Jevil's Encryption Systems

Nadim Kobeissi

Symbolic Software
nadim@symbolic.software

Abstract. Imagine if, given a puzzle, you could encrypt a plaintext detailing the location of a prize reward such that he who solves the puzzle can use this solution to decrypt our prize information, *without knowing the solution to the puzzle yourself.*
The Jevil[1] family of encryption systems is a novel set of real-world encryption systems based on the promising foundation of witness encryption. The first Jevil encryption systems comprise of Pentomino and Sudoku-based encryption, allowing for the encryption of plaintext such that solving a Pentomino or Sudoku puzzle yields to decryption. Jevil encryption systems are shown to be correct, secure and to achieve high performance with modest overhead.

1 Introduction

In 2013, Garg, Gentry, Sahai and Waters published the first formal definition of witness encryption [2]. In their work, witness encryption is defined as an encryption scheme for an NP language L with a corresponding *"witness relation"* R. The idea is that a sender can encrypt a message M to a particular problem instance x, thereby producing a ciphertext C. The recipient can decrypt C and obtain M if x is in the language and they know a witness w where $R(x, w)$ holds.

More practically, let's imagine that Alphys is having trouble solving a Pentomino (Fig. 1) puzzle which has a board N comprised of n squares and a set P comprised of all the Pentomino pieces. Alphys can't figure out the solution to her Pentomino puzzle: she can't fit all the pieces in P such that every square in N is covered exactly once and every piece in P is used only once. However, she knows that her friend Undyne is determined to solve any challenge; using witness encryption, Alphys can encrypt a prize such that anyone who solves the Pentomino puzzle can obtain the encryption key, despite the fact that Alphys herself doesn't know the solution to the puzzle.

[1] The name "Jevil" is inspired by a character in Toby Fox's *Deltarune* fictional universe. In this universe, Jevil is a jailed mad jester that challenges the player to reunite the keys necessary to enter their cell so that they may "play games" of Jevil's design. Jevil is revealed to have achieved surreal powers and to have turned his own jail cell into a world more expansive than the one from which he was locked away. A piano rendition of Jevil's theme is available [1].

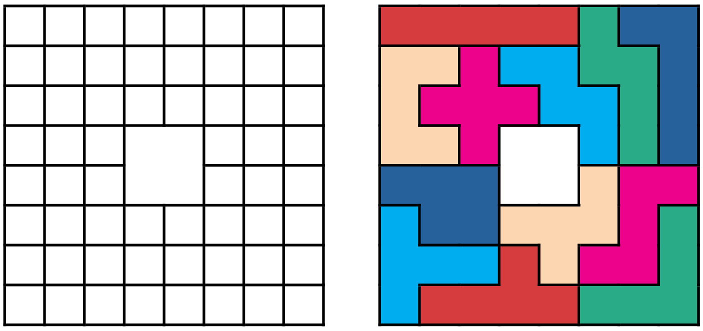

Fig. 1. Pentomino [3] is a puzzle game where a given board must be completely filled by a given set of pieces such that each shape is used exactly once and each square is filled exactly once. In this example, we see a 8x8 board N with the four center squares removed (left) being filled with twelve unique given Pentomino pieces from the set of pieces P (right.) The board and the pieces represent the puzzle while the arrangement on the right represents the solution.

1.1 Related Work

Despite the apparent novelty and uniqueness of the scheme proposed by Garg et al, there appears to have been minimal interest in deriving real-world cryptographic systems from witness encryption. The research posited by Garg et al. relates witness encryption schemes to the exact cover problem, a well-known NP-complete problem where a space and a set of elements are provided and where the goal is to find exactly one subset such that the space is covered fully and without overlap. Pentomino (and other popular puzzles, such as Sudoku) are actually exact cover problems and this paper take advantage of this in order to translate them into correct, secure, usable and performant witness encryption systems.

Shortly after the introduction of witness encryption, Gentry et al expanded upon the security foundations for witness encryption and introduced a new proof framework for proving witness encryption schemes secure under instance independent assumptions [4]. Bellare et al have expanded witness encryption towards potential applications for password-based cryptography [5]. Liu, Jager, Kakvi and Warinschi have expanded witness encryption in order to create "time-lock encryption" [6], where ciphertext can only be decrypted at some specific point in the future (made possible by the existence of a continuous public hash chain, such as the Bitcoin blockchain).

1.2 Contributions

In this work, we introduce the Jevil family of real-world witness encryption systems. We start with practical specifications for witness encryption based Pentomino and Sudoku puzzles. Our hope is to draw more interest in the real-world applicability of witness encryption and to progressively enlarge the Jevil's encryption systems to include more different types of puzzles, including new ones based on NP-complete problems that are not the exact cover problem and the subset-sum problem.

In §2, we cover witness encryption preliminaries as established by Garg et al. Based on these preliminaries, we introduce in §3 Pentomino-based witness encryption and in §4 Sudoku-based witness encryption. We formalize a real-world security model with concrete security bounds and parameters for our encryption systems. §5 presents an argument for the security of the Jevil encryption systems while §6 concludes with a discussion of future work.

2 Preliminaries

Our preliminaries are precisely the same as those established by Garg et al in their original research on witness encryption [2]. We relax the dependence on the definition for an ideal multilinear map [7] in an effort to make our witness encryption primitives more relatable in terms of real-world implementation.

2.1 Exact Cover Problem

The exact cover problem is a well-known NP-complete [8] problem in computer science. Described intuitively, it is the problem of covering some space with a collection of shapes such that no two pieces overlap and such that the space is covered fully.

Usefully, Pentomino, Sudoku and other popular puzzles are reducible to the exact cover problem. Garg et al base themselves around the exact cover problem when describing some elements of their witness encryption schemes [2] and in this work we expand witness encryption based on the exact cover problem in order to achieve real-world, implementable cryptographic systems.

Given an input $x = (n, P_1, \dots, P_l)$ where n is an integer and each $P_i, i \in [l]$ is a subset of $[n]$, our goal is to find a subset of indices $T \subseteq [l]$ that meets both of the following conditions:

1. $\cup_{i \in T} P_i = [n]$
2. $\forall i, j \in T$ where $i \neq j$, $P_i \cap P_j \equiv \emptyset$.

If such a T exists, then it is an exact cover of x.

2.2 *n*-MDDH Assumption

The n-Multilinear Decisional Diffie-Hellman [9] (n-MDDH) problem is defined by Garg et al [2] as the following:

A challenger runs $\mathcal{G}(1^\lambda, n)$ to generate groups and generators of order p. Then it picks a random $s, c_1, \dots, c_n \in \mathbb{Z}_p$. The assumption then states that given $g = g_1, g^s, g^{c_1}, \dots, g^{c_n}$ it is hard to distinguish $T = g_n^{s \prod j \in [1,n]^{c_j}}$ from a random group element in G_n, with better than negligible advantage in security parameter λ.

2.3 Decision Multilinear No-Exact-Cover Assumption

The Decision Multilinear No-Exact-Cover Assumption is defined by Garg et al [2] as the following:

Let $x = (n, P_1, \ldots, P_l)$ be an instance of the exact cover problem that has no solution. Let $\mathtt{param} \leftarrow \mathcal{G}(1^{1+n}, n)$ be a description of a multilinear group family with order $p = p(\lambda)$. Let $a_1, \ldots, a_n, r$ be uniformly random in $\mathbb{Z}_p$. For $i \in [l]$, let $c_i = g_{|P_i|}^{\prod j \in P_i{}^{a_j}}$. For all adversaries $\mathcal{A}$, the distinguishing advantage between the following two distributions is negligible:

$$(\mathtt{param}, c_1, \ldots, c_l, g_n^{a_1 \cdot \ldots \cdot a_n}) \text{ and } (\mathtt{param}, c_1, \ldots, c_l, g_n^r)$$

3 Jevil's Pentomino Encryption System

Jevil's Pentomino Encryption System (JPES) is a witness encryption system with the following public, user-provided components:

- N, a Pentomino board of size n.
- $P = \{P_1, \ldots, P_{63}\}$, a set of Pentomino pieces.[2]

(N, P) together constitute a full description of a specific Pentomino puzzle via its board and its pieces.

Using JPES, the sender generates a key K using (N, P) which can then be used for encryption (or anything else) and a set of public values S. JPES then allows the recipient to obtain K by solving the Pentomino puzzle described by (N, P) and S.

When attempting to transform Pentomino puzzles into a witness encryption system, there are a number of intuitive constraints inherent to the Pentomino puzzle game that we must be able to mathematically capture. In order for a Pentomino puzzle to be considered solved:

- N must be completely filled with no "empty" squares.
- Any one of the 12 pieces in P may not be used more than once and may be used only in a single orientation.
- Squares in N cannot be filled more than once (i.e. pieces in P may not overlap on N).

3.1 Key Generation

Step 1: `JPES-GENBOARDEXP`.

- Choose a prime $p = p(\lambda)$ where λ is a strong security parameter and let g_p be a generator for the group G of prime order p.[3]

[2] If we consider all of the possible 90° rotations of the 12 unique Pentomino piece shapes, we obtain 63 pieces in total.

[3] In practice, we mean that p is a safe prime and that we are working in a secure Diffie-Hellman field. If working with Elliptic-Curve Diffie-Hellman, for example, this could be the prime order for the field of the Curve25519 [10] elliptic curve.

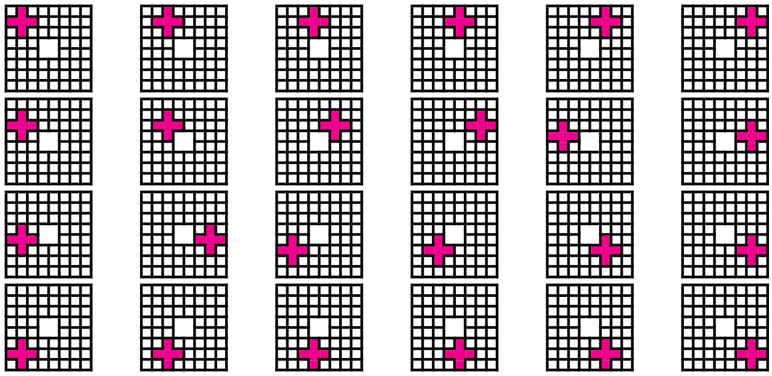

Fig. 2. The JPES-CALCPOSEXP step calculates the exponents for each possible position of a particular Pentomino shape on the board. For example, for piece P_i, the top-left position will generate the Pentomino value $g_p^{P_i^x \cdot N_2^x \cdot N_9^x \cdot N_{10}^x \cdot N_{18}^x}$.

- $\forall i \in N,\ N_i^x \xleftarrow{R} \{0,1\}^\lambda$.
- $\forall i \in P,\ P_i^x \xleftarrow{R} \{0,1\}^\lambda$.[4]
- $K = \mathsf{HASH}(g_p^{(P_1^x \cdot \ldots \cdot P_{12}^x)\cdot(N_1^x \cdot \ldots \cdot N_n^x)})$.[5]

Step 2: `JPES-CALCPOSEXP`**.**

- $S = \{\}$.
- $\forall i \in P$, calculate every possible position of P_i on N as shown in Fig. 2. For each position, insert a triple into the set S which links the resulting exponent with its original shape in P as well as the position on N from which the exponent was derived. For example, the value derived for the top-left position in Fig. 2 would be added to S as $(P_i \times [2, 9, 10, 18] \rightarrow g_p^{P_i^x \cdot N_2^x \cdot N_9^x \cdot N_{10}^x \cdot N_{18}^x})$.

The sender is free to use the symmetric key K to encrypt M as they please. They then send the ciphertext along with public values (p, g_p, N, P, S) to the recipient.

3.2 Key Derivation

The recipient attempts to fit the pieces in P onto N so that they obtain a solution of the exact cover problem as described in §2.1. Once they believe they have a

[4] Elements of P that are rotations of the same piece share the same randomly generated exponent. This means that in total, only 12 exponents are generated for the 63 rotations contained in P.

[5] Here, HASH is any secure hash function, such as for example BLAKE2 [11].

solution, they select from S the subset of exponents $e \subseteq S$ which are related to their positioning of the pieces in P on N and calculate:

$K = \mathsf{HASH}(g_p^{\prod i \in e}) \equiv \mathsf{HASH}(g_p^{(P_1^x \cdot \ldots \cdot P_{12}^x) \cdot (N_1^x \cdot \ldots \cdot N_n^x)})$.

Note that for Pentomino puzzles with no solution, K can never be obtained by the recipient. For Pentomino puzzles with multiple solutions, each solution will result in the same value K.

3.3 Cost and Overhead

Key Generation. Given a security parameter λ of practical size 128 bits (16 bytes), $(n \cdot 16) + (12 \cdot 16)$ pseudorandom bytes must be generated for a board of size n. To put this in perspective, for the puzzle shown in Fig. 1, $(60 \cdot 16) + (12 \cdot 16) = 1152$ pseudorandom bytes must be generated.

$\forall i \in P$, modular exponentiations must be calculated for as many times as P_i is possible to fit in N, which can vary greatly depending on the size and shape of N. To put this in perspective, for the puzzle shown in Fig. 1, ≈ 1500 modular exponentiations must be calculated.[6] Of the values sent to the recipient, (p, g_p, N, P) are of negligible size. S however must contain all of the exponents calculated in this step.

JPES's key generation overhead is considered to be workable when dealing with individual puzzles. However, in the (admittedly difficult to imagine) practical scenario where keys must be generated on many different puzzles successively, a performance bottleneck may be encountered.

Key Derivation. Key derivation costs and overhead are minimal. Essentially, a single modular exponentiation step is carried out, with as many exponents as there are pieces fitted on the board.

4 Jevil's Sudoku Encryption System

Jevil's Sudoku Encryption System (JSES) is a witness encryption system with the following public, user-provided components:

- N, a Sudoku board of size n.
- $P = \{P_1, \ldots, P_9\}$, a set of Sudoku pieces.[7]

(N, P) together constitute a full description of a specific Sudoku puzzle via its board and its pieces.

Using JSES, the sender generates a key K using (N, P) which can then be used for encryption (or anything else) and a set of public values S. JPES then allows the recipient to obtain K by solving the Sudoku puzzle described by (N, P) and S.

[6] Modern implementations of Curve25519 can perform upwards of ≈ 26000 scalar multiplications per second on consumer hardware [12].

[7] $\{P_1, \ldots, P_9\}$ are commonly referred to in a Sudoku puzzle as the numbers 1 to 9.

	1			7			3	
			8			4	6	2
6				5		8		7
	2	5			9			
		8	3					
	7							6
1					2			
	8	3	7					
				4		5		

8	1	2	4	7	6	9	3	5
5	3	7	8	9	1	4	6	2
6	4	9	2	5	3	8	1	7
3	2	5	6	8	9	7	4	1
4	6	8	3	1	7	2	5	9
9	7	1	5	2	4	3	8	6
1	5	4	9	3	2	6	7	8
2	8	3	7	6	5	1	9	4
7	9	6	1	4	8	5	2	3

Fig. 3. Sudoku is a puzzle where, in a 9x9 grid composed of nine 3x3 subgrids, each square must be filled with a number between 1 and 9 such that the number is unique to its row, column and subgrid. An unsolved puzzle is shown on the left with its solution to the right. Sudoku has been shown to be a constraint problem [13].

When attempting to transform Sudoku puzzles (Fig. 3) into a witness encryption system, there are a number of intuitive constraints inherent to the Sudoku puzzle game that we must be able to mathematically capture. In order for a Sudoku puzzle to be considered solved:

- N must be completely filled with no "empty" squares.
- Any column, row and subgrid in N may contain a certain number only once.
- Squares in N cannot be filled more than once (i.e. pieces in P may not overlap on N).

4.1 Key Generation

Step 1: `JSES-GENBOARDEXP`**.**

- Choose a prime $p = p(\lambda)$ where λ is a strong security parameter and let g_p be a generator for the group G of prime order p.
- $\forall i \in N, N_i^x \xleftarrow{R} \{0,1\}^\lambda$.
- $\forall i \in P, P_i^x \xleftarrow{R} \{0,1\}^\lambda$.
- $K = \mathsf{HASH}(g_p^{(P_1^x \cdot \ldots \cdot P_9^x)^{|P|} \cdot (N_1^x \cdot \ldots \cdot N_n^x)^{\frac{|N|}{3}}})$.

Step 2: `JSES-CALCPOSEXP`**.**

- $S = \{\}$.

For all empty squares N_i in N:

- $\forall i \in P$, calculate the relevant exponent as shown in Fig. 4 but for the position of the square N_i. Then, insert a triple into the set S which links the

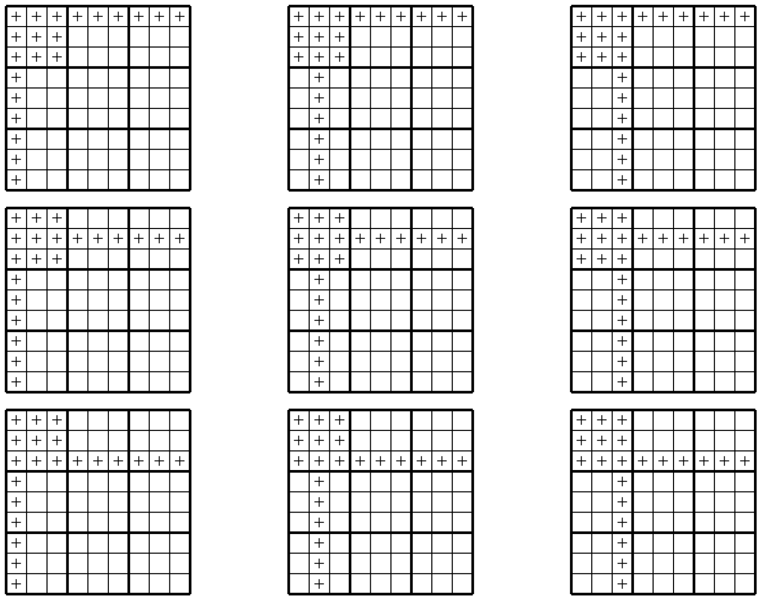

Fig. 4. The `JSES-CALCPOSEXP` step calculates the exponents for each possible "position" of a particular Sudoku piece on the board such that its constraints are captured. For example, for piece P_i when considered in the position col. 2, row 2 (i.e. N_{11}), the value calculated is $g_p^{P_i^x \cdot N_1^x \cdot N_2^x \cdot N_3^x \cdot N_{10}^x \cdot N_{11}^x \cdot N_{12}^x \cdot N_{13}^x \cdot N_{14}^x \cdot N_{15}^x \cdot N_{16}^x \cdot N_{17}^x \cdot N_{18}^x \cdot N_{19}^x \cdot N_{20}^x \cdot N_{21}^x \cdot N_{29}^x \cdot N_{38}^x \cdot N_{47}^x \cdot N_{56}^x \cdot N_{65}^x \cdot N_{74}^x}$. This corresponds to the center example given in this figure, which demonstrates the squares of N involved when calculating exponents for positions N_1, N_2, N_3, N_{10}, N_{11}, N_{12}, N_{19}, N_{20} and N_{21}.

resulting exponent with its original piece in P (i.e. the "number") as well as the position on N. For example, if $N_i = N_{11}$, the following would be added to S:
$(P_i \times 11 \rightarrow g_p^{P_i^x \cdot N_1^x \cdot N_2^x \cdot N_3^x \cdot N_{10}^x \cdot N_{11}^x \cdot N_{12}^x \cdot N_{13}^x \cdot N_{14}^x \cdot N_{15}^x \cdot N_{16}^x \cdot N_{17}^x \cdot N_{18}^x \cdot N_{19}^x \cdot N_{20}^x \cdot N_{21}^x \cdot N_{29}^x \cdot N_{38}^x \cdot N_{47}^x \cdot N_{56}^x \cdot N_{65}^x \cdot N_{74}^x})$.

For all filled squares in N (we take as an example col. 2, row 1 in Fig. 3, i.e. N_2):

– Since the square contains the piece P_1, Calculate the relevant exponent as shown in Fig. 4 but for the position col. 2, row 1. Do so only for P_1. Then, insert a triple into the set S which links the resulting exponent with its original piece in P (i.e. the "number") as well as the position on N. For example, the value derived for this filled square would be added to S as:
$(P_1 \times 2 \rightarrow g_p^{P_1^x \cdot N_1^x \cdot N_2^x \cdot N_3^x \cdot N_4^x \cdot N_5^x \cdot N_6^x \cdot N_7^x \cdot N_8^x \cdot N_9^x \cdot N_{10}^x \cdot N_{11}^x \cdot N_{12}^x \cdot N_{19}^x \cdot N_{20}^x \cdot N_{21}^x \cdot N_{29}^x \cdot N_{38}^x \cdot N_{47}^x \cdot N_{56}^x \cdot N_{65}^x \cdot N_{74}^x})$.

4.2 Key Derivation

The recipient attempts to fit the pieces in P onto N so that they obtain a solution of the exact cover problem as described in §2.1. Once they believe they have a solution, they select from S the subset of exponents $e \subseteq S$ which are related to their positioning of the pieces in P on N and calculate:

$K = \mathsf{HASH}(g_p^{\prod i \in e}) \equiv \mathsf{HASH}(g_p^{(P_1^x \cdot \ldots \cdot P_9^x)^{|P|} \cdot (N_1^x \cdot \ldots \cdot N_n^x)^{\frac{|N|}{3}}})$.

Note that for Sudoku puzzles with no solution, K can never be obtained by the recipient. For Sudoku puzzles with multiple solutions, each solution will result in the same value K.

4.3 Cost and Overhead

Key Generation. Given a security parameter λ of practical size 128 bits (16 bytes), $(n \cdot 16) + (|P| \cdot 16)$ pseudorandom bytes must be generated for a board of size n. To put this in perspective, for a typical Sudoku puzzle where $n = 81$ and $|P| = 9$, $(81 \cdot 16) + (9 \cdot 16) = 1440$ pseudorandom bytes must be generated.

$\forall i \in P$, modular exponentiations must be calculated for as many times as P_i is possible to fit in N. To put this in perspective, for a typical Sudoku puzzle, $(n = 81) \cdot (|P| = 9) = 729$ modular exponentiations must be calculated which each exponent containing 28 multiples.[8] Of the values sent to the recipient, (p, g_p, N, P) are of negligible size. S however must contain all of the exponents calculated in this step.

JSES's key generation overhead is considered to be workable when dealing with individual puzzles. However, in the (admittedly difficult to imagine) practical scenario where keys must be generated on many different puzzles successively, a performance bottleneck may be encountered.

Key Derivation. Key derivation costs and overhead are minimal. Essentially, a single modular exponentiation step is carried out, with as many exponents as there are pieces fitted on the board.

5 Security Argument

JPES (§3) and JSES (§4)'s security is founded entirely on the n-MDDH problem (§2.2) and on the Decision Multilinear No-Exact-Cover Assumption (§2.3). When considering the security of both systems, we concern ourselves mainly with the claim that K can only be obtained by a party with knowledge of sets (N, P, S) if and only if the values within S are organized in such a way that a solution is obtained for the puzzle described by (N, P).

[8] As previously noted in §3.3, implementations of Curve25519 can perform upwards of ≈ 26000 scalar multiplications per second on consumer hardware [12].

5.1 Secrecy of JPES and JSES

Once K is generated using JSES or JPES, it is up to the user to employ that value using their preferred symmetric encryption cipher[9], which would have its own set of security properties.

If the n-MDDH problem is hard, if the Decision Multilinear No-Exact-Cover Assumption holds and if the generator of K is honest, our argument is that the solver is constrained exclusively to $e \subseteq S$ in order to obtain K. We term this as *secrecy* for JPES and SES.

5.2 Correctness of JPES and JSES

If the obtainability of K is accepted to be dependent on a *"correct arrangement"* e of elements in S, the validity of our security argument shifts to become based on whether the values within S are generated such that e does indeed always represent a valid solution to the puzzle described by (N, P), and that no incorrect or invalid solution to the puzzle (N, P) yields K. We term this as *correctness* for JPES and JSES.

For JPES, K is calculated such that every exponent representing a piece (i.e. $(P_1^x \dots P_{|P|}^x)$) is contained once, and every square exponent $(N_1^x \dots N_{|N|}^x)$ is also contained once. Therefore, given that all exponents are random and given that the n-MDDH assumption holds:

- $g_p^{\prod i \in e}$ not containing an exponent of each unique piece in P used in at least one rotation cannot lead to K since K contains each piece exponent exactly once.
- $g_p^{\prod i \in e}$ containing a piece in P used more than once, or used in different rotations, cannot lead to K since K contains each piece exponent exactly once.
- $g_p^{\prod i \in e}$ not containing an exponent for a particular square in N cannot lead to K since K contains each square exponent exactly once.
- $g_p^{\prod i \in e}$ containing an exponent for a particular square in N more than once, cannot lead to K since K contains each square exponent exactly once.

Given that S contains only elements that multiply piece exponents with valid square exponents, the above points argue that K cannot be obtained unless each unique piece in P is used exactly once and in one rotation, and unless every square in N is covered exactly once, which fulfills the traditional problem definition for Pentomino.

For JSES, K is calculated such that every exponent representing a piece (i.e. $(P_1^x \dots P_{|P|}^x)$) is contained $|P|$ times, and every square exponent $(N_1^x \dots N_{|N|}^x)$ is also contained $\frac{|N|}{3}$ times. Therefore, given that all exponents are random and given that the n-MDDH assumption holds:

[9] Any modern symmetric cipher, such as AES [14] or ChaCha20 [15], is suitable.

- $g_p^{\prod i \in e}$ not containing an exponent of each unique piece in P used exactly $|P|$ times cannot lead to K since K contains each piece exponent exactly $|P|$ times.
- $g_p^{\prod i \in e}$ not containing an exponent for a particular square in N $\frac{|N|}{3}$ times cannot lead to K since K contains each square exponent exactly once. This is meant to capture that we mean for a solution chosen from S to "cover" each square in N exactly $\frac{|N|}{3}$ times.

Given that S contains only elements that multiply piece exponents with all the square exponents such that they "cover" all of the squares that coincide with the rules of Sudoku (a piece not occuring twice in the same column, row or subgrid), the above points argue that K cannot be obtained unless each unique piece in P is used exactly $|P|$ times and such that the rules of Sudoku are respected.

6 Discussion and Conclusion

In this work, we introduce the Jevil family of encryption systems and show that it is possible to generalize witness encryption into relatable, interesting real-world applications based on popular puzzles. The underlying promise of witness encryption is truly interesting: being able to propose a decryption based on an unsolved problem could have serious consequences in the realm of finance and, if sufficiently generalized, could profoundly affect how trustless trade occurs.

Adding new systems to the Jevil family of encryption systems and growing that generalization should be the next step. Ideally, systems based on other fundamental NP-complete problems, such as the subset-sum problem, would be added. Practical, usable implementations should also be a focus especially given the simplicity of the systems within the Jevil family. Finally, a great question can be seen on the horizon: is there a formal language for automatically translating *any* NP-complete problem, once described, into a practical witness encryption system?

We plan to keep track of Jevil development at this webpage: https://jevil.info.

Acknowledgements

This paper is dedicated to composer Toby Fox and electric guitar player RichaadEB.

Bibliography

[1] Yvonne Van What and Toby Fox. The World Revolving (Jevil Secret Boss Theme) Piano Tutorial. YouTube, 2018. https://www.youtube.com/watch?v=yuZFH03d9Tg.

[2] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and its Applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476. ACM, 2013.

[3] William Kent Wadsworth. Pentomino Puzzles, June 22 1976. US Patent 3,964,749.

[4] Craig Gentry, Allison Lewko, and Brent Waters. Witness Encryption from Instance Independent Assumptions. In *International Cryptology Conference*, pages 426–443. Springer, 2014.

[5] Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-based Cryptography. In *IACR International Workshop on Public Key Cryptography*, pages 308–331. Springer, 2015.

[6] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. How to Build Time Lock Encryption. Cryptology ePrint Archive, Report 2015/482, 2015. https://eprint.iacr.org/2015/482.

[7] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate Multilinear Maps from Ideal Lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–17. Springer, 2013.

[8] Michael R Garey and David S Johnson. *Computers and Intractability*, volume 29. W.H. Freeman, New York, 2002.

[9] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[10] Daniel J Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.

[11] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-OHearn, and Christian Winnerlein. BLAKE2: Simpler, Smaller, Fast as MD5. In *International Conference on Applied Cryptography and Network Security*, pages 119–135. Springer, 2013.

[12] Jason A. Donenfeld. kBench9000: Simple kernel land cycle counter. ZX2C4, 2018. https://git.zx2c4.com/kbench9000/about/.

[13] Helmut Simonis. Sudoku as a Constraint Problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer, 2005.

[14] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[15] Daniel J Bernstein. Chacha, a variant of salsa20. In *Workshop Record of SASC*, volume 8, pages 3–5, 2008.