mar. 2019, Art. no. 8214681, doi: 10.1155/2019/8214681.

[25] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel, "Secure key-exchange protocol with an absence of injective functions," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 66, no. 6, Dec. 2002, Art. no. 066102.

KOMIT 2022

MULTIKONFERENCJA
ŚRODOWISKA
TELE- I RADIOKOMUNIKACYJNEGO

KKRRi 2022

ANALIZA ZALEŻNOŚĆI POMIĘDZY PARAMETRAMI SIECI TPM A CZASEM SYNCHRONIZACJI

ANALYSIS OF THE RELATIONSHIP BETWEEN TPM NETWORK PARAMETERS AND ITS SYNCHRONIZATION TIME

Miłosz Stypiński[1]

[1] Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Streszczenie**: Kryptografia z wykorzystaniem sieci neuronowych to nowy trend w cyberbezpieczeństwie. Sieci typu Tree Parity Machine oferują alternatywę dla powszechnie stosowanych protokołów uzgodnienia klucza kryptograficznego. Celem tego artykułu jest analiza zależności pomiędzy parametrami sieci TPM, a czasem synchronizacji oraz jej bezpieczeństwem.**
Słowa kluczowe**: bezpieczeństwo, kryptografia, sieci neuronowe, uzgodnienie klucza kryptograficznego**

Abstract**: Neural cryptography is a new trend in cybersecurity. Tree Parity Machine (TPM) networks offer an alternative to commonly used cryptographic key agreement protocols. The purpose of this article is to analyze the relationship between TPM network parameters and the synchronization time and its security.**
Keywords**: cryptography, neural networks, secure key agreement, security**

1. WSTĘP

Zmierzając ku coraz powszechniejszej cyfryzacji, bezpieczeństwo staje się coraz bardziej kluczową składową systemów i aplikacji. Niezależnie od dziedziny zainteresowania, cyberbezpieczeństwo nie może zostać odsunięte na dalszy plan. W innym wypadku dochodzi do bardzo medialnego wykorzystania podatności, tak jak to miało miejsce w 2018 roku w przypadku firmy Facebook, gdzie zostałby wykradzione dane 30 milionów użytkowników [1].

Algorytmy bezpieczeństwa stanowią nierozłączny element dzisiejszych rozwiązań informatycznych, gdzie wśród nich znajdują się algorytmy szyfrowania asymetrycznego. Najpopularniejsze algorytmy szyfrowania asymetrycznego stosowanie obecnie to algorytmy RSA i pochodne; oraz rodzina algorytmów opartych na krzywych eliptycznych. Niestety wymienione grupy algorytmów są bezpieczne warunkowo, to znaczy, że opierają się na problemie obliczeniowym, który na dzień dzisiejszy nie posiada efektywnego algorytmu będącego w stanie odwrócić odpowiednich operacji matematycznych w racjonalnym czasie. Problemy na których oparte są wspomniane algorytmy to odpowiednio faktoryzacja liczb1 pierwszych oraz „dodawanie" punktów na krzywej eliptycznej w pierścieniu modulo. Niestety, oba te problemy mogą być efektywnie rozwiązywanie w przypadku implementacji algorytmu Shor'a na komputerze kwantowym [6]. Aby zapobiec załamaniu bezpieczeństwa wszystkich istniejących systemów informatycznych, powstają algorytmy odporne na działanie komputera kwantowego. Jednym z nich jest uzgodnienie klucza z wykorzystaniem sieci typu Tree Parity Machine (TPM).

2. Sieć TPM

W celu przedstawienia algorytmu synchronizacji dwóch sieci TPM, co pozwala na uzgodnienie klucza kryptograficznego, najpierw zostanie przedstawiona jej struktura, następnie protokół uzgodnienia klucza, a na koniec rozdziału przedstawione zostaną reguły uczące iteratywnie aktualizujące wagi sieci TPM, tworzące klucz kryptograficzny.

2.1. Struktura sieci TPM

Sieć TPM to struktura przypominająca dwuwarstwowy perceptron. W celu dokładnego opisania sieci TPM wykorzystuje się trzy parametry: *K, L, N*, które są kolejno odpowiedzialne za liczbę wejść do każdego z neuronów pierwszej warstwy; za granicę przedziału ograniczającego wartości wag oraz za liczbę neuronów w pierwszej warstwie sieci [2]. Sieć jest uczona binarnym wektorem wejściowym, którego składowe przyjmują wartości -1 lub 1. Na wyjściu sieci znajduje się tylko jeden neuron.

Znaczącą różnicą pomiędzy modelem perceptronu, a strukturą TPM jest fakt, że wagi w sieci TPM mogą przyjmować jedynie wartości całkowitoliczbowe. Ponadto, funkcje aktywacji zostały zmodyfikowane odpowiednio, tak aby zwracały wartości binarne -1 lub 1. Funkcja aktywacji w neuronach Θ pierwszej warstwy jest zmodyfikowaną funkcją signum i ma następującą postać:

$$\sigma = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (1)$$

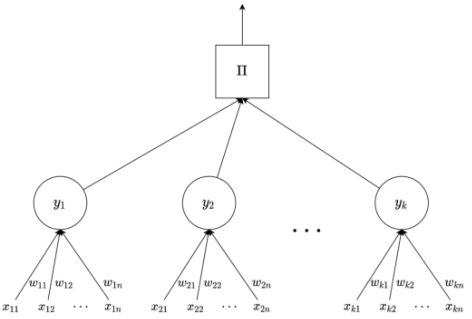
Z wykorzystaniem funkcji (1) możemy przedstawić wzór na wartość wyjściową neuronu pierwszej warstwy (2).

$$y_k = \sigma\left(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}\right) \quad (2)$$

Neuron drugiej warstwy nie posiada powiązanych z nim wag, a funkcja aktywacji jest tożsama z iloczynem wartości w wektorze wejściowym. Z uwagi na fakt, że wektor wejściowy składa się z liczb -1 lub 1, w rezultacie wyjście całej sieci też jest równe -1 lub 1, co jest kluczowe w uzgodnieniu klucza kryptograficznego.

Dodatkowo w [7] autorzy zaproponowali nowy parametr M, odpowiedzialny za dywersyfikację wartości, które może przyjmować wektor wejściowy. Parametr M tworzy przedział wartości $< -M, M >$, który ogranicza wartości poszczególnych elementów wektora wejściowego. Jednak, te wartości muszą pozostać całkowitoliczbowe, aby umożliwić uzgodnienie klucza kryptograficznego. Wprowadzenie parametru M nie ma wpływu na protokół uzgodnienia klucza, a skutkuje przyspieszeniem synchronizacji, a co za tym idzie zwiększa bezpieczeństwo całego systemu. Dalej w badaniach rozważane są tylko sieci TPM o parametrach K, L, M i N. Analiza parametrów sieci TPM bez parametru M jest dostępna między innymi w [4].

Schemat sieci TPM został przedstawiony na Rysunku 1.



Rys. 1 Struktura sieci TPM

2.2. Uzgodnienie klucza kryptograficznego w sieci TPM

Celem synchronizacji dwóch sieci TPM jest uzgodnienie klucza kryptograficznego w bezpieczny sposób, co jest tożsame z tym, że żaden postronny podmiot nie będzie w stanie odtworzyć uzgodnionego sekretu na podstawie wymienionych informacji.

Protokół wykorzystujący sieci TPM do uzgodnienia sekretu został przedstawiony poniżej:

1. strony uzgadniające klucz jawnie określają parametry K, L, M, N sieci TPM;
2. jedna ze stron wybiera losowy wektor X składający się z wartości $x_i \in < -N, N >$;
3. każda ze stron oblicza odpowiedź sieci dla wcześniej ustalonego wektora X i ujawnia odpowiedź publicznie;
4. w przypadku, gdy odpowiedzi obu sieci się zgadzają, każda ze stron aktualizuje wagi swojej sieci stosując odpowiednią regułę uczącą;
5. punkty od 2 do 4 powtarzane są do momentu pełnej synchronizacji obu sieci.

Aby powyższy protokół mógł działać wymagane są reguły uczące, których zastosowanie skutkuje pełną synchronizacją sieci w skończonym czasie. W [5] opisane zostały trzy różne reguły uczące, które zostały zaprezentowane poniżej:

- reguła Hebbian

$$w_{kn}(t+1) = w_{kn}(t) + O(t)x_{kn}(t)\Theta\big(y_k(t), O(t)\big) \quad (3)$$

- reguła anti-Hebbian

$$w_{kn}(t+1) = w_{kn}(t) - O(t)x_{kn}(t)\Theta\big(y_k(t), O(t)\big) \quad (4)$$

- reguła losowego spaceru

$$w_{kn}(t+1) = w_{kn}(t) + x_{kn}(t)\Theta\big(y_k(t), O(t)\big) \quad (5)$$

W powyższych wzorach t oznacza iterację protokołu uzgodnienia klucza, k oznacza k-ty neuron sieci TPM, n oznacz n-te wejście neuronu, a y_n jest wyjściem n-tego neuronu pierwszej warstwy sieci.

W [3] autorzy udowodnili, że stosowanie powyższego protokołu skutkuje synchronizacją sieci TPM w skończonym czasie oraz, co jest równie ważne, w czasie krótszym niż osoby postronne próbujące wykorzystać wymienione dane do odtworzenia uzgodnionego klucza kryptograficznego.

Długość uzgodnionego klucza kryptograficznego zależy od parametrów sieci TPM i wyraża się następującym wzorem:

$$Długość\ klucza\ = K \times N \times log_2(L) \quad (6)$$

Trzeba jednak pamiętać, że wybór parametrów K, L, M, N również wpływa na rozkład wartości wag podczas synchronizacji, więc efektywna długość klucza kryptograficznego będzie odgórnie ograniczona przez (6) [7].

3. WPŁYW PARAMETRÓW NA CZAS SYNCHRONIZACJI

W tej sekcji przedstawione są metodologia badań, opis środowiska symulacyjnego oraz otrzymane wyniki.

3.1. Metodologia badań

W celu sprawdzenia wpływu zmian wartości parametru M na bezpieczeństwo uzgodnionego klucza wykorzystane zostało dedykowane środowisko symulacyjne. W każdym uruchomieniu synchronizowane były 3 różne sieci TPM. Dwie sieci należące do stron chcących uzgodnić kluczy kryptograficzny oraz dodatkowa jedna sieć TPM próbująca odtworzyć klucz kryptograficzny na podstawie wymienianych wektorów wejściowych. Ostatnia z wymienionych sieci w dalszej części artykułu nazywana jest złośliwą siecią TPM.

Przeprowadzone symulacje wykorzystują sieci TPM z parametrami K i N równymi odpowiednio 3 i 50. Parametry M i L były zależne od symulowanego scenariusza i przybierały odpowiednio wartości z następujących zbiorów: {1,2,3,4,5,6,7} oraz {8,9,10,11}. Dla każdego zestawu parametrów przeprowadzono 100 symulacji, podczas których badana była potrzebna liczba iteracji do pełnej synchronizacji oraz podobieństwo złośliwej sieci TPM według następującego wzoru:

$$S_{score} = \frac{\sum_{k=1}^{K}\sum_{n=1}^{N}\Theta\big(w_{kn}, w_{kn}^{A}\big)}{K \times N}, \quad (6)$$

gdzie Θ to funkcja zwracająca 1 w przypadku, gdy jej argumenty są równe, a w przeciwnym przypadku zwracająca 0.

Tab.1 Średnia liczba iteracji oraz średni poziom synchronizacji złośliwej sieci TPM

		Synchronizowane sieci TPM				Złośliwa sieć TPM			
L	M	Średnia	Mediana	Min	Max	Średnia	Mediana	Min	Max
8	1	$\mathbf{1699,6 \pm 784,4}$	**1631**	**1010**	**2614**	$\mathbf{0,133 \pm 0,076}$	**0,127**	**0,067**	**0,207**
	2	$\mathbf{690,5 \pm 310,8}$	**695**	**388**	**1092**	$\mathbf{0,128 \pm 0,087}$	**0,127**	**0,067**	**0,260**
	3	$\mathbf{415,4 \pm 236,8}$	**406**	**619**	**619**	$\mathbf{0,147 \pm 0,089}$	**0,140**	**0,080**	**0,234**
	4	$\mathbf{240,1 \pm 144,3}$	**228**	**141**	**429**	$\mathbf{0,150 \pm 0,0882}$	**0,140**	**0,073**	**0,226**
	5	$\mathbf{205,1 \pm 191,9}$	**178**	**84**	**580**	$\mathbf{0,184 \pm 0,177}$	**0,160**	**0,080**	**0,494**
	6	$\mathbf{131,2 \pm 97,6}$	**119**	**75**	**251**	$\mathbf{0,168 \pm 0,140}$	**0,153**	**0,067**	**0,326**
	7	$\mathbf{118,6 \pm 83,6}$	**110**	**71**	**229**	$\mathbf{0,210 \pm 0,216}$	**0,180**	**0,053**	**0,467**
9	1	$\mathbf{2337,0 \pm 1298,7}$	**2095**	**1456**	**4290**	$\mathbf{0,119 \pm 0,082}$	**0,107**	**0,053**	**0,200**
	2	$\mathbf{910,0 \pm 506,4}$	**869**	**491**	**1462**	$\mathbf{0,129 \pm 0,097}$	**0,120**	**0,067**	**0,253**
	3	$\mathbf{495,6 \pm 275,6}$	**518**	**272**	**518**	$\mathbf{0,130 \pm 0,122}$	**0,120**	**0,393**	**0,393**
	4	$\mathbf{328,6 \pm 166,0}$	**310**	**164**	**470**	$\mathbf{0,150 \pm 0,108}$	**0,133**	**0,080**	**0,287**
	5	$\mathbf{266,2 \pm 225,4}$	**258**	**110**	**596**	$\mathbf{0,141 \pm 0,101}$	**0,140**	**0,073**	**0,340**
	6	$\mathbf{158,7 \pm 106,4}$	**145**	**69**	**299**	$\mathbf{0,152 \pm 0,108}$	**0,147**	**0,087**	**0,360**
	7	$\mathbf{124,6 \pm 72,2}$	**114**	**68**	**215**	$\mathbf{0,209 \pm 0,334}$	**0,187**	**0,067**	**0,980**
10	1	$\mathbf{2552,4 \pm 1288,6}$	**2399**	**1812**	**4681**	$\mathbf{0,116 \pm 0,045}$	**0,113**	**0,073**	**0,160**
	2	$\mathbf{1246,2 \pm 866,8}$	**1114**	**700**	**2175**	$\mathbf{0,114 \pm 0,067}$	**0,113**	**0,053**	**0,173**
	3	$\mathbf{611,1 \pm 280,6}$	**584**	**314**	**912**	$\mathbf{0,113 \pm 0,065}$	**0,113**	**0,033**	**0,200**
	4	$\mathbf{363,3 \pm 221,1}$	**350**	**201**	**694**	$\mathbf{0,141 \pm 0,115}$	**0,133**	**0,040**	**0,314**
	5	$\mathbf{275,28 \pm 174,6}$	**259**	**158**	**470**	$\mathbf{0,134 \pm 0,102}$	**0,120**	**0,080**	**0,340**
	6	$\mathbf{264,6 \pm 164,0}$	**253**	**110**	**550**	$\mathbf{0,150 \pm 0,131}$	**0,133**	**0,087**	**0,326**
	7	$\mathbf{168,3 \pm 95,9}$	**156**	**101**	**262**	$\mathbf{0,152 \pm 0,108}$	**0,147**	**0,053**	**0,280**
11	1	$\mathbf{3019,8 \pm 1453,6}$	**3002**	**1946**	**4665**	$\mathbf{0,121 \pm 0,054}$	**0,120**	**0,067**	**0,180**
	2	$\mathbf{1333,3 \pm 745,9}$	**1290**	**571**	**2493**	$\mathbf{0,112 \pm 0,073}$	**0,100**	**0,060**	**0,233**
	3	$\mathbf{774,0 \pm 444,7}$	**706**	**377**	**1248**	$\mathbf{0,103 \pm 0,062}$	**0,100**	**0,027**	**0,166**
	4	$\mathbf{528,2 \pm 219,6}$	**528**	**311**	**789**	$\mathbf{0,116 \pm 0,090}$	**0,113**	**0,060**	**0,240**
	5	$\mathbf{350,56 \pm 199,0}$	**351**	**159**	**547**	$\mathbf{0,127 \pm 0,117}$	**0,113**	**0,053**	**0,374**
	6	$\mathbf{264,6 \pm 164,0}$	**253**	**115**	**427**	$\mathbf{0,150 \pm 0,131}$	**0,133**	**0,087**	**0,326**
	7	$\mathbf{218,4 \pm 162,9}$	**188**	**107**	**427**	$\mathbf{0,154 \pm 0,140}$	**0,127**	**0,073**	**0,427**

3.1. Analiza wyników

W wyniku symulacji można zauważyć, że wraz ze wzrostem parametru *L* mediana potrzebnych iteracji do pełnej synchronizacji rośnie. Należy jednak pamiętać, że zwiększenie parametru *L* skutkuje również wydłużeniem otrzymanego klucza kryptograficznego zgodnie ze wzorem (6). Ponadto, można zauważyć, że zwiększenie wartości parametru M znacząco redukuje medianę potrzebnych iteracji protokołu. Jednak, negatywnym skutkiem

wzrostu wartości parametru M jest również szybsza synchronizacja złośliwej sieci TPM.

Można też zauważyć, że dla wartości parametru M równego 7, maksymalna synchronizacja złośliwej sieci TPM zbliżała się do wartości 0,5. W przypadku, gdy strona atakująca zwielokrotniłaby atak poprzez synchronizowanie wielu sieci na raz, istnieje prawdopodobieństwo, że klucz został by uzyskany przez stronę atakującą, co skutkowałoby kompromitacją całego systemu bezpieczeństwa.

Wyniki wszystkich symulacji zostały przedstawione w Tabeli 1.

4. PODSUMOWANIE

W artykule przedstawione zostało działanie protokołu uzgodnienia klucza kryptograficznego wykorzystującego sieci TPM. Rozważone zostały tylko sieci TPM z dodatkowym parametrem M. Następnie dla sieci TPM o różnych zestawach parametrów przeprowadzona została analiza zmian wartości parametrów na średnią długość uzgodnienia klucza wyrażoną w iteracja oraz średnią synchronizację złośliwej sieci TPM dla każdego ze scenariuszy. Całość została zakończona analizą otrzymanych wyników.

Sieci TPM mogą stać się konkurencyjnym rozwiązaniem w momencie udanej implementacji algorytmu Shor'a na komputerze kwantowym, ze względu na odporność na działanie tego algorytmu. Przedstawiona analiza wykazała, że kluczowym aspektem tworzenia systemu kryptograficznego wykorzystującego sieci TPM jest dobór jej parametrów, tak aby zapewnić wymaganą długość klucza projektowanego systemu kryptograficznego przy jednoczesnej maksymalizacji bezpieczeństwa wymiany klucza kryptograficznego z wykorzystaniem sieci TPM.

LITERATURA

[1] Blog aplikacji Facebook, https://about.fb.com/news/2018/10/update-on-security-issue/, data dostępu: 2022-05-09

[2] Kanter I., Kinzel W., i Kanter E., "Secure exchange of information by synchronization of neural networks," EPL (Europhysics Letters), vol. 57, 02 2002.

[3] Kinzel W., "Theory of interacting neural networks", ch. 9, pp. 199–217. John Wiley & Sons, Ltd, 2002

[4] Rosen-Zvi M., Kanter I., i Kinzel W., "Cryptography based on neural networks analytical results," Journal of Physics A: Mathematical and General, vol. 35, pp. L707–L713, 10 2002.

[5] Ruttor A., Neural Synchronization and Cryptography. Rozprawa doktorska, Uniwersytet w Wurzburgu, 2007.

[6] Shor P. W., "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124-134, doi: 10.1109/SFCS.1994.365700.

[7] M. Stypiński and M. Niemiec, "Synchronization of Tree Parity Machines Using Nonbinary Input Vectors," in IEEE Transactions on Neural Networks and Learning Systems, 2022, doi: 10.1109/TNNLS.2022.3180197.

energies

MDPI

Article

Security of Neural Network-Based Key Agreement Protocol for Smart Grids

Miłosz Stypiński * and Marcin Niemiec *

Institute of Telecommunications, AGH University of Krakow, Mickiewicza 30, 30-059 Krakow, Poland
* Correspondence: stypinski@agh.edu.pl (M.S.); niemiec@agh.edu.pl (M.N.)

Abstract: Recent developments in quantum computing pose a significant threat to the asymmetric cryptography currently in use. Neural cryptography offers a potential alternative that is resistant to attacks of known quantum computer algorithms. The considered solution is lightweight and computationally efficient. If a quantum computer algorithm were successfully implemented, it could expose IoT sensors and smart grid components to a wide range of attack vectors. Given the lightweight nature of neural cryptography and the potential risks, neural cryptography could have potential applications in both IoT sensors and smart grid systems. This paper evaluates one of the suggested enhancements: the use of integer-valued input vectors that accelerate the synchronization of the Tree Parity Machine. This enhancement introduces a new parameter M that indicates the minimum and maximum values of input vector elements. This study evaluates the nonbinary version of the mutual learning algorithm in a simulated insecure environment. The results indicate that, while the Nonbinary Tree Parity Machine may involve some trade-offs between security and synchronization time, the speed improvement is more substantial than the decrease in security. The impact of this enhancement is particularly significant for smaller adjustments to parameter M.

Keywords: cybersecurity; key agreement; neural networks; mutual learning; smart grids

Citation: Stypiński, M.; Niemiec, M. Security of Neural Network-Based Key Agreement Protocol for Smart Grids. *Energies* **2023**, *16*, 3997. https://doi.org/10.3390/en16103997

Academic Editor: Abu-Siada Ahmed

Received: 19 April 2023
Revised: 3 May 2023
Accepted: 6 May 2023
Published: 9 May 2023

1. Introduction

Asymmetric cryptography and key agreement protocols are necessary for communicating parties over public networks to authenticate each other and perform secure data exchange. The well-known solutions capable of achieving this goal are the RSA algorithm and the Diffie–Hellman protocol. The security of both and their successors is guaranteed by the current inability to effectively solve certain number theory problems, such as prime factorization or discrete logarithm problems. The security of these two and similar algorithms was undisputed until the discovery of Shor's algorithm. This algorithm needs a quantum computer with a sufficient number of qubits. However, it is capable of successfully solving the fundamental problem of number theory: finding the prime factors of an integer. Therefore, modern communication requires new solutions to ensure data security [1].

Numerous research studies have focused on finding algorithms that are resistant to Shor's algorithm. One of the proposed solutions is mutual learning by two neural networks, called Tree Parity Machines (TPMs) [2,3]. TPMs performs key agreement by exchanging arbitrary input vectors and applying learning rules. Properly selected learning rules ensure that the mutual learning process finishes in a finite time [2]. Furthermore, mutual learning algorithms do not require computationally intensive numerical calculations, making them suitable for a wide range of applications, including smart grids. However, maintaining a proper level of security using this solution is still a challenge.

Smart grid systems are no exception here and would become prone to attacks providing successful implementation of Shor's algorithm. Additionally, power systems are being challenged to become more flexible due to factors such as energy distribution from renewable sources [4]. The vastness of interconnected sensors magnify the potential attack vectors

that could be exploited. Hence, power systems become more distributed, and this poses additional threats to the system as a whole. Authorization, authentication and accounting need to be provided by design to prevent data integrity attacks. Further, appropriate cybersecurity mechanisms prevent some physical layer attacks, particularly man-in-the-middle attacks, which require the attacker to access the communication medium. A key agreement protocol plays a crucial role in the cybersecurity system, ensuring that smart grids can resist a wide range of potential attacks. Another important branch in securing smart grid systems is immediate threat detection [5]. Neural cryptography could find its potential application in every component of smart grid systems, such as SCADA components, IoT sensors and all smart appliances [6].

A special case of TPM is Nonbinary Tree Parity Machine (NBTPM), which uses integer-valued input vectors instead of binary ones. This improvement results in reduced synchronization time of TPM and introduces the parameter M, which is responsible for the variability of input vectors and creates another dimension for finding the optimum TPM structure.

Given the difficulty of finding and verifying the security of TPMs, this article aims to evaluate the security of TPM and its nonbinary variant in various attack scenarios. Furthermore, it aims to offer guidance on determining the optimal structure of TPM for a given cryptosystem. The paper presents the results of a security analysis of NBTPM and is organized as follows. Section 2 reviews related literature. Section 3 introduces the TPM model, mutual learning and potential attacks. Section 4 details the research methodology, the results of simulations and the analysis of the findings. The last section concludes the paper.

2. Related Work

The Diffie–Hellman protocol was the first proposed algorithm for two parties to exchange a cryptographic key based on the difficulty of efficiently solving a mathematical problem, specifically the discrete logarithm problem [7]. Further, the algorithm was improved by utilizing elliptical curves [8]. However, both solutions are susceptible to being compromised by quantum computing.

As a response to the potential threat, a new family of post-quantum algorithms was proposed. This family consist of algorithms based on mathematical problems that are resistant to Shor's algorithm, such as supersingular elliptic curve isogeny, error correction codes and lattice-based and multivariate cryptosystems [9]. This paper focuses on another quantum-proof algorithm: TPM and its nonbinary variant. TPMs serve as an alternative to currently known secure key-exchange protocols. Currently, no known algorithms make TPM vulnerable to quantum computing. However, there is no mathematical proof of the security of TPM, and the computational power required to break the mutual learning is also unknown [10].

TPM has been the subject of much research [2,3,10–25]. Metzler et al. [2], Kinzel et al. [3] and Ruttor et al. [11] have shown that interacting neural networks can synchronize efficiently by using the mutual learning algorithm. In [10], the application of TPM in the field of cybersecurity was presented, wherein the synchronization of two neural networks allowed the performing of secure key agreement. The findings from these studies were consolidated in [12].

In [13–19], various improvements to TPMs were proposed. Researchers proposed an improved variant of the learning algorithm in scenarios where both parties are able to share a secret beforehand [13]. Furthermore, Santhanalakshmi et al. [14] and Sarkar et al. [15] proposed improvements based on a genetic algorithm application and whale optimization algorithms, respectively. In [16], the authors proposed a Gaussian distribution-based selection of initial weights that resulted in shorter synchronization time. Another set of improvements is based on altering the values an input vector can take. The first of them is Complex-Valued Tree Parity Machine (CVTPM), which proposes the usage of complex binary numbers during the learning process [17]. Vector-Valued Tree Parity

Machine (VVTPM) is the generalization of this idea and uses binary vectors as inputs [18]. The last improvement from this group is NBTPM, wherein the authors proposed the usage of integers instead of binary numbers [19]. NBTPM speeds up the synchronization of TPMs and introduces a new parameter M, which controls the variability of input vectors and adds an additional dimension for determining the most suitable TPM structure for users. Moreover, in [19], the authors discuss the effect of extrema of values and the reason for synchronization time reduction caused by the introduction of parameter M.

The application of TPM is presented in [20–22]. Sarkar et al. proposed the usage of TPMs in wireless networks and examined the energy consumption of the proposed solution [20]. The research conducted in [21] presents an error reconciliation protocol based on the mutual learning of neural networks in quantum cryptography. Gomez et al. presented hardware implementation of TPM [22].

Additionally, the security of TPMs has been studied extensively [23–25]. Ruttor et al. [23] and Shacham et al. [24] defined two different attacks and evaluated TPM security under them. The first attack is a standard man-in-the-middle genetic attack, and the second one is a geometric attack in which the attacker updates the weight of the neural network even when the outputs of benign parties do not match. Furthermore, in [25], the authors defined a probabilistic approach that could lead the evil party to gain some knowledge about a shared key.

Related research that involves the application of machine learning in quantum communication to achieve cryptographic keys is also presented in [26,27]. In [26], the authors define continuous-variable quantum key distribution and the framework for estimating the phase of the pilot signal. The framework is further expanded in [27]

3. Tree Parity Machine and Mutual Learning

Mutual learning in terms of TPMs is the key agreement process wherein two TPMs share information over a public medium and synchronize their weights. Once both neural networks achieve full synchronization, their weights can be used as a shared secret in further security operations. This section describes the details of the network's model, TPM learning and possible attack scenarios.

3.1. Model

TPM has a unique architecture. The network consists of two layers: an input layer and an output layer. Similar to the feed-forward network model, all of the input neurons are connected to exactly one output neuron. TPM inputs are grouped equally in number and are associated with only one hidden unit (σ_k). The outputs of all hidden units are connected to the output neuron (τ), which creates an output layer. The number of hidden units and the number of inputs per hidden unit are denoted by K and N, respectively. Every input has its associated weight (w_{kn}) that is bounded by range $< -L; L >$, where L is a positive integer. The learning rules guarantee that the values of the weights stay within the specified range. The activation function of every hidden unit is a modified signum function, and its formula is presented in (1). A and B denote the parties taking part in the mutual learning algorithm.

$$sgn'(x^{A/B}) = \begin{cases} 1, & x^A \geqslant 0 \vee x^B > 0 \\ -1, & x^A < 0 \vee x^B \leqslant 0 \end{cases} \tag{1}$$

Having defined the activation function, we are able to define the output of the k-th hidden unit σ_k, which is the sum of the products of weights and inputs. The formula is presented in (2).

$$\sigma_k = sgn'\left(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}\right) \tag{2}$$

The final output τ of TPM is a product of all hidden layer outputs and is defined in (3).

$$\tau = \prod_{k=1}^{K} \sigma_k \tag{3}$$

An example TPM model is presented in Figure 1.

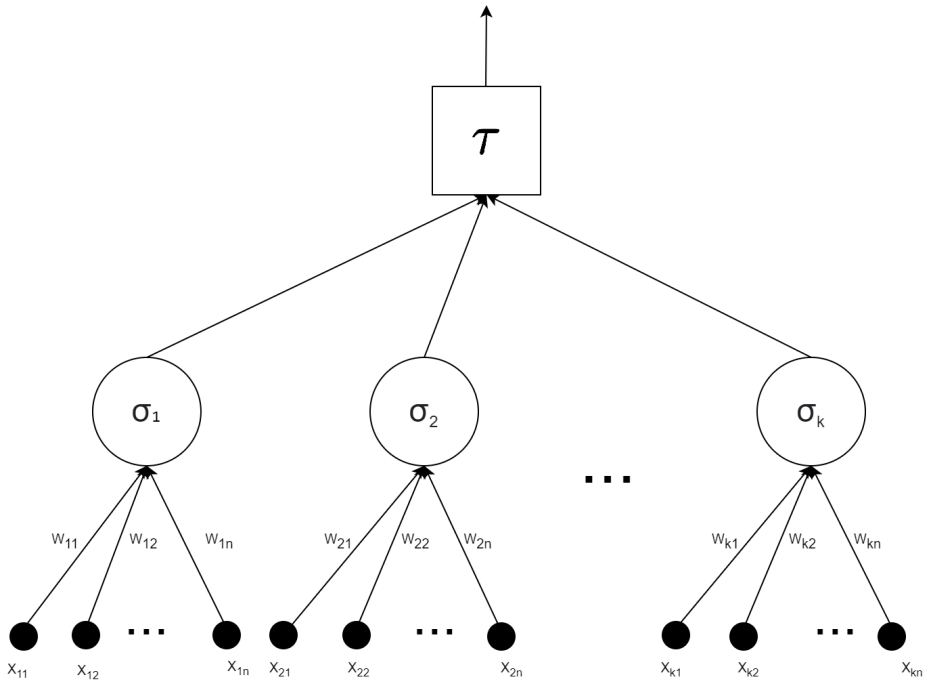


Figure 1. Structure of TPM model.

3.2. Synchronization and Learning Rules

Successful synchronization of TPM is a necessary part of key agreement protocol [28]. Both parties need to execute the following steps to distill the cryptographic key as a result of the whole process.

1. Both parties initialize their own TPM with randomly chosen weights. These neural networks must be the same variant of TPM and must share the same set of parameters uniquely defining its behavior, shape and size (i.e., K, L, M, N parameters).
2. Synchronizing parties randomly choose an input vector X consisting of $K \times N$ elements and share it via a public channel. Based on the selected TPM variant, the input vector consists either of binary values, complex binary values/vectors or integers.
3. Both participants calculate the output of their neural network and share it publicly.
4. The matching outputs are preliminary for updating the TPM weights. The learning rules are responsible for updating the weights, and users can choose one of the following:
    - Hebbian learning rule

$$w'_{kn} = w_{kn} + \tau x_{kn} \Theta(\sigma_k, \tau) \tag{4}$$

    - Anti-Hebbian learning rule

$$w'_{kn} = w_{kn} - \tau x_{kn} \Theta(\sigma_k, \tau) \tag{5}$$

- Random walk learning rule

$$w'_{kn} = w_{kn} + x_{kn}\Theta(\sigma_k, \tau) \tag{6}$$

where Θ is a function returning 1 if all of its arguments are equal and returns 0 otherwise, x_{kn}/w_{kn} denotes the k-th neuron n-th input/weight accordingly, and w'_{kn} denotes the updated weight value.

5. The parties repeat Steps 2–4 until full synchronization of their TPMs is achieved.

Once synchronization is complete, both neural networks are exactly the same. This results in the same weight vector that can be further applied in selected security operations, e.g., as a cryptographic key to encrypt confidential data using symmetric ciphers. All the notations used in the paper are presented in Appendix A in Table A1. The remaining part of this Section presents the training process of TPMs, which is later called synchronization, and attack vectors that might be used against cryptosystems using TPM.

3.3. Attack Scenarios

A man-in-the-middle attack is inherently the most efficient way of exploiting flaws in key agreement protocols. With regard to TPM, the described attack comes down to placing an eavesdropper between benign parties and learning an adversarial neural network based on captured communication. Three possible scenarios can happen when parties A and B are under a man-in-the-middle attack.

- $\tau_A \neq \tau_B$—neither benign TPMs nor the adversarial TPM proceed with synchronization.
- $\tau_A = \tau_B \neq \tau_C$—only TPMs A and B synchronize to each other, while the attacker cannot proceed.
- $\tau_A = \tau_B = \tau_C$—all the TPMs, including the adversarial one, update their weights accordingly.

The last of the described scenarios brings the evil TPM close to performing a successful attack. However, these conditions occur significantly less frequently than the others.

While neural cryptography is quantum-proof, there exist other attack vectors that pose a threat to the security of TPM. The prerequisite for all the attacks presented in this subsection is the sharing of input vectors and neural network outputs via an insecure public channel. This allows an evil party E to eavesdrop on the exchanged information. Possible attacks under these circumstances are as follows [25].

1. Man-in-the-middle attack—the simplest attack, wherein an evil party eavesdrops on the communication between benign parties. Based on this, entity E is able to synchronize the TPM. The synchronization only happens when $\tau_A = \tau_B = \tau_E$, where τ_A and τ_B denote the outputs of the benign parties' TPMs. The attacker may increase the odds of a successful attack by synchronizing more than one neural network.
2. Geometric/flipping attack—an attack that uses a geometric representation of inputs and a weight vector in N-dimensional space. This is an improved man-in-the-middle attack using an additional step that happens when $\tau_A = \tau_B$ but $\tau_A \neq \tau_E$. During this step, the attacker finds the i-th hidden unit with the lowest $\sum_{n=1}^{N} x_{in} \cdot w_{in}$. Once such a hidden unit is found, the evil party flips the output σ_i and applies the learning rule while pretending the outputs τ_A, τ_B, τ_E are equal.
3. Majority attack—an attacker uses a set of TPMs (I) and performs a geometric attack on all of them. However, just before the weight update, the voting process begins on the most common output vector $(\sigma_1^i, \sigma_2^i, \cdots, \sigma_K^i)$ that will be used in the weight update step. All the evil TPMs become more and more similar to each other as the attack progresses, which might be a flaw of the attack scenario. Using the majority attack and the flipping attack interchangeably might mitigate this issue.
4. Genetic attack—a genetic attack uses at most I TPMs to intercept key establishment. While the attacker has fewer than $I/2^{K-1}$, after every step, new 2^{K-1} are created with all possible $(\sigma_1^i, \sigma_2^i, \cdots, \sigma_K^i)$ hidden unit values. Once the limit $I/2^{K-1}$ has been

reached, only the fittest TPMs should remain. The algorithm's success is highly dependent on the fitness function, which determines which neural networks remain between generations.

4. Results

The following section presents the results of testing the security of TPM using non-binary input vectors. This section is split into two parts. The first presents the research methodology, and the second presents the collected results and their analysis.

4.1. Methodology

The conducted research consists of simulations that run attack scenarios on two benign TPMs. Benign TPMs perform the key agreement protocol described in Section 3.2. The attack scenario assumes that an attacker is able to eavesdrop on the communication. Based on this, we performed the attacks defined in Section 3.3. To mimic the conditions of an insecure channel, all the exchanged information between benign parties is also delivered to the malicious party responsible for the attack. Every scenario is repeated 1000 times. Many different sizes and parameters of TPMs are tested against man-in-the-middle attacks. The considered parameters are as following:

- $K = 3$,
- $L \in \{9, 10, 11, 12\}$,
- $M \in \{1, 2, 3, 4, 5\}$,
- $N = 50$.

By using these parameter selections, it is possible to evaluate how different values of parameter M affect various TPM forms (for $M = 1$ NBTPM becomes standard TPM defined in [2]) and to determine if parameter M improves the overall security. Furthermore, utilizing the selected parameters enables the attainment of a secure key length of approximately 500 bits [28].

The selected attacks against TPMs seem to be the most popular and successfully performed ones. Additionally, the attack vectors consist solely of man-in-the-middle attacks, as this is the main attack vector for key agreement protocols. For every attack, the eavesdropper is allowed to have a maximum of 50 neural networks. We consider the synchronization as failed when the eavesdropper is able to achieve a synchronization score equal to 1 for any of the evil TPMs. This means at least one neural network has perfectly synchronized its weights to either of the two TPMs performing mutual learning. The source code of the simulation framework is release to the public space (the source code of the simulation framework is available at https://github.com/mstypinski/tpm (accessed on 8 May 2023).)

During simulations, the number of total iterations, number of weight update steps, synchronization scores of benign TPMs and synchronization scores of the best-synchronized adversarial TPMs are gathered. The formula for synchronization score is defined in (7) [19]. Additionally, the index of synchronization when S_{score} of one evil TPMs is equal to 1 is also collected. In cases where the adversarial party does not fully synchronize any neural network, this value is equal to the total number of synchronization iterations in a particular scenario. All the results are presented along with 95% confidence intervals.

$$S_{score} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta(w_{kn}, w_{kn}^{A})}{K \times N} \tag{7}$$

A genetic attack is a special case of man-in-the-middle attacks. It needs a fitness function that selects the proper individuals across whole populations for the next iteration of the genetic algorithm. The function used for this purpose is:

$$f = max(S_{score_A}, S_{score_B}). \tag{8}$$

In other words, this is a perfect fitness function since it selects the neural networks that have the most common weights with either of the two benign TPMs. In real-world scenarios, this function cannot be implemented because the attacker would need to know the exact weights of the legitimate neural networks. Moreover, the algorithm is allowed to select at most 50 neural networks that persist between iterations of attacks. This means that, internally, the population can be larger and can reach up to $50 \times 2^{K-1}$ individuals.

4.2. Results

Similar to the research conducted by the authors in [19], the benign TPMs synchronize in a faster manner by increasing parameter M. Figure 2 presents the number of required iteration steps and successful synchronizations for all considered neural networks. An iteration in the context of a mutual learning algorithm refers to a single cycle of the algorithm. Synchronization, on the other hand, refers to an iteration that results in an update of the weights for benign TPMs.

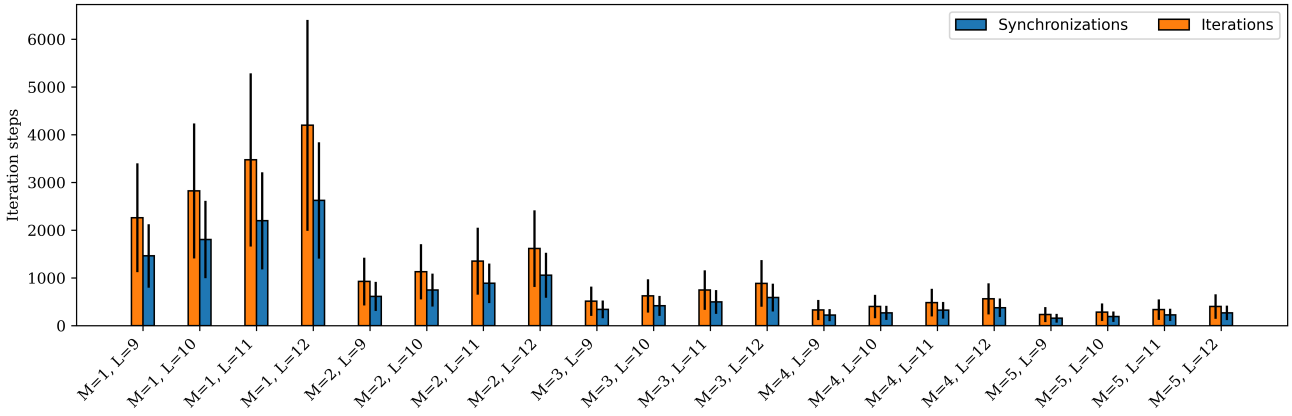


Figure 2. Number of key-exchange algorithm synchronizations and iterations of benign TPMs.

Increasing parameter M improves the security features of TPMs in terms of how quickly the parties are able to synchronize; hence, they can achieve a longer cryptographic key in the same amount of time. However, increasing parameter M also improves the eavesdropper's ability to retrieve the hidden state of the neural network. Figure 3 depicts the average index of iterations at which a malicious neural network executed the attack successfully, later called the adversarial synchronization point. The most notable is the average synchronization index difference between $M = 1$ and $M = 2$, which means the eavesdropper is able to synchronize more effectively with increases to parameter M. However, taking into consideration the decrease in the number of required iterations (Figure 2), the ratio between the median adversarial synchronization point and the median number of required synchronizations stays approximately the same.

Furthermore, it can be observed that the genetic attack performs significantly better than the other three attacks. This is due to the fact that the fitness function acts as an oracle-like determinant, selecting the TPMs most similar to the benign neural networks. To create a fitness function that is applicable in real-world scenarios, one may maximize or minimize the difference between the product of hidden outputs (σ_k) and weights and the input dot product ($w_k \cdot x_k$). The impact of different fitness functions on attack success probability requires further research. Additionally, while the attacker selects only the 50 most-fit TPMs that persist between generations, the generation may consist of more than 50 neural networks. This results in a meaningful advantage over the other attacks.

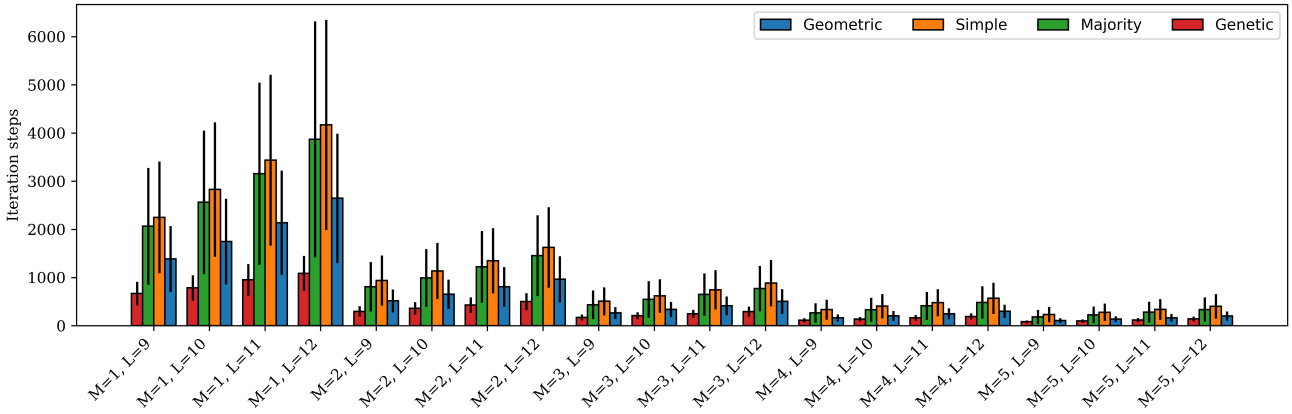


Figure 3. Average synchronization index at which evil party fully synchronizes its TPM.

The attacks against the examined TPM with the aforementioned set of parameters are effective. The least effective attack is the simple attack performed with many adversarial neural networks. Only when parameter $M = 5$ was the malicious party able to retrieve the cryptographic key. This result follows the outcomes from the initial research of NBTPM [19].

The majority attack had a success rate varying from almost 0.2 for $M = 1$, $L = 13$ up to 0.45 for $M = 5$, $L = 9$. The attack success probability rises with increases to parameter M; however, at the same time, the synchronization time decreases. The median of mutual learning iterations decreases more than twice for parameters $M = 1$ and $M = 2$, while the attack probability rises by at most 20.6%. Further, the attack success probability decreases with the increase to parameter L. This is true for all the attacks except the genetic attack. However, the greater values of parameter L result in more key-exchange algorithm iterations.

The geometric attack is successful with 0.8 probability in almost all scenarios. The conclusions from the majority attack are similar to those from the geometric attack. While the attack probability increases by at most 10%, the median number of required iterations decreases by at least 60% for parameters $M = 1$ and $M = 2$. Unfortunately, the success rate being at least 0.8 renders TPMs with the considered parameters ineffective in real-world scenarios. To avoid such situations, it is necessary to make more parameter adjustments, such as increasing the value of parameter L, which leads to a decrease in the median probability of successful geometric attacks. Unfortunately, these adjustments may result in an unwanted increase in synchronization time, which can be observed in the results.

The final attack evaluated is the genetic attack. This attack is successful in the majority of scenarios. However, it is important to note that this level of performance can only be achieved through the use of an ideal fitness function, which is not practical in real-world situations. The minimal increase in resistance to attacks observed for $M = 5$ is due to the greater standard deviation of the average number of mutual learning iterations. This implies that in certain scenarios, a benign TPM may synchronize at a significantly faster rate than an adversarial TPM.

The introduction of parameter M enhances the flexibility of TPMs in practical situations. A value of $M = 1$ corresponds to the standard version of a TPM. As M increases, the required number of iterations drops significantly, particularly for values of $M = 2$ and $M = 3$. The worst-case scenario for $M = 3$ occurs when $L = 11$. Despite a 35% increase in median attack success rate when compared to $M = 2$, there is a 45% reduction in median mutual learning algorithm iterations. The specific results for all simulated scenarios are presented in Table 1.

Table 1. Synchronization time and attack success probability.

		Mutual Learning Iterations				Attacks (S_{score})			
M	**L**	**Mean**	**Median**	**Max**	**Min**	**Simple**	**Majority**	**Geometric**	**Genetic**
1	9	2263.19 ± 1139.781	2182	5711	890	0.0	0.228	0.816	0.999
1	10	2823.69 ± 1413.413	2739	8478	1152	0.0	0.204	0.822	1.0
1	11	3475.25 ± 1814.419	3331	9087	1440	0.0	0.211	0.819	0.999
1	12	4198.83 ± 2209.911	4052	12608	1869	0.0	0.196	0.799	1.0
1	13	5064.37 ± 2820.405	4826	13067	1751	0.0	0.186	0.763	1.0
2	9	927.72 ± 500.816	893	2342	302	0.0	0.271	0.899	0.995
2	10	1131.29 ± 579.344	1098	2847	445	0.0	0.246	0.882	1.0
2	11	1353.59 ± 700.158	1306	3690	558	0.0	0.221	0.850	0.997
2	12	1617.92 ± 802.934	1568	4584	651	0.0	0.215	0.871	0.999
2	13	1908.82 ± 951.497	1839	4434	799	0.0	0.206	0.832	0.999
3	9	514.64 ± 308.012	489	1559	200	0.0	0.330	0.938	0.996
3	10	625.88 ± 348.143	598	1509	238	0.0	0.291	0.917	0.998
3	11	749.19 ± 413.828	714	1783	259	0.0	0.298	0.911	0.999
3	12	888.02 ± 487.737	853	2626	359	0.0	0.282	0.900	0.997
3	13	1031.67 ± 534.667	991	2616	435	0.0	0.235	0.892	0.999
4	9	331.62 ± 209.76	314	986	128	0.0	0.395	0.952	0.998
4	10	403.55 ± 245.221	383	1020	138	0.0	0.360	0.949	0.997
4	11	485.4 ± 289.757	461	1298	177	0.0	0.327	0.941	0.996
4	12	566.62 ± 325.635	544	1568	229	0.0	0.316	0.926	0.998
4	13	661.08 ± 367.354	631	1707	270	0.0	0.322	0.915	0.998
5	9	235.25 ± 158.482	219	695	85	0.003	0.452	0.963	0.998
5	10	285.76 ± 184.754	267	830	96	0.001	0.410	0.966	0.995
5	11	339.3 ± 214.747	320	1043	121	0.001	0.364	0.961	0.997
5	12	403.74 ± 254.932	379	1099	151	0.001	0.348	0.948	0.997
5	13	459.22 ± 272.12	438	1449	171	0.0	0.338	0.938	0.997

5. Conclusions

Quantum-proof cryptography is a growing problem in modern communications. The recent advancements in efficient factorization using quantum algorithms have highlighted the need for secure key agreement protocols. TPMs such as NBTPM have the potential to meet this need; however, further testing and optimization of parameters are necessary to apply it to production-ready smart grid systems.

In this paper, the authors investigated the security of NBTPM under different types of man-in-the-middle attacks: simple, geometric/flipping, majority and genetic. While the TPMs we considered did not attain perfect security, the parameter M provides another dimension for optimizing the TPM architecture for increased security. In particular, a significant improvement might be noticed by increasing parameter M from a value of 1 to 2 and from 2 to 3. Accordingly, while the median attack success rate increases by at most 21% and 35%, respectively, the median number of mutual learning iterations is reduced by at least 58% and 45%, respectively; hence, the conclusion can be drawn that increasing parameter M along with fine-tuning other parameters improves the overall security of the considered solution. Choosing a specific value for parameter M must be done by considering all the pros and cons that follow specific selection. Further, while the increase to M improves efficiency, it cannot be increased infinitely, as security features degrade for large M values that approach the L parameter value. It is worth mentioning that some applications of TPMs require even fewer interactions than assumed (e.g., TPMs used for error correction in quantum cryptography).

In order to implement the proposed solution in a real-world scenario, further parameter adjustment is necessary. It is important to approach the parameter M with caution, as it improves synchronization time but can also enhance the efficiency of adversarial synchronization. The number of inputs per neuron and the number of hidden neurons in

the intermediate layer of TPMs should also be adjusted carefully. Another approach to applying the described solution successfully in a smart grid could be an implementation of authenticated TPMs. Further research in this field should focus on exploring a wider range of parameters, including the number of hidden neurons in the intermediate layer, in order to find more secure variants of TPMs.

Appendix A

Table A1 presents the list of all notations used in the paper and their descriptions.

Table A1. Table of notations.

Variable	Description
K	TPM parameter denoting the number of neurons in a hidden layer
L	TPM parameter denoting the maximum value the weight modulus can take
M	NBTPM parameter denoting the maximum value the input vector element modulus can take
N	TPM parameter denoting the number of inputs per neuron in a hidden layer
x_{kn}	Input value of n-th input of k-th neuron
τ	Output of TPM
σ_k	Output of k-th hidden neuron
Θ	Function returning 1 if all its arguments are equal and 0 otherwise
w_{kn}	Weight value corresponding to n-th input of k-th neuron
w'_{kn}	Next iteration value of w_{kn}
S_{score}	Synchronization score

Author Contributions: Conceptualization, M.S. and M.N.; methodology, M.S.; software, M.S.; validation, M.S.; formal analysis, M.S.; investigation, M.S. and M.N.; writing—original draft preparation, M.S.; writing—review and editing, M.S. and M.N.; visualization, M.S.; supervision, M.N.; project administration, M.N.; funding acquisition, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: Research project partly supported by the program "Excellence initiative—research university" for the AGH University of Krakow.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study—the complete results collected during the simulations—are available online at: https://github.com/mstypinski/tpm_attack_results (accessed on 5 May 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Niemiec, M.; Dziech, A.; Stypiński, M.; Derkacz, J. Quantum-Based Solutions for the Next-Generation Internet. *Inf. Secur. Int. J.* **2019**, *43*, 62–72. [CrossRef]
2. Metzler, R.; Kinzel, W.; Kanter, I. Interacting neural networks. *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip Top.* **2000**, *62*, 2555–2565. [CrossRef]
3. Kinzel, W.; Metzler, R.; Kanter, I. Dynamics of interacting neural networks. *J. Phys. A Math. Gen.* **2000**, *33*, L141. [CrossRef]
4. Ghiasi, M.; Niknam, T.; Wang, Z.; Mehrandezh, M.; Dehghani, M.; Ghadimi, N. A comprehensive review of cyber-attacks and defense mechanisms for improving security in smart grid energy systems: Past, present and future. *Electr. Power Syst. Res.* **2023**, *215*, 108975. .: 10.1016/j.epsr.2022.108975. [CrossRef]
5. Ghiasi, M.; Dehghani, M.; Niknam, T.; Kavousi-Fard, A.; Siano, P.; Alhelou, H.H. Cyber-Attack Detection and Cyber-Security Enhancement in Smart DC-Microgrid Based on Blockchain Technology and Hilbert Huang Transform. *IEEE Access* **2021**, *9*, 29429–29440. [CrossRef]
6. Mohan, A.M.; Meskin, N.; Mehrjerdi, H. A Comprehensive Review of the Cyber-Attacks and Cyber-Security on Load Frequency Control of Power Systems. *Energies* **2020**, *13*, 3860. [CrossRef]
7. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]

8. Barker, E.; Chen, L.; Roginsky, A.; Vassilev, A.; Davis, R. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. [CrossRef]
9. Fernández-Caramés, T.M. From Pre-Quantum to Post-Quantum IoT Security: A Survey on Quantum-Resistant Cryptosystems for the Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 6457–6480. [CrossRef]
10. Kanter, I.; Kinzel, W.; Kanter, E. Secure exchange of information by synchronization of neural networks. *Europhys. Lett.* **2002**, *57*, 141. [CrossRef]
11. Ruttor, A.; Reents, G.; Kinzel, W. Synchronization of random walks with reflecting boundaries. *J. Phys. A Math. Gen.* **2004**, *37*, 8609. [CrossRef]
12. Ruttor, A. Neural Synchronization and Cryptography. Ph.D. Thesis, University of Würzburg, Würzburg, Germany, 2007,
13. Allam, A.M.; Abbas, H.M.; El-Kharashi, M.W. Authenticated key exchange protocol using neural cryptography with secret boundaries. In Proceedings of the The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–8. [CrossRef]
14. Santhanalakshmi, S.; K., S.; Patra, G.K. Analysis of Neural Synchronization Using Genetic Approach for Secure Key Generation. In *Proceedings of the Security in Computing and Communications*; Abawajy, J.H., Mukherjea, S., Thampi, S.M., Ruiz-Martínez, A., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 207–216.
15. Sarkar, A.; Khan, M.Z.; Singh, M.M.; Noorwali, A.; Chakraborty, C.; Pani, S.K. Artificial Neural Synchronization Using Nature Inspired Whale Optimization. *IEEE Access* **2021**, *9*, 16435–16447. [CrossRef]
16. Dolecki, M.; Kozera, R. The Impact of the TPM Weights Distribution on Network Synchronization Time. In *Proceedings of the Computer Information Systems and Industrial Management*; Saeed, K., Homenda, W., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 451–460.
17. Dong, T.; Huang, T. Neural Cryptography Based on Complex-Valued Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4999–5004. [CrossRef] [PubMed]
18. Jeong, S.; Park, C.; Hong, D.; Seo, C.; Jho, N. Neural Cryptography Based on Generalized Tree Parity Machine for Real-Life Systems. *Secur. Commun. Netw.* **2021**, *2021*, 6680782. [CrossRef]
19. Stypiński, M.; Niemiec, M. Synchronization of Tree Parity Machines Using Nonbinary Input Vectors. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–7. [CrossRef] [PubMed]
20. Sarkar, A.; Dey, J.; Bhowmik, A. Multilayer neural network synchronized secured session key based encryption in wireless communication. *Indones. J. Electr. Eng. Comput. Sci.* **2019**, *14*, 169. [CrossRef]
21. Niemiec, M. Error correction in quantum cryptography based on artificial neural networks. *Quantum Inf. Process.* **2019**, *18*, 174 . [CrossRef]
22. Gomez, H.; Óscar, R.; Roa, E. A 65nm CMOS key establishment core based on tree parity machines. *Integration* **2017**, *58*, 430–437. .: 10.1016/j.vlsi.2017.01.010. [CrossRef]
23. Ruttor, A.; Kinzel, W.; Naeh, R.; Kanter, I. Genetic attack on neural cryptography. *Phys. Rev. E* **2006**, *73*, 036121. [CrossRef]
24. Shacham, L.N.; Klein, E.; Mislovaty, R.; Kanter, I.; Kinzel, W. Cooperating attackers in neural cryptography. *Phys. Rev. E* **2004**, *69*, 066137. [CrossRef]
25. Klimov, A.; Mityagin, A.; Shamir, A. Analysis of Neural Cryptography. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2002*; Zheng, Y., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 288–298.
26. Chin, H.M.; Jain, N.; Zibar, D.; Andersen, U.L.; Gehring, T. Machine learning aided carrier recovery in continuous-variable quantum key distribution. *NPJ Quantum Inf.* **2021**, *7*, 20. [CrossRef]
27. Liu, Z.P.; Zhou, M.G.; Liu, W.B.; Li, C.L.; Gu, J.; Yin, H.L.; Chen, Z.B. Automated machine learning for secure key rate in discrete-modulated continuous-variable quantum key distribution. *Opt. Express* **2022**, *30*, 15024–15036. [CrossRef] [PubMed]
28. Stypiński, M.; Niemiec, M. Impact of Nonbinary Input Vectors on Security of Tree Parity Machine. In *Proceedings of the Multimedia Communications, Services and Security*; Dziech, A., Mees, W., Niemiec, M., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 94–103.

Impact of Nonbinary Input Vectors on Security of Tree Parity Machine

Miłosz Stypiński[0000−0001−7137−3195] and Marcin Niemiec[0000−0002−3909−9592]

AGH University of Science and Technology
Institute of Telecommunications
Mickiewicza 30, 30-059 Krakow, Poland
stypinski@agh.edu.pl

Abstract. Key agreement protocol is an essential step for establishing a secure connection. The inevitable advancements in quantum computing technologies pose a huge threat to the key agreement protocol in use today. Neural cryptography is an alternative key agreement protocol that is not susceptible to any known quantum algorithm. Since the invention of mutual learning of TPM, many improvements have been proposed. One of them was the usage of nonbinary input vectors. This study verifies the impact of nonbinary input vectors on TPM security features. A number of iterations, similarity to intruder's TPM and effective key length were taken into account. The results show that the choice between fast synchronization times and higher security of the final key must be performed with perfect care.

Keywords: Mutual Learning · Neural Networks · Key Agreement · Cybersecurity

1 Introduction

The very first algorithm for key exchange based on asymmetric cryptography was the Diffie-Hellman algorithm [2]. It allows the exchange of cryptographic key over an insecure channel between two parties in network environment. While no eavesdropper can recover the distilled cryptographic key, the entire algorithm is vulnerable to a man-in-the-middle attack. However, we handle this problem in practice by taking advantage of public key cryptography. This allows the parties to authenticate each other, which ensures that only the respective entities cooperate to build the final key. The first proposed algorithm allowing public key cryptography was the RSA algorithm [10].

All asymmetric cryptography algorithms come with one inherent drawback. The security they provide is conditional. The conditional security means there exists well-known ways to break the security of these algorithms, but this is not possible in a time-efficient manner with the current computing power available. The algebraic problem used in the Diffie-Hellman algorithm is the discrete logarithm problem, and the RSA algorithm is based on the factorization problem of large prime numbers.

The inevitable revolution that quantum computers will bring will also have a significant impact on the current state of cryptography. Quantum computers operate on qubits rather than on standard bits. Qubits are a superposition of two quantum states which means that they can be in both states simultaneously. This feature allows implementation of quantum algorithms that are significantly faster compared to the standard algorithm counterpart. This is the case for currently used asymmetric cryptography algorithms. Shor's algorithm is an algorithm that reduces the time needed to find a solution for both of the previously mentioned algebraic problems [14]. Many solutions have been proposed based on problems that currently do not have an effective way of solving them on quantum computer. One of them is Tree Parity Machine (TPM) discussed in this article, in particular its variant with non-binary input vectors.

2 Tree Parity Machine

This section introduces basics of the TPM and describes the current status of this special form of artificial neural networks. Additionally, it covers the security considerations.

2.1 Related works

Numerous research has been conducted in the field of neural network applications in cybersecurity. The first study showing that two TPMs are able to synchronize in finite time was conducted by Kanter *et al.* [7] and Rosen-Zvi *et al.* [11]. The very first application of neural networks on the subject of key agreement was proposed in [5]. This has become the basis for many improvements and applications of TPM. In [13] authors proposed an enhancement to the synchronization process by finding the best-fit weight vector using a genetic algorithm. Alam *et al.* in [1] proposed an improvement in mutual learning based on sending erroneous output bits during mutual learning. Furthermore, TPM finds applications in error reconciliation. In [9] studies have proven that neural networks successfully find and reconcile errors in quantum key distribution protocols. Other improvements change the value range of the input vectors. In [3, 4, 15] modifications called Complex-Valued Tree Parity Machine (CVTPM), Vector-Valued Tree Parity Machine (VVTPM) and Nonbinary Tree Parity Machine (NBTPM) were proposed accordingly. The latter variant of TPM is used as reference model in this paper and the names TPM and NBTPM are used interchangeably hereafter. All the concepts presented are the same for both TPM and NBTPM except for random input generation. For TPM all the input values are selected from $-1, 1$ while for NBTPM vector values are selected from $< -M, M >$ range. Details are provided later in this section.

While all CVTPM, its generalization VVTPM and NBTPM change the values the input vectors can take, the difference between the first two and the last one is significant. VVTPM effectively performs mutual learning of multiple TPMs where the number of synchronized TPMs is bound by the length of

an exchanged vector. NBTPM synchronizes exactly one TPM but in a faster manner.

2.2 Architecture of Tree Parity Machine

Basically, TPM is just a two-layer perceptron with K neurons in the first layer and only one neuron at the output of the neural network. The inputs are exclusively connected to only one neuron. The weights associated with corresponding inputs are integers varying from $-L$ to L. Each neuron in the first layer has N inputs. In [12] the TPM architecture and all the algorithms allowing for successful key agreement are presented in detail.

In the first design of TPMs, the inputs were binary vectors consisting of $K \times N$ elements. However, in this paper, we only consider TPMs with nonbinary input vectors (NBTPM) presented in [15]. Hence, the input vector elements are integers varying from $-M$ to M. The four parameters K, L, M and N uniquely define NBTPM.

NBTPM output is a binary value. To ensure that the output value is either -1 or 1 custom activation functions are used in the network. The first layer neuron activation function is a modified signum function and is defined in (1).

$$activation(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \tag{1}$$

The final output of $k-th$ neuron is defined in (2) where x_{kn} denotes the $n-th$ input of $k-th$ neuron and w_{kn} is its corresponding weight.

$$\sigma_k = activation(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}) \tag{2}$$

The final output of TPM is the product of the first layer output and is presented in (3).

$$O = \prod_{k}^{K} \sigma_k \tag{3}$$

The example TPM with 3 neurons and 5 inputs per neuron is presented in Figure 1.

2.3 Tree Parity Machine key agreement protocol

To establish a secure key via an insecure channel, the parties must follow a key agreement protocol. This is accomplished by sharing some information publicly between entities performing key agreement. However, the key agreement protocol must ensure that any eavesdropper will not be able to retrieve the key from publicly sent messages.

Let us suppose Alice and Bob want to exchange the secret using TPMs. To achieve this goal by using nonbinary input TPMs they need to perform mutual learning algorithm described below.

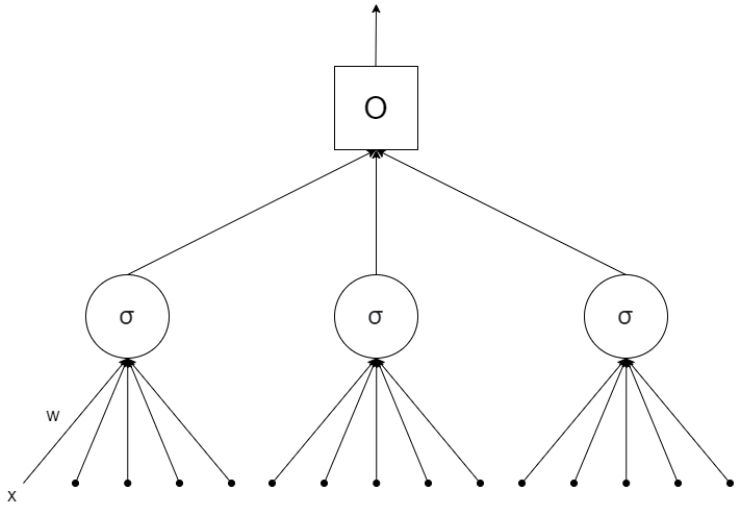


Fig. 1. TPM with $K = 3$ and $N = 5$

1. Both Alice and Bob initialize the weights w_{kn} in their private TPM with random integer within $< -L; L >$ range.
2. Either Alice or Bob generate an input vector X and share it over a public insecure channel. Every vector element x_{kn} is an integer varying from $-M$ to M.
3. After sharing the input vector parties compute the response O of their network using (1) - (3) and share it via public channel.
4. If the responses are the same Alice and Bob will apply one of the learning rules. Otherwise, the input vector is discarded and the process is repeated starting from step 2. The learning rules depend on the chosen algorithm. Examples are presented below.
   (a) *Hebbian learning rule*

$$w_{kn}^{t+1} = w_{kn}^{t} + O^t x_{kn}^t \Theta(\sigma_k^t, O^t) \tag{4}$$

   (b) *Anti-Hebbian learning rule*

$$w_{kn}^{t+1} = w_{kn}^{t} - O^t x_{kn}^t \Theta(\sigma_k^t, O^t) \tag{5}$$

   (c) *Random walk learning rule,*

$$w_{kn}^{t+1} = w_{kn}^{t} + x_{kn}^t \Theta(\sigma_k^t, O^t) \tag{6}$$

   The parameter t denotes the current iteration of the algorithm and Θ is a function which returns 1 if both parameters are the same, and returns 0 otherwise.

5. Steps 2-4 are repeated until the full synchronization is achieved. Fully synchronized TPMs have equal corresponding weights. The vector W consisting of weights w_{kn} might be later used in other security algorithms, e.g. as a shared key in symmetric cryptography.

Both successful and unsuccessful iterations of the algorithm require exactly 3 message exchanges between the parties. Therefore, the algorithm should be finished in the least possible number of steps to prevent eavesdropping parties from recovering the key and to minimize the duration of the key agreement process.

2.4 Security of Tree Parity Machine

Key agreement protocols allow sharing keys over a public insecure channel in such a manner no eavesdroppers are able to recover the cryptographic key from the messages exchanged. Furthermore, the exact algorithm of key agreement protocols are openly available. If no intruder is able to extract the agreed key from information provided, we call it as a secure protocol.

The security of TPM has been the subject of many studies. In [6] Kinzel *et al.* showed that TPM with less than 3 hidden neurons in the first layers provides significantly less security than a neural network with 3 or more neurons. Moreover, in [8] the authors defined the following attacks to which TPM might be susceptible.

- Man-in-the-middle attack – an attack in which an evil party synchronizes one or more TPMs. The evil TPM successfully performs mutual learning in the case when $O_A = O_B = O_E$, where O_E denotes evil TPM output. This attack allows the eavesdropper to synchronize their weight by 60% on average based on research.
- Brute force attack – there have not been proposed any attack that would allow recovering the security key in time-efficient manner.
- Genetic algorithms attack – the attacker utilizes genetic algorithm to predicts weights. Based on conducted research mostly neural networks with only one input neuron are susceptible to this attack.
- Classification-based attack using neural networks attack – studies show that a properly trained Artificial Neural Network (ANN) is able to predict the sign of every weight with approximately 100% accuracy. This means the evil party using this technique with a brute force attack might be able to break the algorithm in half of the time required by just a brute force attack.

To check how changing the parameters affects the security of TPMs, all the simulations presented in Section 3 are performed in an insecure environment. All the exchanged messages are eavesdropped on by an intruder carrying out a man-in-the-middle attack.

3 Results

In this section, we check the impact of different parameter values on the TPM security. To check the quality of distilled key the entropy-based measure is presented in Section 3.1. The collection of results was performed on a dedicated environment which description is presented in Section 3.2. The last Section 3.3 contains the analysis of collected results.

3.1 Key quality measurement methodology

After mutual learning is completed both parties have the same TPMs. Weights create a shared key. The length of the key is impacted by the initial parameters of TPM and is presented in (7).

$$Key_{length} = K \cdot \ N \cdot \ log_2(2L+1) \tag{7}$$

However, the key length equation (7) only applies in an ideal scenario where the value distribution of weights is uniform. Studies show that the distribution of weight values in real-world scenario is not consistent with the uniform distribution. The updated key length formula that takes into account the uneven distribution of weights is presented in (8)[15]. In the equation, E denotes the estimated entropy which is calculated based on simulation results and takes into account probability distribution of weights. Additionally, the key length is an integer bounded by (8), therefore the floor is applied to the right side of the equation.

$$Key_{length} = \lfloor K \cdot N \cdot E(W) \rfloor \tag{8}$$

Furthermore, the performed simulations assume that an eavesdropper is trying to synchronize evil TPM. A similarity measure is required to determine how far evil TPM has been from fully synchronizing with the parties performing key agreement. The number of the same corresponding weights suits these requirements later being called the synchronization score. Its formula is presented in (9). In the equation w^A_{kn} denotes the weight vector of the adversarial TPM. The $Sync_{score}$ varies from 0 to 1, where the measure returns 1 for fully synchronized neural networks and 0 for TPMs where any pair of corresponding weight is not equal to.

$$Sync_{score} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta(w_{kn}, w^A_{kn})}{K \times N} \tag{9}$$

3.2 Simulation environment

The simulation environment mimics the real-world scenarios where two parties willing to exchange key perform mutual learning. The whole communication is eavesdropped on by the intruder who is trying to retrieve the key by performing a man-in-the-middle attack. The simulation consists of different scenarios in which different sets of parameters are used. The parameters tested are $M \in$

$\{1, 2, 3, 4, 5\}$ and $L \in \{9, 10, 11, 12, 13\}$. The parameters K an N are constant and are set to 3 and 50 accordingly. The results are gathered over 1000 simulations and include a number of iterations required to fully synchronize TPMs, key length calculated by (8) and evil TPM weight vector similarity calculated by (9). All the results are annotated with 95% confidence intervals.

3.3 Simulation results

The simulation results section is divided into three following parts: analysis of the number of iterations, average evil TPM similarity measurement and comparison of maximum and effective key length.

The average number of iterations, successful iterations and evil TPM iterations are presented in Figure 2. Increasing the M parameter significantly reduces the required number of iterations required to share the cryptographic key successfully. This results in a faster and more secure key agreement protocol. Noteworthy is the considerable difference in required and successful iterations between the parameters $M = 1$ and $M = 2$. This change of parameters provides the most meaningful improvement. Further changes to the parameter still increase efficiency, however the decrease of required number of iterations is less significant.

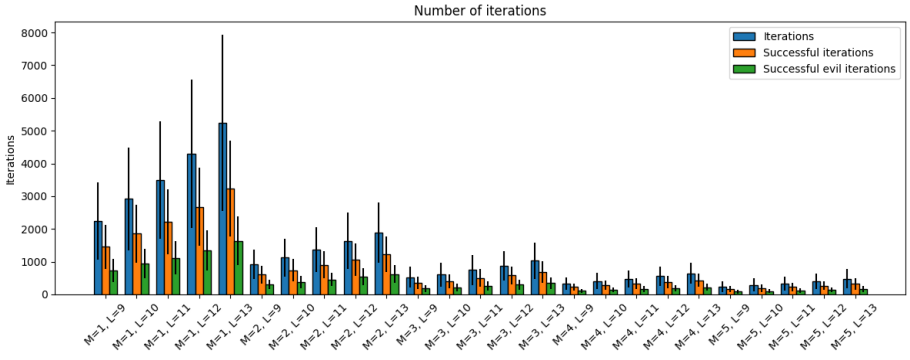


Fig. 2. Average number of iterations, successful iterations and evil iterations

The increase of parameter M also results in faster synchronization of evil TPM. Table 1 presents the average synchronization score for the eavesdropping party, including minimum, maximum and median values. For legitimate parties performing mutual learning, the $Sync_{score}$ will be equal to 1. In the average and median values, only minor growth is noticeable but yet the range of confidence interval increases by a magnitude twenty times. Furthermore, the maximum evil party $Sync_{score}$ more than doubles for parameters $M = 1, L = 9$ and $M = 5, L = 9$. The larger the values of parameter M, the higher the probability of

full synchronization of evil NBTPM. This situation would result in cryptographic key compromise and should be avoided at all costs.

Table 1. Evil TPM synchronization score

		Synchronization score $Sync_{score}$			
M	L	Average	Min	Max	Median
1	9	0.12 ± 0.07	0.047	0.24	0.113
1	10	0.12 ± 0.068	0.033	0.26	0.113
1	11	0.11 ± 0.068	0.033	0.227	0.1
1	12	0.1 ± 0.062	0.04	0.22	0.1
1	13	0.1 ± 0.069	0.033	0.227	0.1
2	9	0.12 ± 0.088	0.04	0.32	0.113
2	10	0.11 ± 0.074	0.04	0.24	0.107
2	11	0.11 ± 0.076	0.033	0.253	0.107
2	12	0.1 ± 0.073	0.033	0.227	0.1
2	13	0.1 ± 0.066	0.027	0.213	0.093
3	9	0.13 ± 0.098	0.047	0.387	0.12
3	10	0.12 ± 0.089	0.027	0.333	0.113
3	11	0.12 ± 0.088	0.04	0.3	0.113
3	12	0.1 ± 0.076	0.033	0.233	0.1
3	13	0.1 ± 0.062	0.02	0.2	0.093
4	9	0.14 ± 0.116	0.04	0.467	0.13
4	10	0.12 ± 0.093	0.027	0.447	0.12
4	11	0.12 ± 0.09	0.02	0.267	0.113
4	12	0.11 ± 0.099	0.047	0.313	0.1
4	13	0.11 ± 0.083	0.04	0.253	0.107
5	9	0.16 ± 0.129	0.053	0.593	0.14
5	10	0.14 ± 0.111	0.047	0.393	0.133
5	11	0.13 ± 0.121	0.02	0.36	0.12
5	12	0.12 ± 0.101	0.033	0.407	0.113
5	13	0.12 ± 0.096	0.04	0.34	0.107

Additionally, with the increase of parameter L, the number of required iteration raises to fully synchronize both legitimate NBTPMs. However, simultaneously the growth of parameters L directly extends the cryptographic key. The relationship between parameters of TPM and maximum/effective key length is presented in Figure 3. Furthermore, with the growth of parameter M, the slight decrease in effective key length is noticeable. This fact is caused by the extrema values effect described in [15].

4 Conclusion

In this study, the impact of different TPM parameters on security features of mutual learning key agreement protocol has been presented. In the research, re-

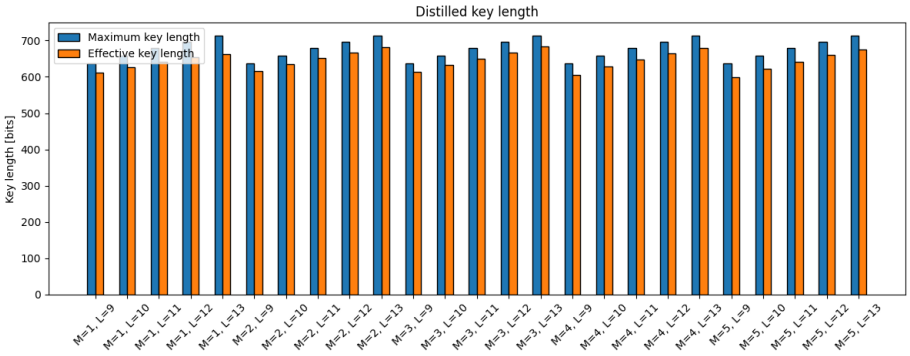


Fig. 3. Maximum and effective key length

quired number of algorithm iterations, average eavesdropping party TPM similarity and effective key length were considered.

The studies have shown that cautious selection of parameters is significant while designing cryptosystems using TPMs. Additionally, the parameter M introduced as the part of NBTPM creates another layer of adjustment. With the increase of this parameter, the synchronization time is reduced significantly, however, the security features also start to degrade slightly. It is worth noting this parameter cannot increase infinitely since the trend of the average evil party synchronization score has a growing trend. This means at some point eavesdropping parties would be able to reconstruct the cryptographic key with ease resulting in a compromise of the whole cryptosystem.

As quantum computing advances, more interest may arise in the use of TPMs in the subject of cybersecurity. Hence the choice between fast synchronization times and higher security of the key agreement must be performed with perfect care.

Acknowledgements

This work was supported by the ECHO project which has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement no. 830943.

References

1. Ahmed M. Allam and Hazem M. Abbas. On the improvement of neural cryptography using erroneous transmitted information with error prediction. *IEEE Transactions on Neural Networks*, 21(12):1915–1924, 2010.
2. Whitfield. Diffie and Martin. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

3. Tao Dong and Tingwen Huang. Neural cryptography based on complex-valued neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4999–5004, 2020.
4. Sooyong Jeong, Cheolhee Park, Dowon Hong, Changho Seo, and Namsu Jho. Neural cryptography based on generalized tree parity machine for real-life systems. *Security and Communication Networks*, 2021:6680782, Feb 2021.
5. Ido Kanter, Wolfgang Kinzel, and Ido Kanter. Secure exchange of information by synchronization of neural networks. *Europhysics Letters (EPL)*, 57(1):141–147, jan 2002.
6. Wolfgang Kinzel and Ido Kanter. *Interacting Neural Networks and Cryptography*, pages 383–391. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
7. Wolfgang Kinzel, Ralf Metzler, and Ido Kanter. Dynamics of interacting neural networks, 1999.
8. Alexander Klimov, Anton Mityagin, and Adi Shamir. Analysis of neural cryptography. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, pages 288–298, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
9. Marcin Niemiec. Error correction in quantum cryptography based on artificial neural networks. *Quantum Information Processing*, 18(6):174, Apr 2019.
10. Ron. L. Rivest, Adi. Shamir, and Leonard. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.
11. Michal Rosen-Zvi, Einat Klein, Ido Kanter, and Wolfgang Kinzel. Mutual learning in a tree parity machine and its application to cryptography. 2002.
12. Andreas Ruttor. *Neural Synchronization and Cryptography*. PhD thesis, University of Würzburg, 2007.
13. S. Santhanalakshmi, T. S. B. Sudarshan, and Gopal K. Patra. Neural synchronization by mutual learning using genetic approach for secure key generation. In Sabu M. Thampi, Albert Y. Zomaya, Thorsten Strufe, Jose M. Alcaraz Calero, and Tony Thomas, editors, *Recent Trends in Computer Networks and Distributed Systems Security*, pages 422–431, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
14. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
15. Miłosz Stypiński and Marcin Niemiec. Synchronization of tree parity machines using nonbinary input vectors. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–7, 2022.

Computer Networks 257 (2025) 110988

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Weight equalization algorithm for tree parity machines☆

Miłosz Stypiński *, Marcin Niemiec

Institute of Telecommunications, AGH University of Krakow, Mickiewicza 30, 30-059, Krakow, Poland

ARTICLE INFO

Keywords:
Mutual learning
Key agreement
Security
Artificial neural networks

ABSTRACT

Key agreement plays a crucial role in ensuring secure communication in public networks. Although algorithms developed many years ago are still being used, the emergence of quantum computing has prompted the search for new solutions. Tree parity machines have been put forward as a potential solution. However, they possess inherent shortcomings, one of which is the uneven distribution of values in the secured key obtained after the key agreement process, especially when non-binary vectors are used during the synchronization process. This paper introduces a novel algorithm designed to address this issue. The results demonstrate a substantial enhancement in the quality of the secured key obtained.

1. Introduction

Secure key agreement refers to a category of protocols in which two or more participants employ established algorithms to mutually generate a cryptographic key. Despite sharing all the necessary information for key agreement over a publicly insecure channel, potential attackers are incapable of deducing any information about the key.

The Diffie–Hellman algorithm [1] and its elliptic curve variant [2] are among the most widely used key exchange algorithms. Both of them are backed by algebraic number theory problems, which are supposed to be difficult to revert. However, successful implementation of Shor's algorithm of sufficiently powerful quantum computers would be capable of breaking the security of widely used key exchanged algorithms [3]. This issue has become sufficient to create a list of three NIST-approved quantum-proof algorithms [4–6]. The proposed new algorithms are based on problems of finding short vectors in high-dimensional lattices and difficulty of inverting hash functions. These post-quantum algorithms has proven resistant against both quantum and classical attacks. The algorithms working names of the first two standards are CRYSTALS-Kyber and CRYSTAL-Dilithium which are responsible for key encapsulation and digital signature respectively. Both are based on module lattices which were inspired by learning with errors problems [7]. The latter standard covers the algorithm named SPHINCS$^+$, which utilizes hash functions. Hash functions are inherently one-way, meaning they are difficult to invert using both classical and quantum algorithms. This characteristic makes them particularly attractive for use in quantum-resistant cryptography.

One of the alternative quantum proof algorithms for key distribution and agreement is mutual learning using specific kind of neural networks – tree parity machine (TPM). TPMs have been extensively studied [8–19]. In [8–11] the authors introduced an innovative key agreement protocol employing artificial neural networks (ANNs). Through mutual learning and learning rule definition they have achieved synchronization of TPMs. Additionally, the authors have demonstrated that TPM synchronization finishes in finite time. The systematic presentation of this work can be found in [12].

Several enhancements have been suggested for TPMs, including Dong et al.'s proposal to use complex values instead of binary values during the learning process [13]. This idea was further generalized in [14], where the authors introduced the use of binary vector values as inputs. In [15] the authors have proposed a usage of integer inputs instead of binary. These improvements are named, accordingly, the complex-valued Tree Parity Machine (CVTPM), vector-valued tree parity machine (VVTPM), and non-binary tree parity machine (NBTPM). This paper focuses on exploring the latter.

TPMs have applications in numerous fields. Sarkar et al. have proposed the usage of neural network synchronization in wireless systems [16]. In [17] TPMs are used as key reconciliation mechanism in quantum key distribution networks. Another application is described in [18] where TPM was responsible for key establishment on microprocessors. Mutual learning, as described in [19], could also find applications in the field of smart grids.

The aim of this article is to propose and evaluate a novel algorithm for TPM weight equalization. The problem is significant in networking environment as we are currently in the search of alternatives to key agreement protocols base on factorization problem. The use of

☆ Research project partially supported by the program "Excellence initiative – research university" for the University of Krakow.
* Corresponding author.
E-mail address: stypinski@agh.edu.pl (M. Stypiński).

https://doi.org/10.1016/j.comnet.2024.110988
Received 30 January 2024; Received in revised form 16 November 2024; Accepted 9 December 2024
Available online 16 December 2024

NBTPMs and proposed algorithms significantly improves the communication security features and renders TPMs as an alternative to currently used key agreement protocols. Furthermore, NBTPMs result in smaller key agreement protocol overhead compared to standard TPM because lesser number of mutual learning iterations are required to successfully generate secure key.

The purpose of this article is to present and assess a novel algorithm for weight equalization in TPM. The problem is significant in networking environments, particularly as there is a need for alternatives to key agreement protocols based on factorization problems. Incorporating NBTPMs and the proposed algorithms brings notable enhancements in communication security, positioning TPM as a viable substitute for current key agreement protocols. Moreover, NBTPM leads to a reduced key agreement protocol overhead compared to standard TPM, as it requires a lesser number of mutual learning iterations for the successful generation of a secure key.

2. Tree parity machine

The original TPM is a two-layered, binary input, binary output ANN [8–11]. The hidden layer consists of K neurons with N inputs per neuron. With every input, there is a weight associated with it that is an integer and is constrained within the range from $-L$ to L and where L is a natural number greater than 0. Similarly, in the feed-forward neural network, in TPM all the outputs of the layer are connected to the neurons of the succeeding layer. NBTPM alters the values that the input vectors can take and allows them to take values from the range of $-M$ to M, where M is a natural number greater than 0 [15]. It is worth noting that NBTPM is a generalization of the typical TPM, and for $M = 1$ there is no difference between these two. To describe a specific NBTPM one needs the quadruple of parameters K, L, M, N which, respectively, denote the number of hidden neurons, range of weight values, variability of input vectors and number of inputs per neuron in hidden layer. Due to the requirement of integer weights in the described model, backpropagation is not applicable in TPMs. Therefore, other learning methods are needed, which will be described later in this section.

To maintain the binary nature of TPMs the activation function of neurons in the hidden layer is a function returning either -1 or 1. Specifically, this is an altered signum function (f_{signum}) that does not produce a 0 result. Instead, it returns -1 or 1, depending on whether it represents the communication of the sender or the recipient. The formula to calculate the output of neurons is presented in (1), where y_k denotes the output of the kth neuron, x_{kn} denotes the input of the nth neuron, and w_{kn} denotes the weight associated with the input.

$$y_k = f_{signum}(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}) \tag{1}$$

TPM output is defined by $O = \prod_k^K y_k$ where y_k denotes the output from the kth hidden neuron. Since y_k gives a binary value, the final output is also binary. The described TPM architecture is presented in Fig. 1.

Mutual learning, in the context of TPM, refers to a specialized key agreement protocol. It involves two neural networks that execute a predefined algorithm to establish synchronization between them. Synchronization is an iterative process and results in both neural networks having exactly the same weights, which can be used as a shared secret in further cryptographic operations. The learning algorithm is responsible for updating the weights. During each successful iteration step, the weights are updated in a manner that draws both TPMs closer to each other. Key agreement protocols assume that there is a public channel which may be eavesdropped on. The steps of mutual learning are summarized below [12].

1. Key agreement parties agree on an exact value for TPM parameters (K, L, M, N) and initialize their own copy of TPM with randomly chosen weights.
2. One of the participants generates a random input vector X the values of which are bound by M and shares it with the other.
3. Both parties calculate the output of their TPM and share it publicly. In case the outputs match, they employ one of the following learning rules:

   - *Hebbian learning rule*

   $$w_{kn}(t+1) = w_{kn}(t) + O(t)x_{kn}(t)\Theta(y_k(t), O(t)), \tag{2}$$

   - *Anti-Hebbian learning rule*

   $$w_{kn}(t+1) = w_{kn}(t) - O(t)x_{kn}(t)\Theta(y_k(t), O(t)), \tag{3}$$

   - *random walk learning rule*

   $$w_{kn}(t+1) = w_{kn}(t) + x_{kn}(t)\Theta(y_k(t), O(t)), \tag{4}$$

   where t denotes the t-th iteration step and Θ is a function with returns 1 if all its arguments are equal.

4. Steps 2–3 are repeated until full synchronization is achieved.

Once mutual learning is completed, both TPMs are synchronized and have their corresponding weights equal. Both parties are able to use the distilled weights as their shared secret for further cryptographic purposes. The secret length is variable and depends on the size of the TPM. Assuming an equal distribution of values in the weight vector W the secret length would be equal to $K \cdot N \cdot log_2(2L+1)$. However, studies show that the distribution differs from the uniform distribution [12,15], and therefore the formula needs to be updated to take this imperfection into account. In [15] the authors defined a formula that meets these conditions and presented it in (5), where K and N are the parameters of TPM and $E(W)$ is an entropy of the weight vector W.

$$len_{secret} = K \cdot N \cdot E(W) = K \cdot N \cdot (-\sum_{l=-L}^{L} p_l log_2 p_l) \tag{5}$$

3. Weight equalization

NBTPM shortens the mutual learning process that results in improved security features of the key agreement protocol. The cost of this improvement is a more uneven distribution of the weight vector values. The values w_{kn}, where $|w_{kn}| = L$ occur more frequently than other values. This phenomenon is known as the Extrema Values Effect and is described in detail in [15]. A similar phenomenon occurs in standard TPM, albeit with a less pronounced effect. This paper proposes a novel algorithm in terms of TPM, later called weight equalization, that equalizes the probability of the occurrence of frequent values in the weight vector. The algorithm is inspired by histogram equalization [20].

The weight equalization algorithm comprises the equalization, dropout, and substitution phase. The equalization phase is responsible for replacing more commonly occurring values with those that occur less frequently. To execute this algorithm, a weight vector W is required that has been derived from synchronized TPMs. Additionally, the algorithm takes a set of parameters (K, L, N) that define the TPM size as the input. During the equalization phase, the algorithm goes through each element of the weight vector W. If, during current iteration, the current weight w_{kn} is the most frequently occurring one, it is exchanged with the least frequent value encountered so far in the vector. To identify the least frequent value among the already processed values, the algorithm keeps a count of how many times each value has appeared, using a vector called F as a cache. This weight equalization process is outlined in Algorithm 1. It is important to note that in this algorithm, the functions argmax and argmin return sets of indices where the maximum and minimum values occur, respectively. Furthermore, the algorithm updates the values in place, so the weight vector must be mutable.

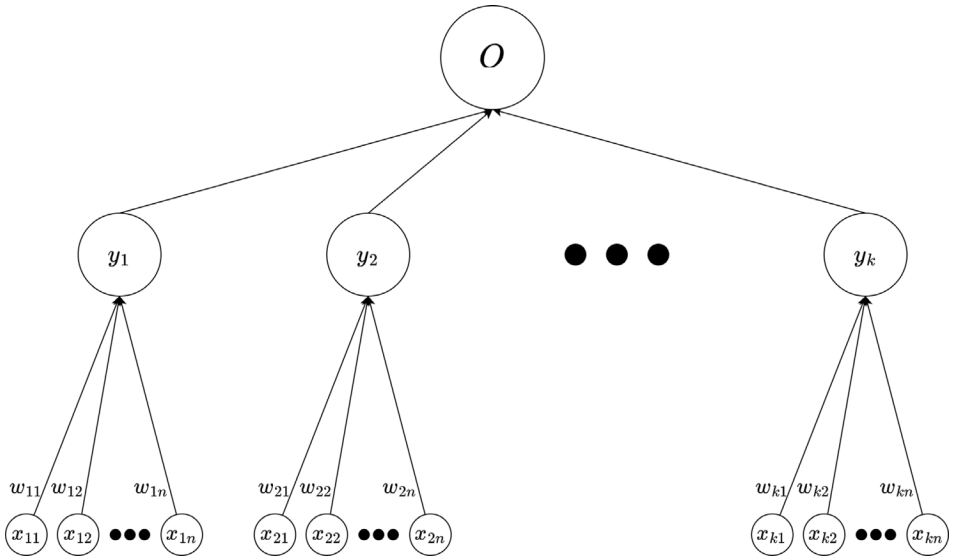


Fig. 1. Architecture of the tree parity machine.

Algorithm 1 Equalization phase

Input: $W \leftarrow [w_{11}, \cdots, w_{1n}, w_{21}, \cdots, w_{kn}]$, K, N, L
Output: W
  $F \leftarrow [f_{-L}, \cdots, f_L] = [0, \cdots, 0]$
  $k \leftarrow 1$
  while $k \leq K$ **do**
    $n \leftarrow 1$
    while $n \leq N$ **do**
      if $W_{kn} \in argmax\ F$ **then**
        $w_{kn} \leftarrow min(argmin\ F)$
      end if
      $F_{w_{kn}} \leftarrow F_{w_{kn}} + 1$
      $n \leftarrow n + 1$
    end while
    $k \leftarrow k + 1$
  end while

The dropout phase is responsible for improving the quality of the distilled key. Once the equalization is complete, the key length is longer than the limit created by Eq. (5). To address this issue the dropout phase was incorporated into the algorithm. Algorithm 2 keeps track of the current secret length, and if the weight exceeds the theoretical limit at some point, it is dropped instead.

Algorithm 2 Dropout phase

Input: K, N, L, W before equalization
Output: W'
  $E(W) = -\sum_{l=-L}^{L} p_l log_2 p_l$
  $len_{current} \leftarrow 0$
  W', which is a variable length vector with elements
  $k \leftarrow 1$
  while $k \leq K$ **do**
    $n \leftarrow 1$
    while $n \leq N$ **do**
      if $len_{current} < ((k-1) \cdot N + n) \cdot E(W)$ **then**
        $W'.append(w_{kn})$
        $len_{current} \leftarrow len_{current} + log_2(2L+1)$
      end if
      $n \leftarrow n + 1$
    end while
    $k \leftarrow k + 1$
  end while

The final stage of the algorithm involves substitution. The objective in this phase is to further enhance the randomness of the secret and propagate single errors to affect the entire secret. The most suitable tools for achieving this goal are cryptographic hash functions, which, thanks to the avalanche effect, exhibit substantial changes in their output even when there are minor variations in the input. Furthermore, these functions are difficult to reverse, offering an additional advantage. During this step, the secret is divided into blocks, each with a length equal to that of the hash function's output. Subsequently, each block is replaced with the output of the hash function for that specific block. If the last block happens to be shorter than the hash function's output, it is simply omitted.

Both of the mutual learning participants execute the presented algorithm to improve their secret key quality. The algorithm is deterministic, hence both parties will always obtain the same secret once the algorithm is executed.

4. Evaluation

The algorithm's performance assessment involves two main aspects. First, it involves comparing the probabilities of attaining specific weights in the weight vector both before and after executing the algorithm. Second, the evaluation also includes a comparison of the results of the NIST Test Suite [21] for the weight vector after representing it in binary form, both before and after the application of the weight equalization algorithm. To obtain the weight vectors, mutual learning has been performed using NBTPM with following parameters: $K = 3$, $L = 8$, $M = \{1; 3; 5\}$, and $N = 60$. Mutual learning was performed using the Hebbian learning rule. The TPMs were synchronized 1000 times using a dedicated simulation framework.[1] The selected hash function for the substitution phase was SHA-256.

NIST Special Publication 800–22 defines a set of tests that aims to assess the quality of random and pseudo-random number generators for cryptographic applications [21]. The test suite assesses a wide range of properties of random binary string, such as frequency, runs, correlation and patterns. Therefore, using this test set we are able to objectively assess whether a binary sequence is random — which is crucial for the security of symmetric keys. Furthermore, tests' descriptions include details implementation guide, examples and suggested test parameters. The following is the list of tests with a brief description and selected

[1] The source code is available at https://github.com/mstypinski/wea.

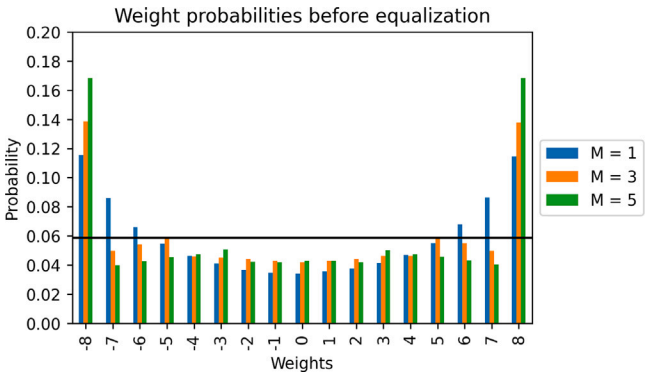


Fig. 2. Weight probabilities before equalization.

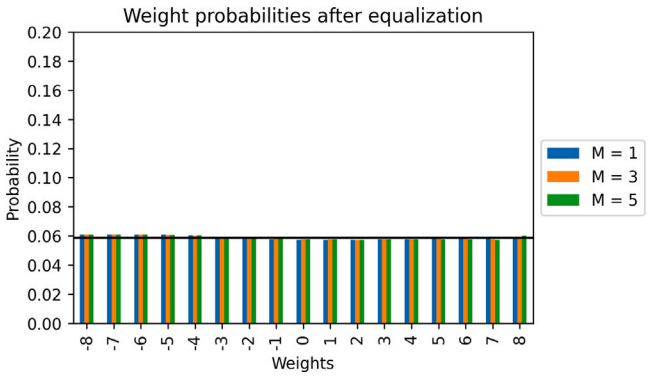


Fig. 3. Weight probabilities after equalization.

parameters. If the test parameters are not provided, the default values defined in [21] are used.

1. *Frequency (Monobit) Test* – aims to check the ratio of zeroes and ones in a random sequence.
2. *Frequency Test within a Block* – evaluates proportions of ones in M-bit block. The block size used in this paper is equal to 128 bits.
3. *Runs Test* – analyzes whether the oscillation between ones and zeros stays within the boundary for random sequence.
4. *Test for the Longest Run of Ones in a Block* – examines the same bit sequence in an M-length block.
5. *Binary Matrix Rank Test* – checks the linear dependence between the subsequences of the original random values.
6. *Discrete Fourier Transform (Spectral) Test* – looks for anomalies in the Discrete Fourier Transform of the random sequence
7. *Non-overlapping Template Matching Test* – checks the frequency of occurrences of the M-bit sequence defined beforehand.
8. *Overlapping Template Matching Test* – the aim of this test is the same as the previous one; however, if it finds a sequence it only moves the search window only by one bit instead of M bits.
9. *Maurer's Test* – evaluates whether the examined sequence is compressible using a lossless algorithm.
10. *Linear Complexity Test* – evaluates the length of a linear feedback shift register. The block used for this test is equal to 500 bits.
11. *Serial Test* – tests the frequency of all M-bit permutations in the examined sequence.
12. *Approximate Entropy Test* – evaluates the occurrence rate of every conceivable M-bit pattern.
13. *Cumulative Sums (Cusum) Test* – assesses whether the cumulative sums of a random binary string are within an acceptable range for a particular random sequence. This assessment is carried out both forwards and backwards.
14. *Random Excursions Test* – analyses how many times the specific state has been visited during the cumulative sums random walk performed on the sequence. Before running the test, 0 in a random value is swapped for the value −1. The results are presented for states ranging from −4 to 4 omitting 0.
15. *Random Excursions Variant Test* – test methodology is the same as the previous one; however, the states consider the range from −9 to 9 without 0.

Firstly, the probability of the presence of a specific value in the weight vector is examined. The probabilities of weight occurrence are not equal. With the increase of M the distribution becomes more dominated by limit values. While for $M = 1$ the probability of $\omega_{kn} \in \{-L; L\}$ is equal to 0.23, for $M = 3$ and $M = 5$ the same probability is equal to 0.28 and 0.34, respectively. This is an example of the Extrema Values Effect phenomenon. The probabilities for all weights before running the equalization algorithm are detailed in Fig. 2. The black horizontal line depicts the ideal uniform distribution $y = \frac{1}{2L+1}$

Following equalization, the probabilities follow an almost uniform distribution. The greatest difference from ideal uniform distribution is 0.0023, which is equal to 3.89% of $\frac{1}{2L+1}$. For comparison, the weights following equalization are presented in Fig. 3.

Another evaluation method consists of testing the weight vector against the NIST testing suite. The tests return on or more P-values which are the determinants of the probability that the tested sequence is fully random. If the P-value is equal to 1 then the tested sequence is fully random. On the other hand, if the P-value is equal to 0, then the tested sequence is generated in a deterministic manner. The test is successful when the returned P-value is greater than 0.01. Tests have been run on the weight vector obtained from the aforementioned simulations. The weights have been encoded using little-endian encoding. Before applying the equalization algorithm, the values of 0 have been removed from the weight vector to facilitate encoding.

The tests demonstrate that direct usage of TPM and NBTPM produces a secret key that cannot be used for security purposes. Only two tests passed for all TPM variants before using the equalization algorithm: the *Binary Matrix Rank Test* and *Linear Complexity Test*. On the other hand, the weight equalization algorithm significantly improves the overall results of the test suite on the obtained key. The only tests with produce allegedly poorer results for specific scenarios are the *Random Excursions Test* and *Random Excursions Test Variant* for some states. However, the sequence produced is pseudo-random and could have been drawn in such a way that gave poorer results in this case. Moreover, the result of the *Approximate Entropy Test* for $M = 5$ was close to a failure. This conclusion follows studies conducted in [15] where the authors stated that the values of M approaching the values of L results in a less secure NBTPM variant.

The results obtained from the simulations are presented in Table 1. Abbreviations *b. e.* and *a. e.* in the table header mean the results before and after equalization, respectively. For the *Random Excursions Test* and *Random Excursions Variant Test* results are sorted by states from −4 to 4 and from −9 to 9 respectively.

5. Summary

This paper introduces a novel algorithm designed for weight equalization in TPMs. The algorithm consists of three phases: equalization, dropout, and substitution. Its utilization addresses the issue of uneven weight distribution that may arise after synchronization of TPMs. Moreover, it effectively mitigates the disadvantages introduced by the use of NBTPM, the Extrema Values effect in particular. The paper also investigates the impact of the algorithm on the resulting security key. It examines the probability of specific weight values appearing in the weight vector before and after applying the algorithm. Additionally, the

Table 1
NIST Test suite results.

	P-values					
Test	$M = 1$		$M = 3$		$M = 5$	
	b. e.	a. e.	b. e.	a. e.	b. e.	a. e.
(1)	0.0	0.961	0.0	0.665	0.0	0.397
(2)	0.0	0.66	0.0	0.011	0.0	0.734
(3)	0.0	0.063	0.0	0.686	0.0	0.155
(4)	0.0	0.726	0.0	0.073	0.0	0.394
(5)	0.461	0.587	0.791	0.953	0.104	0.491
(6)	0.0	0.171	0.0	0.623	0.0	0.517
(7)	0.0	0.052	0.0	0.369	0.0	0.668
(8)	0.0	0.443	0.0	0.268	0.0	0.462
(9)	0.0	0.518	0.0	0.135	0.0	0.903
(10)	0.126	0.813	0.294	0.337	0.535	1.0
(11)	(0.0, 0.711)	(0.522, 0.788)	(0.0, 0.028)	(0.717, 0.452)	(0.0, 0.0)	(0.713, 0.667)
(12)	0.0	0.3	0.0	0.477	0.0	0.025
(13) (Fwd)	0.0	0.676	0.0	0.812	0.0	0.746
(13) (Bwd)	0.0	0.63	0.0	0.758	0.0	0.583
(14)	0.845, 0.278, 0.89, 0.915, 0.626, 0.24, 0.082, 0.25	0.982, 0.402, 0.113, 0.56, 0.685, 0.933, 0.721, 0.109	0.014, 0.0, 0.341, 0.021, 0.514, 0.883, 0.211, 0.053	0.418, 0.351, 0.631, 0.638, 0.93, 0.642, 0.423, 0.468	0.926, 0.932, 0.232, 0.579, 0.326, 0.877, 0.046, 0.021	0.394, 0.216, 0.085, 0.396, 0.028, 0.346, 0.805, 0.478
(15)	0.42, 0.374, 0.357, 0.414, 0.512, 0.71, 0.783, 0.722, 0.538, 0.806, 0.67, 0.545, 0.816, 0.712, 0.436, 0.357, 0.325, 0.339	0.088, 0.104, 0.207, 0.411, 0.537, 0.564, 0.331, 0.113, 0.108, 0.745, 0.63, 0.949, 0.914, 0.65, 0.569, 0.437, 0.358, 0.481	0.903, 0.949, 1.0, 1.0, 0.803, 0.637, 1.0, 0.386, 0.317, 0.617, 0.773, 0.655, 0.705, 0.677, 0.546, 0.579, 0.606, 0.628	1.0, 0.974, 0.967, 0.821, 0.588, 0.755, 0.56, 0.236, 0.293, 0.453, 0.174, 0.268, 0.684, 0.707, 0.734, 0.934, 0.821, 0.552	0.878, 0.87, 0.861, 1.0, 1.0, 0.811, 0.671, 0.855, 0.527, 0.752, 0.715, 0.888, 0.72, 0.673, 0.634, 0.661, 0.683, 0.701	0.53, 0.33, 0.258, 0.228, 0.195, 0.259, 0.224, 0.142, 0.141, 0.948, 0.175, 0.103, 0.3, 0.539, 0.401, 0.265, 0.308, 0.287

secret key generated is subjected to testing using the NIST randomness test suite. Future research should focus on examining the quality of the key obtained after using the proposed algorithm, for instance, through additional randomness testing with alternative randomness test suites, and determining whether its use facilitates potential attacks on the mutual learning process. Furthermore, additional research should investigate the feasibility of using NBTPMs in real network environments, such as wired and wireless networks, and low-energy IoT devices..

TPMs are neural networks that have applications in key synchronization and key reconciliation processes. By utilizing improvements, such as the proposed algorithm, TPMs becomes more secure and more competitive against other algorithms.

CRediT authorship contribution statement

Miłosz Stypiński: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Marcin Niemiec:** Writing – review & editing, Validation, Supervision, Methodology.

Funding

The research has been partially supported by the program "Excellence initiative – research university" for the AGH University of Krakow.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory 22 (6) (1976) 644–654.
[2] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, Tech. Rep., National Institute of Standards and Technology, 2018.
[3] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (5) (1997).
[4] National Institute of Standards and Technology, Module-Lattice-Based Key-Encapsulation Mechanism Standard, Tech. Rep. 203, Federal Information Processing Standards Publication, 2024.
[5] National Institute of Standards and Technology, Module-Lattice-Based Digital Signature Standard, Tech. Rep. 204, Federal Information Processing Standards Publication, 2024.
[6] National Institute of Standards and Technology, Stateless Hash-Based Digital Signature Standard, Tech. Rep. 205, Federal Information Processing Standards Publication, 2024.
[7] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, J. ACM 56 (6) (2009) 1–40.
[8] R. Metzler, W. Kinzel, I. Kanter, Interacting neural networks, Phys. Rev. E 62 (2 Pt B) (2000) 2555–2565.
[9] W. Kinzel, R. Metzler, I. Kanter, Dynamics of interacting neural networks, J. Phys. A: Math. Gen. 33 (14) (2000) L141.
[10] I. Kanter, W. Kinzel, E. Kanter, Secure exchange of information by synchronization of neural networks, Europhys. Lett. 57 (1) (2002) 141.
[11] A. Ruttor, G. Reents, W. Kinzel, Synchronization of random walks with reflecting boundaries, J. Phys. A: Math. Gen. 37 (36) (2004) 8609.
[12] A. Ruttor, Neural Synchronization and Cryptography (Ph.D. thesis), University of Würzburg, 2007.
[13] T. Dong, T. Huang, Neural cryptography based on complex-valued neural network, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2020) 4999–5004.
[14] S. Jeong, C. Park, D. Hong, C. Seo, N. Jho, F. Lombardi, Neural cryptography based on generalized tree parity machine for real-life systems, Sec. Commun. Netw. 2021 (2021).
[15] M. Stypiński, M. Niemiec, Synchronization of tree parity machines using nonbinary input vectors, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–7.
[16] A. Sarkar, J. Dey, A. Bhowmik, Multilayer neural network synchronized secured session key based encryption in wireless communication, Indonesian J. Electr. Eng. Comput. Sci. 14 (1) (2019) 169.
[17] M. Niemiec, Error correction in quantum cryptography based on artificial neural networks, Quantum Inf. Process. 18 (6) (2019).
[18] H. Gomez, Ó. Reyes, E. Roa, A 65nm CMOS key establishment core based on tree parity machines, Integration 58 (2017) 430–437.
[19] M. Stypiński, M. Niemiec, Security of neural network-based key agreement protocol for smart grids, Energies 16 (10) (2023).

[20] S.M. Pizer, E.P. Amburn, J.D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J.B. Zimmerman, K. Zuiderveld, Adaptive histogram equalization and its variations, Comput. Vis. Graph. Image Process. 39 (3) (1987) 355–368.
[21] L.E. Bassham, A.L. Rukhin, J. Soto, J.R. Nechvatal, M.E. Smid, E.B. Barker, S.D. Leigh, M. Levenson, M. Vangel, D.L. Banks, N.A. Heckert, J.F. Dray, S. Vo, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Tech. Rep., National Institute of Standards and Technology, 2010.

Miłosz Stypinski received his M.Sc. in ICT from AGH University of Krakow, in 2019. Krakow. He is now pursuing a Ph.D. degree in cybersecurity at the same university. His current research interests focus on the application of artificial intelligence algorithms in cybersecurity, in particular symmetric and asymmetric ciphers which use neural networks.

Marcin Niemiec is working as a university professor at the Institute of Telecommunications, AGH University of Krakow. His research interests focus on cybersecurity and data protection, especially security services, symmetric ciphers, network security, intrusion detection, and quantum cryptography. He has actively participated in 6th and 7th FP European programs (ePhoton/ONE+, BONE, SmoothIT, INDECT), Horizon 2020 Framework Programme (SCISSOR, ECHO), Eureka-Celtic (DESYME), and many national research projects. He co-authored over 100 publications and reports.

KONFERENCJA RADIOKOMUNIKACJI
I TELEINFORMATYKI
KRiT 2024

WPŁYW PARAMETRÓW SIECI TYPU TREE PARITY MACHINE NA BEZPIECZEŃSTWO ALGORYTMU WYRÓWNUJĄCEGO WAGI

THE IMPACT OF TREE PARITY MACHINE NETWORK PARAMETERS ON THE SECURITY OF THE WEIGHT EQUALIZATION ALGORITHM

Miłosz Stypiński

AGH Akademia Górniczo-Hutnicza,
Instytut Telekomunikacji,
al. Mickiewicza 30, 30-059 Kraków

Streszczenie: Znaczący postęp w technologi obliczeń z wykorzystaniem komputerów kwantowych jest ogromnym zagrożeniem dla obecnie wykorzystywanej kryptografii asymetrycznej. Sieci neuronowe typu Tree Parity Machine są alternatywnym algorytmem uzgodnienia klucza kryptograficznego. W ostatnim czasie zaproponowano szereg rozwiązań usprawniających właściwości klucz kryptograficznego otrzymanego z wykorzystaniem wspomnianego rozwiązania. Celem tego artykułu jest zbadanie zależności pomiędzy bezpieczeństwem usprawnień sieci Tree Parity Machine, a jej parametrami i rozmiarem.

Abstract: Significant advancements in quantum computing technology pose a major threat to currently used asymmetric cryptography. Tree Parity Machines are an alternative algorithm for cryptographic key agreement. Recently, several solutions have been proposed to enhance the properties of cryptographic keys obtained using this approach. The aim of this article is to investigate the relationship between the security of Tree Parity Machine network improvements and its parameters and size.

Słowa kluczowe: uzgodnienie klucza kryptograficznego, uczenie wzajemne, sztuczne sieci neuronowe, kryptografia postkwantowa

Keywords: key agreement, mutual learning, artificial neural networks, postquantum cryptography

1. WSTĘP

Komputery kwantowe stanowią potencjalną rewolucję w dziedzinie zaawansowanych obliczeń. Dzięki zastosowaniu zasad mechaniki kwantowej, takich jak superpozycja i splątanie kwantowe, komputer kwantowy pozwala na przyspieszenie rozwiązywania problemów należących do klasy Bounded-error Quantum Polynomial Time (BQP) [2].

Aktualnie wykorzystywane algorytmy kryptografii asymetrycznej korzystają z funkcji algebraicznych, które są na dzień dzisiejszy niezwykle trudne do odwrócenia. Niestety algorytm Shora pozwala na rozwiązanie problemów takich jak faktoryzacja dużych liczb, bądź znalezienie logarytmu dyskretnego w ciele skończonym [14]. W wyniku wspomniane problemy również należą do klasy BQP, a w rezultacie komputer kwantowy o odpowiedniej liczbie kubitów będzie w stanie zagrozić bezpieczeństwu wykorzystywanej kryptografii asymetrycznej.

Znalezienie algorytmów kryptografii postkwantowej, czyli odpornych na działanie komputera kwantowego, jest niezwykle ważne dla utrzymania porównywalnego poziomu bezpieczeństwa w telekomunikacji. Znane instytucje standaryzacyjne prowadzą badania nad wspomnianymi algorytmami. Znaczącymi osiągnięciami w tej kwestii jest rozstrzygnięcie konkursu prowadzonego przez National Institute of Standards and Technology (NIST) [1], gdzie wyłoniono trzech finalistów. Ponadto, rozwijane są biblioteki programistyczne implementujące algorytmy odporne na działanie komputera kwantowego. Znaczącym osiągnięciem w tej tematyce jest projekt Open Quantum Safe [15].

Alternatywnym protokołem uzgodnienia klucza kryptograficznego, będącym tematem tej publikacji, jest wzajemna synchronizacja sieci neuronowych typu Tree Parity Machine (TPM). W [5, 7, 10] została po raz pierwszy zaproponowana struktura sieci typu TPM. Ponadto we wspomnianej publikacji zostało udowodnione, że wzajemne uczenie tych sieci neuronowych skutkuje pełną synchronizacją tych sieci, czyli odpowiadające sobie wagi w obu sieciach posiadają dokładnie takie same wartości. Ponadto autorzy . Do sieci neuronowych typu TPM zaproponowane wiele usprawnień. W [11] i [12] zostały przedstawione usprawnienia polegające na wykorzystaniu odpowiednio algorytmów genetycznych oraz inspirowanego naturą algorytmu optymalizacyjnego. Zmiana sposobu uczenia sieci typu TPM jest oddzielną rodziną usprawnień. Zaproponowane usprawnienia z tej rodziny zostały opisane w [3, 4, 17], gdzie autorzy zaproponowali wykorzystanie odpowiednio wartości zespolonych, wektorów oraz liczb niebinarnych jako wartości wektora wejściowego podczas synchronizacji omawianych sieci neuronowych. Ponadto sieci TPM zostały poddane różnym testom bezpieczeństwa [6, 9, 13]. Wśród przeprowadzonych testów nie zdefiniowano żadnego algorytmu, który umożliwiałby przyspieszenie ataku przy użyciu komputera kwantowego.

Sieci neuronowe typu TPM wykorzystujące niebinarne wektory wejściowe są przedmiotem badań tej publikacji (Nonbinary Tree Parity Machine (NBTPM)).

Wykorzystanie niebinarnych wektorów wejściowych skutkuje przyspieszeniem procesu uczenia sieci neuronowych, a w rezultacie skraca czas wymagany do uzgodnienia klucza kryptograficznego. Negatywnym efektem wy-

korzystania niebinarnych wartości wejściowych jest szybsza synchronizacja jednostki wykonującej atak typu man-in-the-middle [16] oraz większa podatność sieci NBTPM na odzyskanie klucza kryptograficznego w wyniku podsłuchiwania kanału komunikacyjnego. Dalsza ewaluacja bezpieczeństwa sieci neuronowych NBTPM została przeprowadzona w [18]. W celu zmniejszenia negatywnych skutków wynikających z zastosowania niebinarnych wektorów wejściowych, w [19] zaproponowano algorytm, który sprowadza otrzymany klucz kryptograficzny do rozkładu jednorodnego. W tej publikacji zostanie przebadany wpływ rozmiaru sieci NBTPM na skuteczność wspomnianego algorytmu.

2. OPIS MODELU

Struktura TPM to dwuwarstwowa sieć neuronowa o binarnej odpowiedzi. W zależności od rozważanego wariantu wartości uczące mogą przybierać różne wartości. W pierwowzorze sieci TPM wykorzystywane są wartości binarne, jednak w tym artykule rozważany jest wariant sieci NBTPM. Dla tego modelu wektor uczący składa się z liczb całkowitych x_{kn} przybierających wartości od $-M$ do M. W pierwszej warstwie sieci neuronowej typu NBTPM znajduje się K neuronów. Każdy z nich posiada N dedykowanych wejść. Z każdym wejściem rozważanego modelu istnieje powiązana waga w_{kn}, która może przyjmować wartości całkowite z przedziału od $-L$ do L. Funkcja aktywacji w każdym neuronie pierwszej warstwy to zmodyfikowana funkcja signum, która zamiast przybierać wartość równą 0 zwraca -1 albo 1. Wartość y_k, będąca odpowiedzią k-tego neuronu, została przedstawiona w równaniu (1). Wszystkie wartości wynikowe neuronów pierwszej warstwy przekazywane są bezpośrednio do drugiej warstwy, będącej jednocześnie ostatnią. Funkcja aktywacji wspomnianego pojedynczego neuronu jest równa iloczynowi wejść, co stanowi ostateczną, binarną odpowiedź sieci. Model sieci TPM został przedstawiony na rysunku 1.

$$y_k = f_{signum}(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}) \tag{1}$$

Proces uczenia, dalej nazywany synchronizacją lub uczeniem wzajemnym, nie jest zgodny ze standardowym algorytmem propagacji wstecznej ze względu na nieciągłość wartości jakie mogą przyjmować wagi modelu sieci NBTPM. W celu wzajemnego nauczenia dwóch sieci, czyli doprowadzeniu ich do posiadania identycznych wag, wykonywany jest dedykowany algorytm, którego skutkiem jest pełna synchronizacja, co jest tożsame z tym, że obie sieci neuronowe są identyczne. Algorytm uczenia jest przedstawiony poniżej [8].

1. Członkowie synchronizacji uzgadniają zestaw parametrów synchronizowanej sieci NBTPM, czyli wartości dla parametrów (K, L, M, N) oraz niezależnie losują wstępne wartości wag z przedziału liczb całkowitych od $-L$ do L.
2. Losowy uczestnik generuje wektor losowy X i udostępnia go przez publiczny kanał drugiemu uczestnikowi synchronizacji. Wektor składa się z wartości x_{kn}, gdzie x_{kn} jest liczbą całkowitą z przedziału od $-M$ do M
3. Uczestnicy obliczają odpowiedź swoich sieci NBTPM i dzielą się nią wzajemnie
4. W przypadku, gdy odpowiedzi sieci są sobie równe uczestnicy stosują wybraną wcześniej metodę uczenia. Poniżej przedstawiono dostępne metody uczenia, gdzie t oznacza numer iteracji algorytmu, a funkcja Θ to funkcja przyjmująca wartość 1, jeżeli wszystkie jej argumenty są sobie równe; w przeciwnym wypadku przyjmuje wartość 0.
   - *Hebbian*
     $$w_{kn}(t+1) = w_{kn}(t)+O(t)x_{kn}(t)\Theta(y_k(t), O(t)),$$
   - *Anti-Hebbian*
     $$w_{kn}(t+1) = w_{kn}(t)-O(t)x_{kn}(t)\Theta(y_k(t), O(t)),$$
   - *błądzenie losowe*
     $$w_{kn}(t+1) = w_{kn}(t) + x_{kn}(t)\Theta(y_k(t), O(t)).$$
5. Kroki 2-4 wykonywane są aż do osiągnięcia pełnej synchronizacji obu sieci neuronowych.

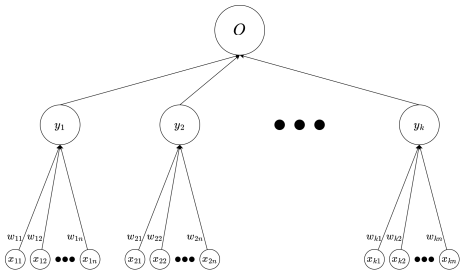
Po wykonaniu algorytmu obie sieci neuronowe są identyczne, czyli odpowiadające sobie wagi posiadają równe sobie wartości. Wektor W stworzony z wag w_{kn} stanowi uzgodniony klucz kryptograficzny, który może następnie zostać wykorzystany w utworzeniu bezpiecznego kanału komunikacyjnego.

Poza pozytywnymi skutkami zastosowania wariantu NBTPM istnieją też skutki negatywne, takiej jak nierównomierny rozkład wartości w wektorze W zwany efektem wartości krańcowych [17]. Jako rozwiązanie problemu tego problemu zaproponowano algorytm, który sprowadza rozkład wag do rozkładu jednostajnego [19]. Działanie algorytmu składa się z trzech etapów przedstawionych poniżej.

- *Wyrównywanie wag* – w tej fazie algorytm analizuje wektor wag W element po elemencie, jednocześnie mierząc częstotliwość występowania konkretnej wartości. Jeżeli podczas danej iteracji rozważana wartość jest najczęściej występującą to jest podmieniana na wartość dotychczas występującą najrzadziej;
- *Odrzucenie wag* – w wyniku nierównomiernego występowania wag, entropia każdego elementu jest mniejsza od entropii hipotetycznego rozkładu jednostajnego. Aby pozbyć się informacji nadmiarowych, pojedyncze elementy wektora W są odrzucane w sposób deterministyczny;
- *Podmiana wartości* – w celu poprawienia losowości otrzymanego wektora, zostaje on zamieniony na wektor binarny, a następnie podzielony na bloki równej długości. Następnie tak przygotowane bloki są podmieniane przez wartość zwracaną przez wcześniej wybraną funkcje haszującą. Dodatkowo, wybór funkcji haszującej implikuje długość bloku na jaką jest dzielona binarna forma wektora W.

Dokładne działanie algorytmu zostało przedstawione w [19].

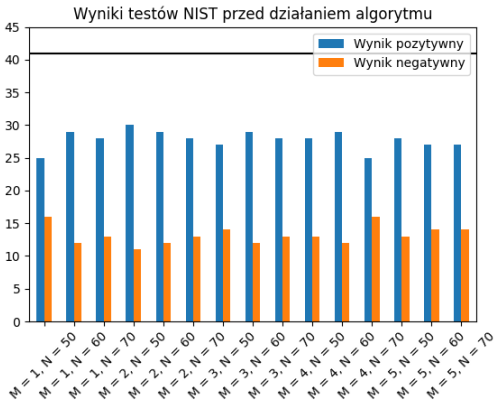
W dalszej części publikacji przedstawione zostaną pogłębione badania nad opisany algorytmem. Celem badań będzie sprawdzenie zależności pomiędzy parametrami (rozmiarem) sieci neuronowej typu NBTPM, a jakością uzyskanego klucza kryptograficznego.



Rysunek 1: Architektura modelu sieci typu Tree Parity Machine

3. WYNIKI

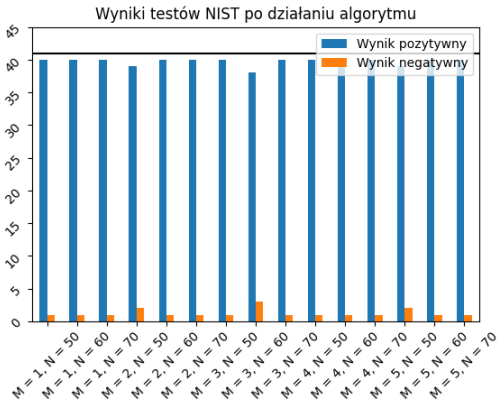
W celu analizy zależności pomiędzy rozmiarem sieci NBTPM, a jakością otrzymanego klucza kryptograficznego przeprowadzone zostały badania symulacyjne. W trackie symulacji dwie sieci NBTPM synchronizowały się ze sobą czego efektem było uzyskanie klucza kryptograficznego. Tak utworzony klucz kryptograficzny był następnie przetwarzany przez algorytm sprowadzający wagi do rozkładu jednostajnego. Następnie obie wersje otrzymanego klucza kryptograficznego zostały poddawane testom opisanym w standardzie NIST 800-22 [1], których celem jest badanie jakości generowanych liczb losowych do zastosowań kryptograficznych. Zestaw testów NIST 800-22 składa się z 15 różnych testów, z których niektóre są parametryzowane, co łącznie daje 41 wyników. Wykorzystywane testy badają różne właściwości rozważanego ciągu losowego, m. in.: częstotliwość występowania wartości 0 i 1 w całym badanym ciągu binarnym jak i jego podciągach, widmo badanego klucza kryptograficznego, częstotliwość występowania wzorców oraz entropię. Podczas symulacji rozważano następujące zestawy parametrów (rozmiary) sieci NBTPM $K = 3, L = 8, M = 1; 2; 3; 4; 5, N = 50, 60, 70$, a każdy scenariusz został powtórzony 100 razy w celu otrzymania dłuższego klucza kryptograficznego do badań.



Rysunek 2: Wyniki testów przed wykorzystaniem algorytmu

Przed zastosowaniem algorytmu równoważącego wartości w otrzymanym wektorze wag zestaw testów NIST 800-22 oscyluje od 11 do 16 z 41 testów zakończonych niepowodzeniem. Następujący fakt jest spowodowany nierównomiernym rozkładem wag, gdzie wartości graniczne lub bliskie zeru występują częściej w zależności od wykorzystanej reguł uczącej [8]. Wyniki przed wykonaniem algorytmu rozważanego w tym artykułu przedstawione są na rysunku 2.

Po zastosowaniu algorytmu wyrównującego wagi liczba testów zakończonych niepowodzeniem znacząco spada, bez znaczących zależności pomiędzy parametrami sieci, a liczbą niepowodzeń. Liczba testów zakończonych niepowodzeniem mieści się w przedziale od 1 do 3, gdzie maksimum ma miejsce dla $M = 3, N = 60$. Pojedyncze testy nie dające wyniku pozytywnego spowodowane są probabilistyczną naturą testów. Stosunek testów zakończonych powodzeniem do testów zakończonych wynikiem negatywnym został przestawiony na rysunku 3.



Rysunek 3: Wyniki testów po wykorzystaniu algorytmu

4. PODSUMOWANIE

W niniejszym artykule przedstawiono postęp prac nad rozwojem kryptografii z wykorzystaniem sieci neuronowych. Zaprezentowana została zasada działania sieci TPM, zaproponowane usprawnienie w postaci wariantu NBTPM oraz korzyści i ryzyka płynące z jego zastosowania. Ponadto omówiono algorytm, którego celem jest równoważenie rozkładu wag i zostały przeprowadzone dalsze badania, które weryfikowały działanie algorytmu dla różnych wariantów sieci NBTPM.

Wykonane badania wykazały, że zastosowanie algorytmu sprowadzającego rozkład otrzymanych wag do rozkładu jednostajnego pozytywnie wpływa na jakość klucza kryptograficznego otrzymanego w skutek uczenia wzajemnego. Ponadto działanie algorytmu jest niezależne od parametrów wykorzystywanej sieci NBTPM.

Kryptografia wykorzystująca sieci neuronowe umożliwia bezpieczne uzgodnienie klucza kryptograficznego, odpornego na ataki komputerów kwantowych. Jednak w celu osiągnięcia wymaganego bezpieczeństwa kluczowe jest odpowiednie dobranie parametrów sieci oraz zapobieganie wpływowi efektu wartości krańcowych. Konieczne są także dalsze badania, aby zweryfikować tę metodę w środowiskach podatnych na ataki kryptograficzne.

LITERATURA

[1] Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., Leigh, S. D., Levenson, M., Vangel, M., Banks, D. L., Heckert, N. A., Dray, J. F., and Vo, S. Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., Gaithersburg, MD, USA, 2010.

[2] Cao, Y., Zhao, Y., Wang, Q., Zhang, J., Ng, S. X., and Hanzo, L. The evolution of quantum key distribution networks: On the road to the qinternet. *IEEE Communications Surveys Tutorials 24*, 2 (2022), 839–894.

[3] Dong, T., and Huang, T. Neural cryptography based on complex-valued neural network. *IEEE Transactions on Neural Networks and Learning Systems 31*, 11 (2020), 4999–5004.

[4] Jeong, S., Park, C., Hong, D., Seo, C., and Jho, N. Neural cryptography based on generalized tree parity machine for real-life systems. *Security and Communication Networks 2021* (Feb 2021), 6680782.

[5] Kinzel, W., Metzler, R., and Kanter, I. Dynamics of interacting neural networks. *Journal of Physics A: Mathematical and General 33*, 14 (apr 2000), L141.

[6] Klimov, A., Mityagin, A., and Shamir, A. Analysis of neural cryptography. In *Advances in Cryptology — ASIACRYPT 2002* (Berlin, Heidelberg, 2002), Y. Zheng, Ed., Springer Berlin Heidelberg, pp. 288–298.

[7] Metzler, R., Kinzel, W., and Kanter, I. Interacting neural networks. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics 62*, 2 Pt B (2000), 2555–2565.

[8] Ruttor, A. *Neural Synchronization and Cryptography*. PhD thesis, University of Würzburg, 2007.

[9] Ruttor, A., Kinzel, W., Naeh, R., and Kanter, I. Genetic attack on neural cryptography. *Phys. Rev. E 73* (Mar 2006), 036121.

[10] Ruttor, A., Reents, G., and Kinzel, W. Synchronization of random walks with reflecting boundaries. *Journal of Physics A: Mathematical and General 37*, 36 (aug 2004), 8609.

[11] Santhanalakshmi, S., K., S., and Patra, G. K. Analysis of neural synchronization using genetic approach for secure key generation. In *Security in Computing and Communications* (Cham, 2015), J. H. Abawajy, S. Mukherjea, S. M. Thampi, and A. Ruiz-Martínez, Eds., Springer International Publishing, pp. 207–216.

[12] Sarkar, A., Khan, M. Z., Singh, M. M., Noorwali, A., Chakraborty, C., and Pani, S. K. Artificial neural synchronization using nature inspired whale optimization. *IEEE Access 9* (2021), 16435–16447.

[13] Shacham, L. N., Klein, E., Mislovaty, R., Kanter, I., and Kinzel, W. Cooperating attackers in neural cryptography. *Phys. Rev. E 69* (Jun 2004), 066137.

[14] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing 26*, 5 (1997).

[15] Stebila, D., and Mosca, M. Post-quantum key exchange for the internet and the open quantum safe project. In *Selected Areas in Cryptography – SAC 2016* (Cham, 2017), R. Avanzi and H. Heys, Eds., Springer International Publishing, pp. 14–37.

[16] Stypiński, M., and Niemiec, M. Impact of nonbinary input vectors on security of tree parity machine. In *Multimedia Communications, Services and Security* (Cham, 2022), A. Dziech, W. Mees, and M. Niemiec, Eds., Springer International Publishing, pp. 94–103.

[17] Stypiński, M., and Niemiec, M. Synchronization of tree parity machines using nonbinary input vectors. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–7.

[18] Stypiński, M., and Niemiec, M. Security of neural network-based key agreement protocol for smart grids. *Energies 16*, 10 (2023).

[19] Stypiński, M., and Niemiec, M. Weight equalization algorithm for tree parity machines, 2024.

Oświadczenie udziałów procentowych w publikacji

"Quantum-based Solutions for the Next-generation Internet"; Niemiec Marcin, Dziech Andrzej, Stypiński Miłosz, Derkacz Jan; Information & Security: An International Journal 43, 21-33; https://doi.org/10.11610/isij.4306

Oświadczamy, że niniejsza publikacja powstała przy następującym udziale procentowym autorów:

1.	Marcin Niemiec	25%	podpis	Marcin Niemiec
2.	Andrzej Dziech	25%	podpis	
3.	Miłosz Stypiński	25%	podpis	M Stypiński
4.	Jan Derkacz	25%	podpis	

Oświadczenie udziałów procentowych w publikacji

"Synchronization of Tree Parity Machines Using Nonbinary Input Vectors"; Miłosz Stypiński, Marcin Niemie; IEEE Transactions on Neural Networks and Learning Systems, 2022; https://doi.org/10.1109/TNNLS.2022.3180197

Oświadczamy, że niniejsza publikacja powstała przy udziale procentowym autorów:

1. Miłosz Stypiński 75% podpis MStypiński
2. Marcin Niemiec 25% podpis Marcin Niemiec

Oświadczenie udziałów procentowych w publikacji

"Impact of Nonbinary Input Vectors on Security of Tree Parity Machine", Miłosz Stypiński, Marcin Niemiec; Multimedia Communications, Services and Security, MCSS 2022. Communications in Computer and Information Science, vol 1689. Springer, Cham; https://doi.org/10.1007/978-3-031-20215-5_8

Oświadczamy, że niniejsza publikacja powstała przy udziale procentowym autorów:

1. Miłosz Stypiński 90% podpis MStypiński
2. Marcin Niemiec 10% podpis Marcin Niemiec

Oświadczenie udziałów procentowych w publikacji

"Security of Neural Network-Based Key Agreement Protocol for Smart Grids"; Stypiński, Miłosz, Marcin Niemiec; Energies 16, no. 10: 3997, 2023; https://doi.org/10.3390/en16103997

Oświadczamy, że niniejsza publikacja powstała przy udziale procentowym autorów:

1. Miłosz Stypiński 80% podpis M Stypiński
2. Marcin Niemiec 20% podpis Marcin Niemiec

Oświadczenie udziałów procentowych w publikacji

"Weight equalization algorithm for tree parity machines", Miłosz Stypiński, Marcin Niemiec, Computer Networks, Volume 257, 2025, 110988, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110988.

Oświadczamy, że niniejsza publikacja powstała przy udziale procentowym autorów:

1. Miłosz Stypiński 80% podpis M Stypiński
2. Marcin Niemiec 20% podpis Marcin Niemiec